AI Based Position Estimation

Itay Geva and Avichay Ashur

Introduction

Traditional navigation is based on triangulation of known antennas (GPS). In this method the antennas transmit their location and time of transmission, and the receiver uses this information to calculate the distance to the known location. This gives a sphere around the locations of the antennas at time of transmission. The intersection of three spheres is a single point, and thus we need to recognize at least three antennas to use this system.

GPS systems are very accurate but not reliable. Since they have a known frequency of operation they can easily be blocked (large noise signal at the same frequency to cover the transmissions). This and more have led researchers over the years to find other methods for finding location with accuracy.

One such team was our reference research, which was conducted as a part of a Hackathon at SONY. In their project, the researchers wanted to use the antennas that are already part of modern smartphones and use the signals they receive from existing antennas in order to find the location.

Obtaining Data and Pre-Processing

In the reference project, the researchers went out to a neighborhood in Tel-Aviv and recorded with an RF -catcher the signals in the air in different positions in the neighborhood. The following image details the routes taken in the recording.

Figure 1: Routes of Recording

The data was very sparse due to different antenna' properties at different positions, i.e the antennas transmitted in different ranges and the RF-catcher recorded them all. The data was then pre-processed by the research team, so the scale of each feature is the same and more processing that is deeply connected to the nature of the data[1]. In the end they obtained a feature vector that is 96 feature long.

Reference Architecture and Results

In the original research they tried two main methods, and variation upon them. The first was FC-NN with 6 hidden layers. The second was a decision trees method called CatBoostRegressor, which is a method designed for sparse data. The results they obtained are the following:

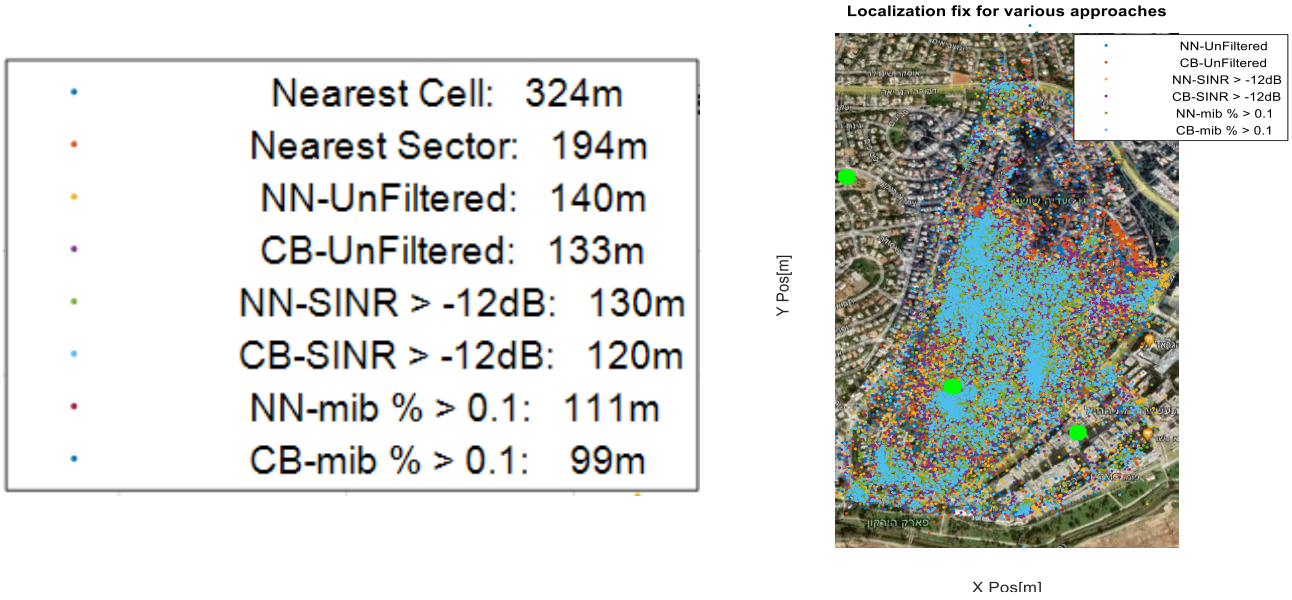

Figure 2 Reference project results, the variations on the base model are with manipulation of data which we did not do.

We can see that they got better results with decision trees, and that their best result is of 99m average deviation.

Understanding the Data

When we first got the data, we decided we do not know enough about the nature of the raw data to try and work directly from it. However, after speaking to the original project's

[1] We considered further pre-processing (as can be seen later in the document) but after experimentation realized we do not have the RF background required to either: start from raw data and do our own pre-processing, or try to do feature normalization on the pre-processed data.

creators about what the features mean, we understood that there might be an inert local information in the data, i.e we thought that different antennas correspond to different features, and thus it is highly imported to keep the order of the features (like in a time series).

In order to confirm or deny this notion we ran some statistical evaluation on the data set. In this report we will show some of those. First we looked at a simple histogram for each feature:

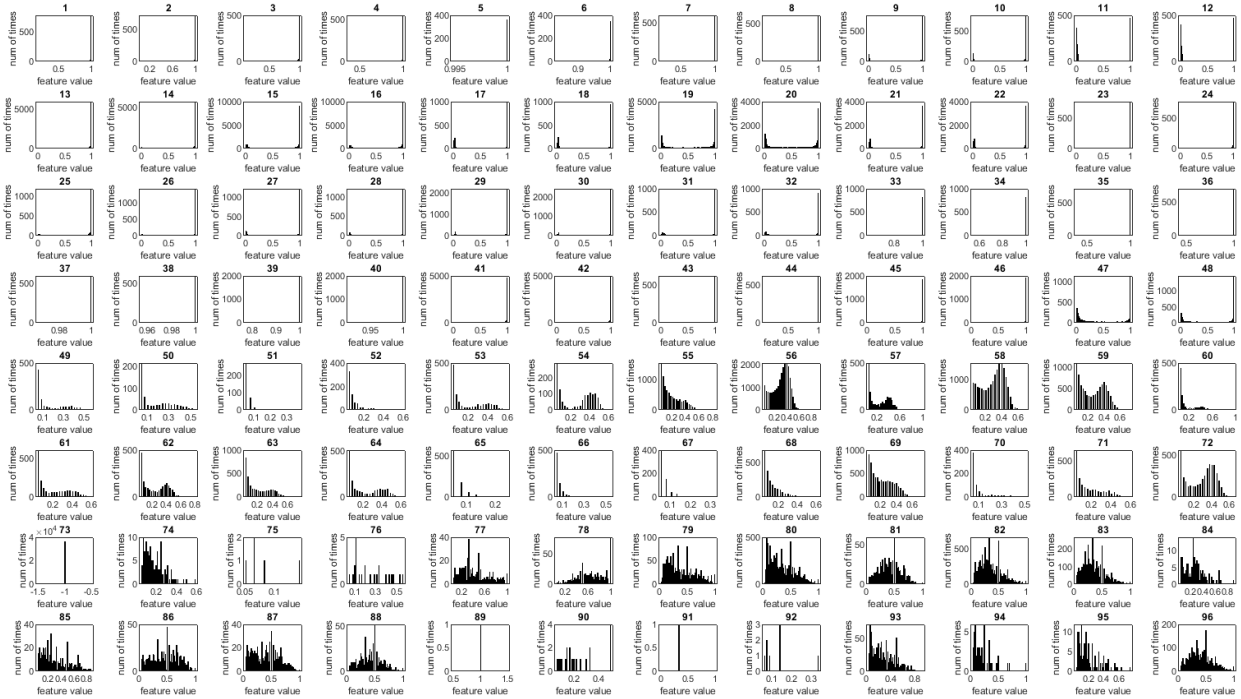

Figure 3: Histograms of features. Graph name indicates feature number.

We can see that the features vary a lot. In addition we recorded the number of “activations” each feature had- the number of times it was not zero:

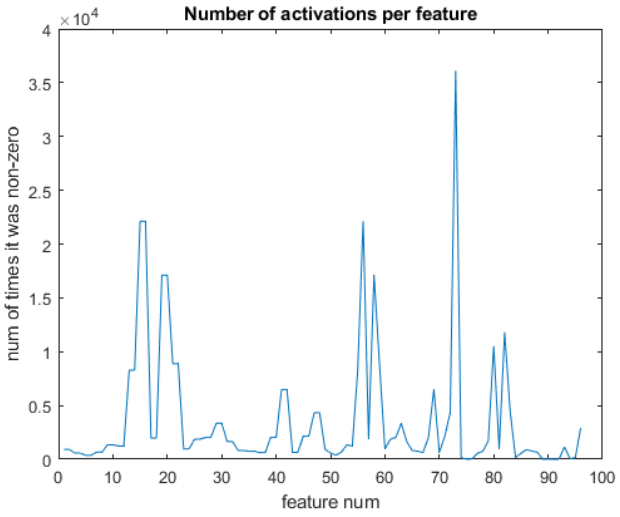

Figure 4: number of activations per feature.

We had thought about using this graph to perform feature normalization, but realized that again, we can't know if the less active features contain more data, because we don't know enough about them.

Lastly to check the locality we calculated the mean of all features when the i-th feature is maximal. In this we try to create a "typical sample" when each feature is most dominant. Locality in such graph will translate as the same group of features being fired together across all members of the group (their graph will be similar). The graph we got is the following:

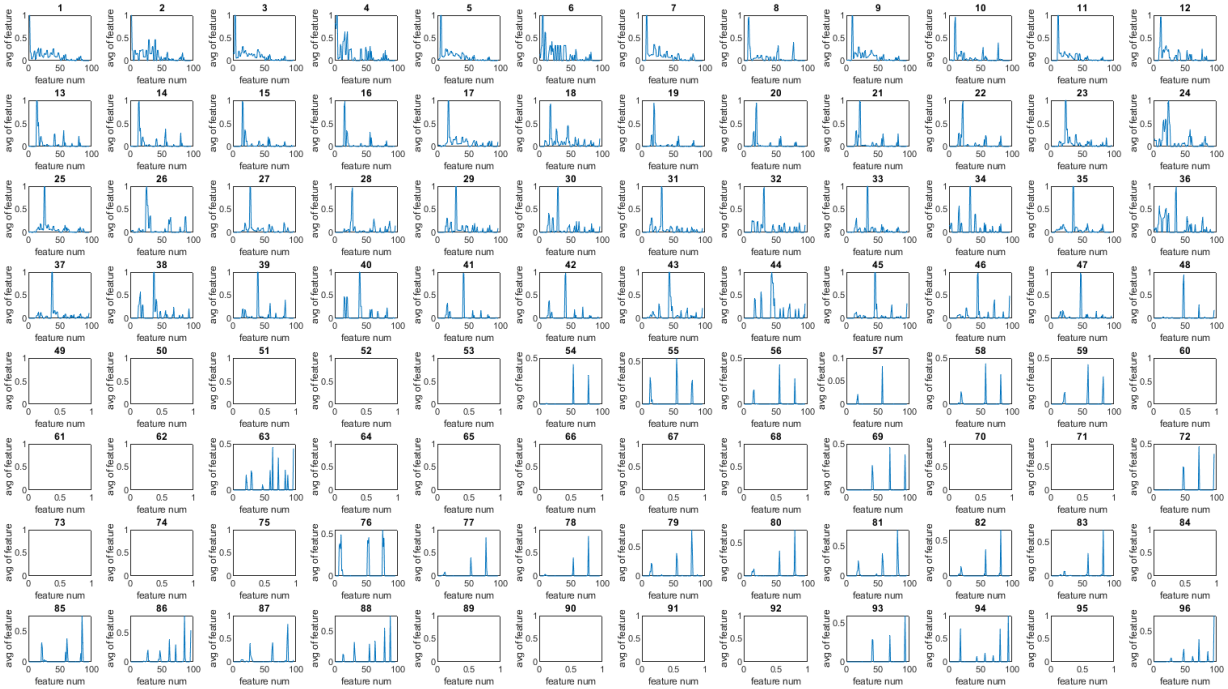

Figure 5: "typical sample" when the i-th feature is most dominant.

We can observe some locality in the later features, but a firing of almost all features the early graphs. We will explain what we did in the model to try and tackle this observation.

Model Architecture

Early on we decided on an approach to do a simple CNN and then change it in accordance with its performance. The performance was evaluated with Euclidean Distance, which in our case is also MSE. Here is the architecture we ended up with:

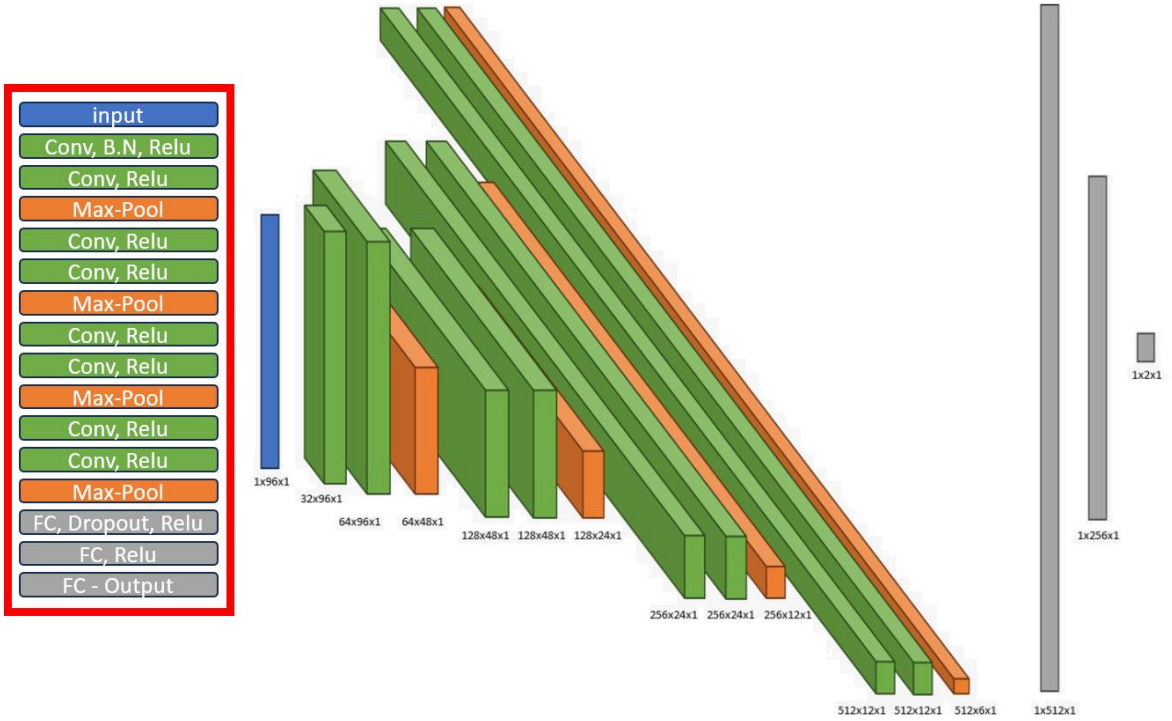

Figure 6: our model architectures.

We can see that the input is the 96 feature vector, and output is a 2x1 vector that represents x position and y position. We will now explain some of the architectures we tried before landing on this one. Since we saw some locality in the data, and convolution kernels do not handle locality well, we thought of two approaches to solve this. The first is to put a FC layer before the CNN. Once this yielded worse results, we tried a skip of the original input to the first FC layer. This did not work well. We then tried to do the skip with a FC layer in between. This yielded similar results to the regular design, but in the end we decided to stick with the simple design due to slightly better results.

Besides architecture, we also optimized (manually) the layout for batch normalizations and the layout for dropout. In addition we also optimized the learning rate (with and without scheduler) and batch size. We used ADAM and MSE loss.

We tried to perform augmentations manually but this yielded worse results.

Results

We ran the model with the chosen hyperparameters until it converged (about 500 epochs) and here are the results we obtained:

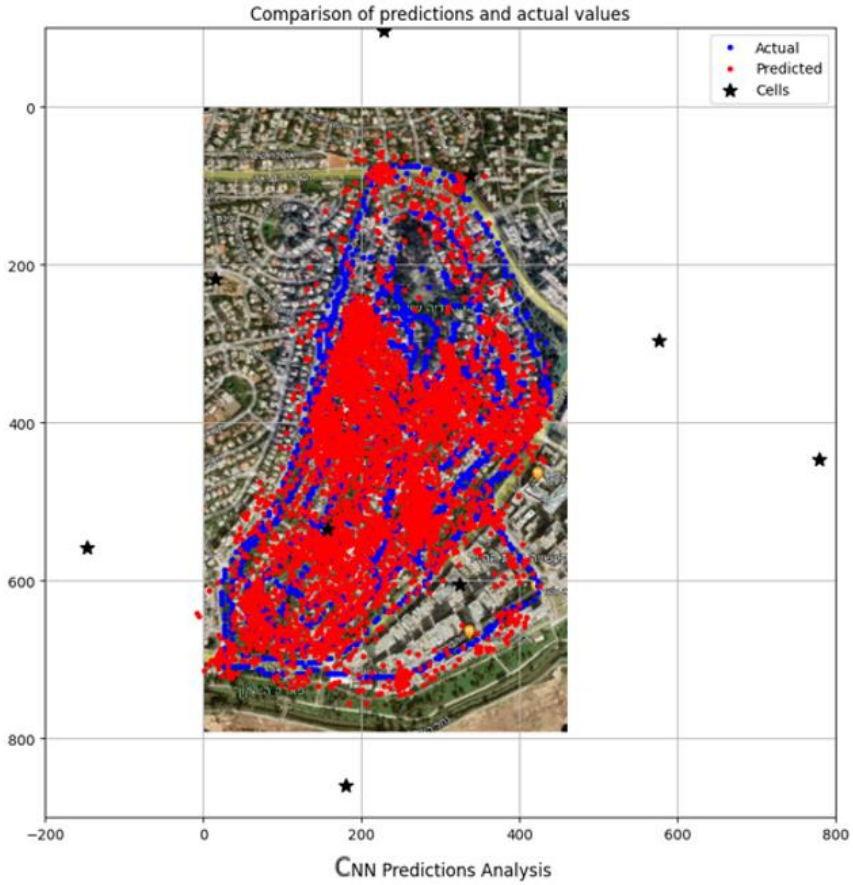

Figure 7: actual vs predicted location. axis in meters.

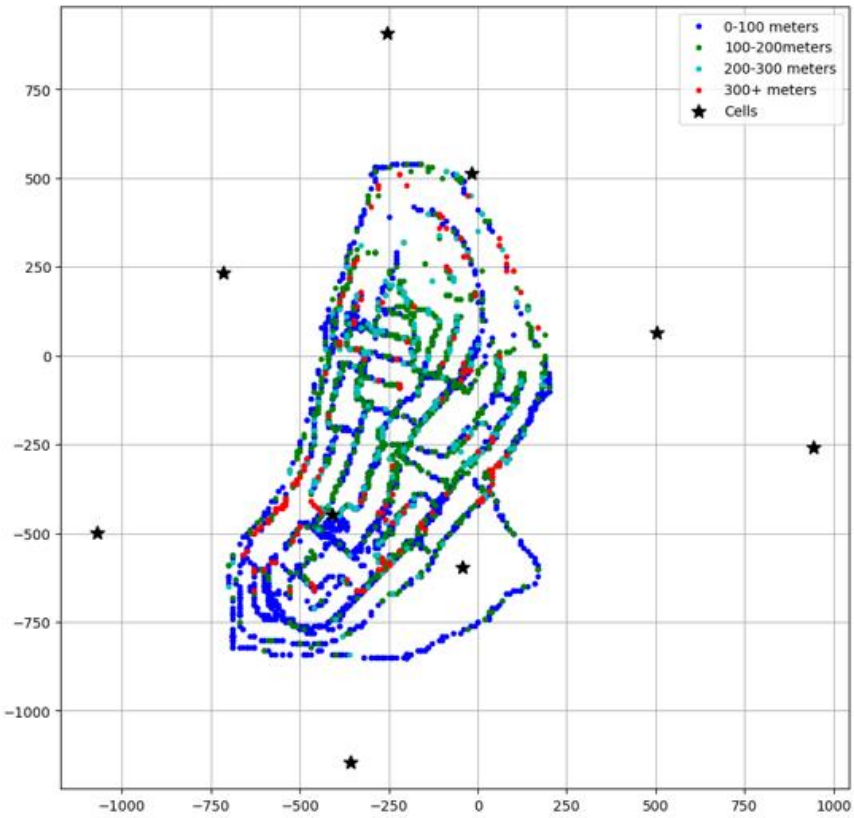

Figure 8: actual locations of measurement, color represent distance of predicted to actual location.

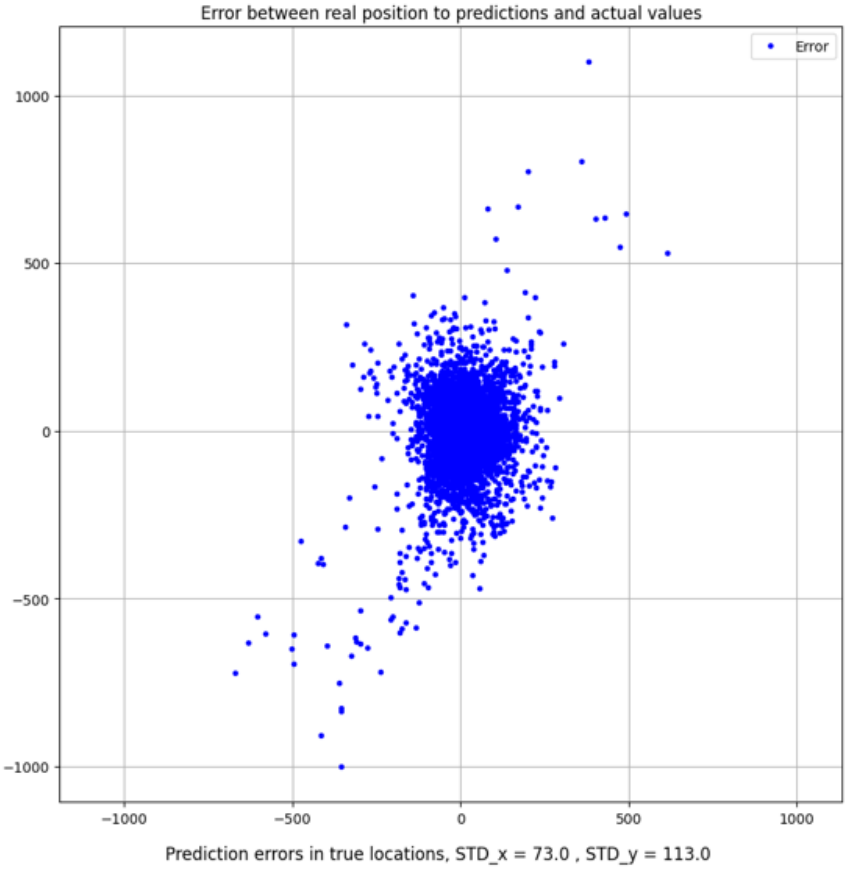

Figure 9: error between real and predicted positions in meters

Comparison to Previous Works

The following table compares the results of our model to the original project's models:

Source	Model	σ_x[m] ↓	σ_y[m] ↓	$\sigma[m]$ ↓	Test loss ↓
GPS	Nearest Cell	---	---	324	---
GPS	Nearest Sector	---	---	194	---
Sony Team	FC NN, 5 layers residuals	78.0	130.0	111	0.0115
Sony Team	CatBoostRegressor	76.0	118.5	99.5	0.0096
Our Project	CNN, 11 layers residuals	**73.0**	**113.0**	**95.1**	**0.0086**

Table 1:comparison between results of our model to the models suggested in the original project.

We can see that we achieve better than the original project in every metric.

Future Works

We believe a lot can be done to better our results, but it requires more time and knowledge of the underlying physics. Firstly we suggest understanding if and how feature normalization should be performed on this model. Second we think that an augmentation method other than small gaussian noise needs to be considered and evaluated. Lastly, we believe that if we had more experience in the field we could have come up with a better architecture.

Acknowledgements

We would like to thank Sony Israel for giving us access to the reference project, including all data and code, and allowing us to use it in our project.

Secondly we want to thank our course TA, Tal Daniel, for advising us on the model and writing of this report.