Next Position Prediction using LSTM Neural Networks

Conference Paper · September 2020

DOI: 10.1145/3411408.3411426

5 authors, including:

John Violos
National Technical University of Athens
39 PUBLICATIONS 271 CITATIONS
SEE PROFILE

Stylianos Tsanakas
11 PUBLICATIONS 29 CITATIONS
SEE PROFILE

Georgios Palaiokrassas
Yale University
21 PUBLICATIONS 142 CITATIONS
SEE PROFILE

Theodora Varvarigou
National Technical University of Athens
400 PUBLICATIONS 4,890 CITATIONS
SEE PROFILE

Next Position Prediction using LSTM Neural Networks

John Violos
violos@mail.ntua.gr
Electrical and Computer Engineering, National Technical University of Athens

Stylianos Tsanakas
el09727@mail.ntua.gr
Electrical and Computer Engineering, National Technical University of Athens

Maro Androutsopoulou
mandroutsopoulou@mail.ntua.gr
Electrical and Computer Engineering, National Technical University of Athens

Georgios Palaiokrassas
geopal@mail.ntua.gr
Electrical and Computer Engineering, National Technical University of Athens

Theodora Varvarigou
dora@telecom.ntua.gr
Electrical and Computer Engineering, National Technical University of Athens

ABSTRACT

Movement data is a valuable source of information in the context of many applications. Deep Learning (DL) provides useful methods for the modeling and the knowledge extraction from them. An adaptation of a DL approach to time series analysis can improve significantly the prediction accuracy with the cost of increasing the training time and consequently the resource demands. In this paper we propose the use of Artificial Neural Networks (ANN) with LSTM layers for the next position prediction of moving objects using a genetic algorithm and a transfer learning method. The genetic algorithm makes a smart search in the the hypothesis space to estimate a close to optimal ANN architecture and the model leverages a transfer learning method to exploit a repository of well trained ANN models in order to speed up the training process. We evaluated our proposed model with real data from the trajectory of various vessels and compared with a state of the art trajectory prediction method and two well-known AutoML meta models. The experimental results showed that the proposed model has better accuracy than other models and the transfer learning process decreases dramatically the training time. The results encourage us for the applicability of our method.

CCS CONCEPTS

• **Machine learning approaches** → **Neural networks**; • **Information systems applications** → **Spatial-temporal systems**; • **Bio-inspired approaches** → Genetic algorithms.

KEYWORDS

Long Short-Term Memory, Deep Learning, Trajectory Prediction, Hyperparameter Optimization, Transfer Learning

ACM Reference Format:
John Violos, Stylianos Tsanakas, Maro Androutsopoulou, Georgios Palaiokrassas, and Theodora Varvarigou. 2020. Next Position Prediction using LSTM Neural Networks. In *SETN 2020: 11th Hellenic Conference on Artificial Intelligence, September 02–04, 2020, Athens, Greece.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

The next position prediction for a moving object is a topic of high interest in many domains such as transportation, security, data migration and smart cities. An analysis of how an object was moving in previous time steps can be used to build a prediction model for the next time steps positions. Many types of prediction models based on traditional machine learning algorithms have been successfully used.

The advent of the Deep Learning (DL) era has made us to rethink, redesign and conduct research in many prediction applications that use traditional machine learning techniques. In addition, the sequence of positions over time of a moving object can be modelled as a time series problem. The idea of an Artificial Neural Network (ANN) approach and the applicability of time series in the context of the next position prediction made us to propose the use of Long Short-Term Memory (LSTM) layers.

An LSTM neuron in contrast with regular neurons have units with memory that perceive the order of observations and can learn temporal dependencies. LSTM can have generalization and local smoothing properties if they are trained with painstaking attention to detail. Many DL experts conclude that the accuracy of a DL model has three main factors: The quality of the data that we use, the training methodology that we follow, and the architectural design of the ANN. In order to conclude to an optimal or close to optimal architectural design for a given dataset we can use a genetic algorithm.

An ambition in the DL domain is the use of transfer learning techniques. Many times we have well trained and high accurate models for a set of use cases and we want to expand our knowledge for different but similar use cases. In this context we examine the assumption of making an exhaustive model training in some trajectories and leverage them to make a short but accurate training to new trajectories.

We evaluated our proposed model using real data from various vessels and trajectories. The results are measured using the Vincenty formula and distance bearing instead of latitude and longitude. The results show that the proposed model is practical, effective and outperforms state of the art methods.

The major contributions of this paper is listed as follows:

- We design and build an accurate model for the next position prediction of vessels.
- We demonstrate theoretically and practically the suitability of LSTM ANN for geo-spatial data.
- We experiment how to find a close to optimal ANN architectures with LSTM layers using a genetic algorithm.
- We examine how to improve the training process using a repository of ANN models with a transfer learning approach.

The rest of this document is structured as such: Section 2 briefly reviews the use of trajectory analysis and how we can leverage DL techniques. Section 3 gives a framework how we can apply time series analysis for next position forecasting. Section 4 focus on the design of an LSTM ANN model and the training methodology. Section 5 describes the experimental evaluation and gives an interpretation of the results. Finally, in Section 6, we state the derived conclusions and the future work.

2 RELATED WORK

Movement analysis for a human, vehicle, vessel or a any kind of moving object provides highly valuable information in the context of many applications. Prominent applications are the recommendation of sight-seeings for tourists [1] and the identification of boats that face difficult situations [2]. The analysis, the prediction and the knowledge extraction of movement behaviours belong in the domains of spatio-temporal data mining [3] and trajectory analysis [4].

The techniques to extract knowledge from trajectories can be grouped in the following three machine learning approaches. Firstly, in the classification approach [5] where we assign new coming observations into predefined classes based on a labeled training dataset. Secondly, in the clustering approach [6], where the trajectories are grouped together based on similar features. Finally, in the pattern recognition approach [7] where regularities of movement behaviors are recognized. The framework of knowledge extraction from trajectories can exploit additionally machine learning techniques to refine and prepare data [8] such as feature engineering, data normalization and noise removal.

In this paper, we focus on the problem of predicting the next position of the movement of an object based on its previous positions. This problem has been addressed in the literature with methods such as Markov Chains [9] and well-established machine learning models [10]. Our research is different from the previous because we use an LSTM ANN to forecast the next position and transfer learning to improve the training process.

LSTM is a powerful DL predictive model used effectively in the domain of Times Series and Natural Language Processing [11]. While time series approach has been used to many spatio-temporal applications [12], to our best of our knowledge LSTM ANN have not been used in predicting the next position of a moving object.

LSTM is a special kind of Recurrent Neural Networks (RNN). RNN have been used in the domain of trajectory analysis for the needs of point-based classification [13] and have presented very high predictive outcomes. LSTM is more advanced than RNN because they have special memory units in addition to standard neurons. The memory units keep information from a sequence of data for long periods of time.

DL models involve the synaptic weights and biases, which are the learnable parameters. These parameters are estimated using an optimization process which include back propagation and a gradient descent method such as RMSprop, AdaDelta, and Adagrad [14]. The capacity of the ANN, also known as the hyperparameters is the number of layers and the number of neurons that each layer has. The capacity is often selected based on domain-expertise or in a trial and error method. In our research we use a genetic algorithm for optimum or a close to optimum hyperparameter selection. Genetic algorithms have been used in the past for ANN architecture that contain only sequential layers [15]. To the best of our knowledge this research is the first time they are applied with LSTM layers.

3 TIME SERIES AND DEEP LEARNING APPROACH

The spatio-temporal observations of moving objects have a data sequence structure that can be modeled as time series. The consecutive observations are correlated because every position is depended from the previous positions, the sequential observations are more closely related than observations that are far away. Furthermore, the mobility process follows specific laws of physics and inherent inertia, unless an external force changes the mechanical energy of the moving object.

The next position of a moving object cannot merely be predicted by the movement laws of physics and the previous observations. In real life circumstances the problems are more complicated. For instance, the captain of a vessel may alter speed or direction to avoid a collision. But even if the captain took a decision to navigate in a different direction, the navigation process follows a sequence of steps that can be recognized by machine learning techniques. So, The captain's new decision will have as result wrong inference of the model only in the first few time-steps.

Models that leverage the serial dependence in the observations can enhance the predictive accuracy while we should take into consideration which additional techniques can inference the mobility behaviour. The historical geo-location observations contain valuable pieces of information about the objects mobility, but the learning and the forecasting ability of a model is not based on the memorization of previous circumstances but in the generalization and the prediction under new circumstances.

Moving objects such as vessels have persistence in the sequence of their positions. The next latitude and longitude positions are dependent from the previous, distance and bearing too. In addition the velocity and the direction change gradually in a deterministic way that follow common patterns. LSTM by having a chain-like structure, are able to connect information for long periods and map patterns of movement to the next positions.

Taking into consideration that we need a model capable of leveraging the serial dependence in the observations, learn from movement patterns and generalize in new unknown situations, we propose the use of ANN with LSTM and fully-connected layers.

4 PROPOSED MODEL

The model of next position prediction involves three different pipelines as depicted in Fig. 1. The two of them, the training and the transfer learning are used to build an accurate prediction model leveraging the experience of previous models. The third pipeline namely the inference is to make real time predictions based on the current position of the vessel. The pipelines consist of a sequence of separate processing nodes. The key elements of the processing nodes are described in the next subsections.

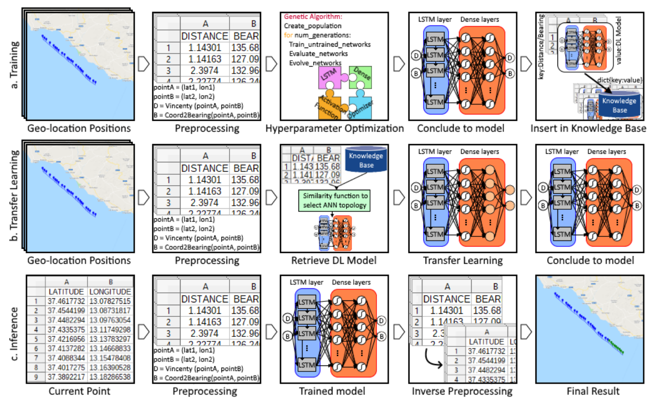

Figure 1: Process

The training pipeline Fig. 1.a takes as input a batch of vessel trajectories. Each trajectory contains a sequence of geolocations i.e. latitude and longitude as depicted in the map. The geolocations are transformed to normalized distance and bearing features. The data features are input in a genetic algorithm that makes a smart search to different ANN topologies in order to conclude to an ANN architecture close to optimal including fully-connected and LSTM layers. This hyperparameter optimization technique is an automated and efficient way that test different architecture configurations. The outcome models are stored in a knowledge base with a mapping data structure which has as key the distance, bearing features and as value the corresponding trained DL model. This map data structure will be used in the transfer learning pipeline to estimate applicable pre-trained DL architectures for new trajectories.

In the transfer learning pipeline Fig. 1.b we take as input a new trajectory, we apply the same preprocessing transformations to get the distance and bearing features. Using a similarity function between the distance bearing features of the new trajectory with the trajectories of the knowledge base we select an ANN topology. In this research we used the Euclidean distance and it is a future work to try different similarity functions. In the retrieved ANN model we apply the transfer learning process which include to keep some layers freeze and retrain the remaining layers with the data of the new trajectory or just to keep the same hyperparameters and retrain all the ANN weights. In the experimental evaluation we experiment both.

The inference pipeline Fig. 1.c uses the output model of transfer learning pipeline. It takes as input the current vessel geolocation, follows the same preprocessing transformations and provides a prediction using the trained deep learning model. In the output of the DL model is applied an inverse transformation to get the latitude and latitude predictions. This process can takes place in real time and for every current position to have the next prediction shaping the trajectory.

4.1 Long Short-Term Memory layer

Long Short-Term Memory (LSTM) is a non-linear time series model that allows information to persist using a memory state and is capable to learn the order dependence between observations in a sequence. LSTM originates from RNN but outperforms it because it can resolve two main weaknesses: the vanishing gradient problem and the short-term memory. The former weakness makes the RNN earlier layers incapable to be trained sufficiently. The latter weakness makes the RNN incapable to carry information from earlier time observations to later ones and as result RNN forgets what is seen in long series.

LSTM consists of a cell state and three gates that regulate the flow of the derived knowledge as illustrated in the Fig. 2. The forget gate is responsible to throw away information from the cell state that is not relevant, the input gate decides the degree that the cell state will be updated and the output gate decides what information will be included in the hidden state and be used for the prediction. All the gates include a sigmoid activation function and trained in a supervised way based on historical observation sequences.

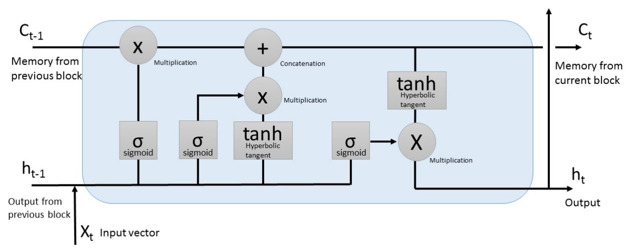

Figure 2: Long Short-Term Memory Unit

LSTM combined with dense layers are building blocks that can be used in different ways to make a forecasting model for the next position of a moving object. The first architecture decision that we should examine is if there will be separate models for the prediction values i.e. the distance and bearing or one composite model, which learns to predict both of them as two separate values in one output layer. Based on previous research in this type of designing decisions [18] and experiments we have seen that composite models can have slightly better prediction results, but they are more demanding in the training process, leading to the need for more epochs with each epoch requiring more time to be completed.

4.2 Data Features and Prepossessing

The data features and the input/output data representation play a key role to the accuracy of the model. We tested different data transforms in the input features and the output values of the ANN as we will see in the section of experimental evaluation. The use of distance and bearing features instead of geolocation with latitude and longitude is a way to improve the accuracy. The distance and bearing measures can be used in the air, land and sea navigation as a different system from GPS coordinates.

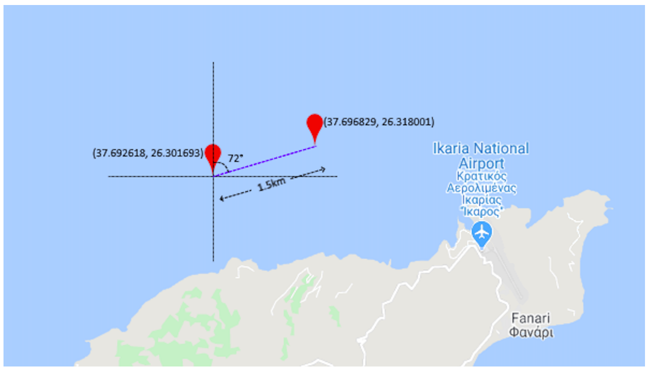

Figure 3: Distance/bearing transformation

In the Fig. 3 are depicted both systems. Latitude and longitude represent a point's distance from the equator and the prime meridian accordingly and they are measured in degrees and values between (-90,90) and (-180,180). In distance/bearing system, the distance is used as the distance between the current and the previous geolocation of the vessel and the bearing is the clockwise angular movement between the two positions. The trigonometric equations that make the transformation from the one system to the other are given in equations 3 and 4.

$$a = sin^2(\Delta\varphi/2) + cos\varphi_1 \cdot cos\varphi_2 \cdot sin^2(\Delta\lambda/2) \quad (1)$$

$$c = 2 \cdot arctan2(\sqrt{a}, \sqrt{(1-a)}) \quad (2)$$

$$Distance = R \cdot c \quad (3)$$

$$\begin{aligned} Bearing = arctan2(&sin(\Delta\lambda) \cdot cos(\varphi_2), \\ &cos(\varphi_1) \cdot sin(\varphi_2) - sin(\varphi_1) \cdot cos(\varphi_2) \cdot cos(\Delta\lambda) \quad (4) \end{aligned}$$

Where R is the earth radius, φ stands for latitude and λ for longitude for points one and two. φ and λ are converted to radiance before they can be used in the equations. θ result is in (-180,180) range so it needs to be transformed in (0,360) range in addition to being converted to degrees.

The advantage of using distance and bearing instead of coordinates can be reasoned by the fact that predicting the next position, the network limits the search space in a way similar to how a human mind would, acknowledging that there is a limited distance (which can be easily estimated by either the previous measurements or velocity and timespan until the next observation). The vessel could travel in the given time to it, putting much focus on trying to estimate the next position in a circle around the original position.

The data features transformed individually by scaling between zero and one. After the inference, we used an inverse transformation in order to have the real values of distance and bearing. Using the current position and the estimated distance and bearing we can calculate the next estimated position in terms of latitude, longitude.

4.3 Learning Process

The parameters of an ANN are estimated using an optimization method that minimizes an objective function. The objective function expresses the difference between the predicted output and the actual output. The most commonly used objective functions are the Mean Absolute Error L1 and the Mean Squared Error L2. In our research we wanted to avoid the large errors so we used L2 as it gives a higher weight to large errors than L1. The literature proposes a rich set of optimization techniques. In our research we tested the Stochastic Gradient Descent (SGD), Root Mean Square Propagation (RMPSrop), Adaptive Gradient Algorithm (Adagrad), its extension Adadelta, the Adaptive Moment Estimation (Adam) and its variants AdaMax, and Nadam which use the Nesterov momentum.

Optimisation methods usually calculate the gradients i.e. the partial derivative of the objective function with respect to the ANN parameters weights and biases. These parameters are updated in the opposite direction of the calculated gradient. This process is repeated until the objective function reaches the minima. The literature has long discussions about not to converge in local minimum but to search for the global minimum. A second important topic in optimization techniques is the convergence speed of the parameters.

SGD performs parameters update for each training example and it is usually fast-convergence technique. But it has the disadvantage that it complicates the convergence and it may overshoot the minima due to the frequent fluctuations. RMSProp and Adam are Stochastic Gradient Descent approaches with an adaptive learning rate. RMSProp has the Momentum approach in which the gradient in every iteration is the sum of the current gradient plus the previous gradients and as result it restricts the oscillation. Adam in a similar way uses the Momentum technique, but it also finds an invariant direction of slope in contrast with the other oscillating directions, with the navigation through saddle points. AdaMax is a special case of Adam where its second-order momentum is replaced by infinite-order momentum.

An additional technique that is used in the learning process is the regularization. Dropout is the most successfully regularization technique but does not work well with LSTM layers [20]. There are alternative regularization methods that apply correctly with LSTM but they are task specific and we decided to leave it for future work to investigate which regularization technique can be applied better to the trajectory prediction.

4.4 Architecture Design with Hyperparameter optimization

The hyperparameters decisions mainly involve the number of layers of an ANN and the number of units that each layer has. In addition, it may also include the type of activation function and the training algorithm. In many cases the hyperparameters are defined based on domain expertise and manually tweaking.

A large number of layers and units may increases the capacity and the memorization of the model, but it may decreases the generalization. From the other side, a small number of layers and units decreases the learning abilities of the model. The decision between large ANN that overfit the data and small ANN that underfit is known as the bias - variance trade-off. An exhaustive searching of all potential configurations is not feasible so we end up in an heuristic method. The use of a genetic algorithm is one of the most prominent approaches for an automatic and systematic search through architecture space.

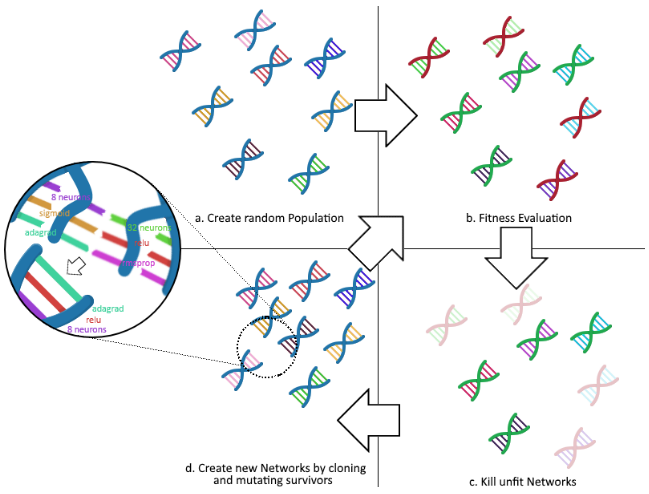

Figure 4: Genetic Algorithm

The main steps of the Generic Algorithm Fig. 4 could be summarized as follows:

(a) Firstly, we pick a population size (N) and generate a population of ANN with random hyperparameters.
(b) Secondly, we a pick number of generations for the algorithm. By iterating over the generations we perform the following steps:
(c) We train and test the entire population and then sort it by some metric
(d) The (B) best ANN advance to the next generation, along with some randomly chosen (R) , while being careful to avoid converging too fast;
(e) Generation of the remaining (N-R-B) members of the population by picking the hyperparameters of the new network choosing from the hyperparameters of two parent ANN, chosen at random from (B U R) with the probability of a random hyperparameter changing to a completely different value (mutation).

4.5 Transfer Learning

The relation between trained models from different time series should be investigated in order to transfer knowledge from one model to the other. The initial approach is to build a predicting model from the first part of positions of the moving object and then predict the next positions. In this approach, we will have a separate model for every trajectory and moving object. A more interesting approach is to investigate the relation between the trajectories. We can initialize the model of a new trajectory with the parameters from historic high accurate trained models. In this way the knowledge from one model is transferred to the next and it also adapted to the new observations. The more advanced approach is if we can have a generic model trained from many different moving objects and trajectories and capable to provide predictions under any circumstances.

5 EXPERIMENTAL EVALUATION

The prediction of next position using the LSTM ANN model has been tested and evaluated with a dataset of thousand trajectories from different vessel types in order to demonstrate its applicability and effectiveness. We tested different variations until conclude to the most accurate setup.

An example of the trajectory prediction of a vessel is depicted in the Fig. 5. The model was trained with first part of the trajectory and afterwards providing each time the current position it predicts the next position in a specific time step. In the figure, we see the true next positions with green colour and the prediction with red.

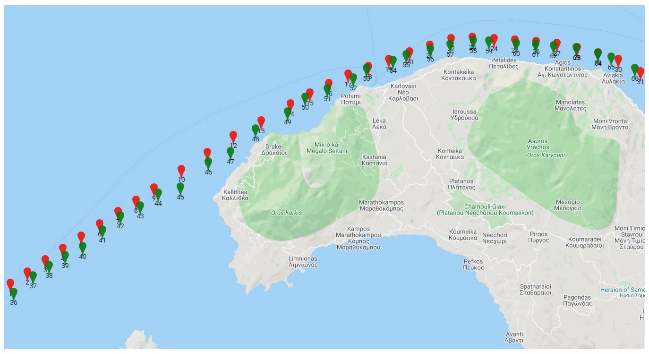

Figure 5: Trajectory predictions

The dataset consists of different types of vessels, timesteps and positions in the latitude and longitude format. For each vessel we have from four hundreds to six thousands position tracks. In the data prepossessing stage, we filtered out many duplicate and invalid values due to the communication problems of the position-mechanism. Vessels, courses and geolocations can be modeled in a convenient way with NumPy arrays and tensor data type of TensorFlow.

5.1 Evaluation Metrics

For the evaluation of the proposed model we measured the discrepancy between the actual next positions from the predictions with the Vincenty formula (Vin_{m}) and the training time ($Train_{\mathrm{sec}}$) of the models. In addition for each model we counted the probability to provide the best predictions (Pr_{best}), the percentage to have a prediction error less than 1km ($Err_{<1\text{km}}$) and less than 0.5km ($Err_{<0.5\text{km}}$). These measures are used in the Table 1 and 2. In Table 2 for the evaluation of the applicability of the transfer learning we also used the The Mean Squared Error (MSE) which measures the average of the squares of the distance and bearing errors.

The Vincenty formula calculates the distance between two given points (Lat, Lon). In the next position prediction problem, it is a better way of evaluating our predictions because it provides a more understandable expression of the error, but mostly because using a more commonly used evaluation metric (such as MSE) when the output is a tuple would fail to correctly distinguish the difference in importance of the two elements, especially in the case of distance/bearing. This means that a slight error in absolute distance has a different effect than a slight error in bearing, and that is not shown when using traditional DL Regression evaluation metrics.

5.2 Comparison with State of the Art Models

In order to evaluate the proposed trajectory prediction LSTM model with a genetic algorithm for hyperparameter optimisation we make two different types of comparisons. The proposed model is compared with the reported results of another state of the art vessel trajectory prediction model which employ traditional machine learning algorithms [10] and with two state of the art machine learning meta models.

In [10] the authors experiment with multiple machine learning models such as Linear Regression, Random Forests and Multi-Layer Perceptions (MLP) and concluded to construct two different MLPs one for predicting latitude and one for longitude. The MLPs consists of one hidden layer, dense connected neurons and the topology constructed with manual trial and error tests. In the time interval of 10 min they reached an average accuracy of 0.65km.

Meta models containing a set of traditional machine learning models, prepossessing techniques, and a mechanism for hyper parameter optimization have been proposed under the umbrella term AutoML [21]. A probabilistic model to capture the relationship between the components of the meta-model, the hyperparameter settings and the model accuracy can make a smart search in the hypothesis space trading off exploration and exploitation in order to conclude to an ensemble model. Explorations stands for searching in areas where the model is uncertain and exploitation for focusing in the space where the model performs well.

Gradient boosted decision trees is also an ensemble method which use a gradient hyperparameter optimization approach to construct an accurate prediction model which consists of many weak estimators. It follows an iterative process in which new models are added to correct the errors made by existing models. The gradient descent algorithm is applied to the loss function in order to select the close to optimal models to be added [22].

5.3 Model Implementation and Frameworks

The Next Position Prediction model using LSTM ANN is implemented and evaluated in Python 3 programming language using the frameworks TensorFlow 2, pandas, NumPy, Scikit-learn, GeoPy, logging and statistics modules in the Jupyter notebook environment of the Google Colaboratory. The knowledge base of the trained models consists of Hierarchical Data Format version 5 (HDF5) files. The experiments' source code is available for any kind of reproduction and reexamination in a publicly accessible URL: https://github.com/STsanakas/trajectory-prediction

Auto-sklearn is an automated machine learning toolkit that makes a pipeline of preprocessing, regression, and hyperparameter tuning using a Bayesian optimization process. It is available in Python 3 and is built on top of the most Scikit-learn models. Auto-sklearn contains a repository of previous optimization runs in order to start training from good saved settings. Auto-sklearn has won the prestigious ChaLearn AutoML challenge, it has been used in many research papers and it is considered by the data scientist community as one of the best AutoML frameworks. This makes us want to experiment with it in the same trajectory task and compare its results using the same dataset.

XGBoost is a software library that implements the model of Gradient boosted decision trees described in the previous subsection. It is available in many programming languages including Python and compatible with Scikit-learn. XGBoost has gained much popularity because it has been used by many Kaggle winning solutions as well as KDD Cup winners. We provided to the XGBoost the same trajectory dataset and constructed two single output Ensemble tree models. One corresponds to predicted distance, and the other to predicted bearing applying a gradient hyperparameter optimization approach and making the same preprocessing transformations as with our proposed model.

5.4 Evaluation Process and Results

For the evaluation we followed the the out-of-sample method that preserve the temporal order of the observations [19] splitting the movement data positions of each trajectory into two sequences. The training sequence and the testing sequence. The training sequence consist of the first 66% geo-positions which used to estimate the most similar model from the knowledge base and train the forecasting model. The testing sequence consists of the next 34% geo-positions and is used for the evaluation with the Vincenty metric. In the context of time series applications we do not use any shuffling mechanism as we make with traditional machine learning evaluation, since it would distort the sequence of observations.

We experimented with two separate ANN LSTM regressors one for distance and one for bearing vs one unified regressor which has two outputs and concluded that the unified model outperforms the two separate. In order to find the close to optimal topologies of the ANN models which will be stored in the knowledge base we used a genetic algorithm. The knowledge base consists of a dictionary with key the preprocessed dataset and value the prediction model.

The genetic algorithm initiates with the training of a population of 14 individuals and 7 generations. After the evaluation of the first, randomly chosen generation is complete, top 50 of the networks advance to the next generation, along with a 10 chance for every other one. The remaining population spots are filled with networks generated by mating two networks that already advanced. Mating consists of randomly picking hyperparameters for the new network from the hyperparameter list of the two parent-networks. This process is repeated until the 7th generation evaluation is complete, where the best performing network is chosen.

The candidate ANN topologies consist of:

- An input layer
- (0-2) LSTM Layers
- (1-5) Dense Layers
- An output layer

The hyperparameters of each LSTM Layer are:

- $lstm_{implementation}$: Implementation mode, either 1 or 2. Mode 1 will structure its operations as a larger number of smaller dot products and additions, whereas mode 2 will batch them into fewer, larger operations. These modes will have different performance profiles on different hardware and for different applications.
- $lstm_{units}$: Dimensionality of the output space with values: (2, 8, 16, 32, 64, 128)
- $lstm_{activation}$: Activation function with values: (tanh, sigmoid, relu, linear, $hard_{sigmoid}$)
- $recurrent_{activation}$: Activation function to use for the recurrent step with values: (tanh, sigmoid, relu, linear, $hard_{sigmoid}$)

The hyperparameters of each Dense Layer are:

- $number_{neurons}$: Dimensionality of the output space with values: (2, 8, 16, 32, 64, 128)
- *activation*: Activation function to use with values: (tanh, sigmoid, relu, linear)

The network's hyperparameters include:

- $LSTM_{Layers}$: Number of LSTM Layers the network with values: (0,1,2)
- $Dense_{Layers}$: Number of Dense Layers the network with values: (1, 2, 3, 4, 5)
- *Optimizer*: The network's training optimizer with values: (rmsprop, adam, sgd, adagrad, adadelta, adamax, nadam)

The XGBoost also takes as input the distance and Bearing between the last two points but calculates two different ensemble models. One corresponds to predicted distance, and the other to predicted bearing to the original point. XGBoost made a smart search through 20736 combinations of trees with the hyperparameters:

- *gamma*: with values: (0, 0.2, 0.4) Minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be.
- $depth_{max}$: values:(3, 5, 10, 15) Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit.
- $childWeight_{min}$: values:(3, 7) Minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than $childWeight_{min}$, then the building process will give up further partitioning. In linear regression task, this simply corresponds to minimum number of instances needed to be in each node. The larger $childWeight_{min}$ is, the more conservative the algorithm will be.
- $colsample_{bytree}$: values:(0.3, 0.7) $colsample_{bytree}$ is the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed.

Table 1: Evaluation of Models

Method	Vin_{m}	$Train_{\mathrm{sec}}$	Pr_{best}	$Err_{<1\mathrm{km}}$	$Err_{<0.5\mathrm{km}}$
		Plain Coordinates			
DL Genet	2180	7416	12.75	60.78	27.45
		Distance and Bearing			
AutoSKL	1584	720	8.82	54.58	32.35
XGBoost	578	2519	15.36	90.20	49.02
DL Genet	421	3585	63.07	97.39	70.92

$learning_{rate}$: *values*:(0.05, 0.15, 0.3)

Auto-sklearn provides out-of-the-box supervised machine learning searching for the right learning algorithm and optimizes its hyperparameters with only configuration the training time. We experimented with different training times for different trajectories and concluded that there was not any accuracy improvement with training times larger than 12 mins.

5.4.1 Outcome models. We summarize the various setups of the proposed model and a comparison with the existing models in Table 1. The experimental evaluation showed that using the distance and bearing of the last known position to predict the next position seemed to produce much better results than plain coordinates. By confining the search space to a circle around the current position, similarly to how a human mind would treat this problem, it outperformed ANNs that tried to estimate the raw coordinates of the next position.

Comparing the results between Gradient Boosted Trees, AutoML and DL Networks trained with the Genetic algorithm, the latter performed better with the majority of its results being under 0.5km of mean error. It is mentioned as a significant improvement comparing with the state of the art that the mean error in proposed model for 10 minute interval is 421 m while in the research [10] was 650 m. In addition the proposed model outperforms the other AutoML models. Its adaptability allows us to adjust its use according to our needs, simply by increasing/decreasing the number of generations or the population size, catering to either more precision or less time needed.

Regarding the prediction time, the response time of the first prediction varies in a range from 50 msec to 300 msec. The exact number depends from the number of the ANN layers. All the next response times as depicted in Table 2 are close to 25,5 msec in case of a single prediction ($TestS_{\mathrm{ms}}$) and 27,5 msec for a batch prediction ($TestB_{\mathrm{ms}}$) of 100 instances. These numbers are profiled during the use of the model in the environment of colaboratory which runs in the google cloud and locally in commodity computers with slightly differences. This response time is sufficiently low, it is lower than the response time reported in [10] so we didn't continue our research work to improve it more. From the other side, we cannot overlook that the training time in our proposed model is considerably longer than the other models. The reason is the multiple hyperparameters and their combinations that the genetic algorithm examines. After examined the output models we noticed that exist groups with common patterns in their topologies. This should be investigate more as future work but a possible interpretation is that same type

Table 2: Training and Prediction Time

Method	$MSE_{10^{-6}}$	$TestS_{\text{ms}}$	$TestB_{\text{ms}}$	$Train_{\text{sec}}$
Original Train	22898	25.66	27.70	3585
Model Retrain	23264	25.57	27.12	66
Transfer Train	23341	25.40	26.99	33

of vessels or navigations may produce similar types of ANNs. As our next goal was to find a viable solution to decrease the training time we tried to leverage this phenomenon.

We made the next set of experiments after we built a feasible amount of models. In this case, instead of using the genetic algorithm to initialize the model, we selected the hyperparameters of a well-trained model that it is estimated to exhibit the same pattern in its topology. This process is known in the literature as transfer learning and the next subsection describes our results.

5.4.2 Transfer Learning. In the Transfer Learning process, we speed up the training process with a little sacrifice in accuracy. Instead of searching for a close to optimal ANN topology using the genetic algorithm and training the model from scratch, we initialized the training process with a pre-trained model and examined two different types of transfer learning.

In Model Retrain, we take as granted the hyperparameters of the model and make a complete training to the parameters, namely the weights and biases. In Transfer Train, we freeze some layers and retrained the remaining layers with the new trajectory data. The retrained layers had been initialized with the weights of the knowledge base model. The performance of these models are presented in the Table 2. The Original Training is the case that we make a training from scratch with genetic algorithms.

The Transfer learning process makes the assumption that the first layers of the ANN keep the general knowledge of vessels mobility while the last layers are related with the particularities of the specific trajectories. A similar phenomenon take place in the visual deep learning in which the first layers understand the general structure of shapes and colors while the last layers make specific object recognition.

In the Transfer Train process we made some trials and concluded to retrain the last two layers. This is a preliminary research and the selection of the last two layers is not a final solution. There is still a large space for further research on how to apply the transfer learning in mobility prediction but sure the experimental results deemed promising. Both full re-training and transfer training of the final layers achieve a similar MSE value to our original training while there is a dramatically decrease in Training time from 3585 to only 33 secs.

6 CONCLUSIONS AND FUTURE WORK

In this research, we demonstrate how we can apply a DL approach with LSTM neurons in order to tackle the problem of next position prediction of a moving entity. We saw that DL can be a powerful model with high accuracy results, if it has a well designed architecture. We used a heuristic approach to conclude to the ANN architecture using a genetic algorithm. In addition, when an ANN is initialized in the beginning of the training process with the parameters of a well trained model on a different dataset then it needs much less training time than a random parameter initialization.

We believe that we are in the beginning of a long-year research in the use of RNN for the needs of trajectory analysis. The topics that have not even been touched in this paper are many. Firstly the retrieval method of the ANN topologies in the context of transfer learning should be investigated further. What is the reason that we find groups of common patterns in the output models should be explained. Is it a side result of the genetic algorithm? Or does it actually express common mobility behaviours? Different hyperparameter optimization techniques should be applied and compare their outcomes.

An approach that gains a growing popularity is the raw mobility tracks to be combined with contextual data in order to propose a semantic next position forecasting RNN model. Afterwards, we should study other types of RNNs such as Gated Recurrent Units that combine the forget and input gates into a single update gate with the advantage that they are speedier to train than LSTM and the Time-Aware LSTM Networks that can handle irregular time intervals.

A recent trend in DL is the Attention mechanisms which enable the ANN to focus on relevant parts of the input sequence more than the irrelevant parts during the forecasting process. It is an open question if and how the attention mechanism can be applied in order to improve the the accuracy of a next position prediction model, but it is possible to have significant improvement in the accuracy outcomes.

Our last idea as future work is a modular DL architecture with many input parameters in which each module is responsible for specific task and in the end the module outputs are consolidated. In this case, we can have DL modules for the speed, course and other input parameters such as the weather conditions or the vicinity of close moving objects.

ACKNOWLEDGMENTS

This work is part of the ACCORDION project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871793".

REFERENCES

[1] Nardini, F.M., Orlando, S., Perego, R., Raffaetà, A., Renso, C., Silvestri, C., 2018. Analysing Trajectories of Mobile Users: From Data Warehouses to Recommender Systems, in: Flesca, S., Greco, S., Masciari, E., Saccà, D. (Eds.), A Comprehensive Guide Through the Italian Database Research Over the Last 25 Years, Studies in Big Data. Springer International Publishing, Cham, pp. 407–421. https://doi.org/10.1007/978-3-319-61893-7_24
[2] Varlamis, I., Tserpes, K., Etemad, M., Soares Júnior, A., Matwin, S., 2019. A network abstraction of multi-vessel trajectory data for detecting anomalies.
[3] Caluwe, R. de, Tré, G.D., Bordogna, G., 2013. Spatio-Temporal Databases: Flexible Querying and Reasoning. Springer Science & Business Media.
[4] Parent, C., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., Damiani, M.L., Gkoulalas-Divanis, A., Macedo, J., Pelekis, N., Theodoridis, Y., Yan, Z., 2013. Semantic Trajectories Modeling and Analysis. ACM Comput. Surv. 45, 42:1–42:32. https://doi.org/10.1145/2501654.2501656
[5] Ferrero, C.A., Petry, L.M., Alvares, L.O., Zalewski, W., Bogorny, V., 2019. Discovering Heterogeneous Subsequences for Trajectory Classification. arXiv:1903.07722 [cs, stat].
[6] Liu, B., Souza, E.N. de, Matwin, S., Sydow, M., 2014. Knowledge-based clustering of ship trajectories using density-based approach, in: 2014 IEEE International Conference on Big Data (Big Data). Presented at the 2014 IEEE International Conference on Big Data (Big Data), pp. 603–608. https://doi.org/10.1109/BigData.2014.7004281

[7] Souza, E.N. de, Boerder, K., Matwin, S., Worm, B., 2016. Improving Fishing Pattern Detection from Satellite AIS Using Data Mining and Machine Learning. PLOS ONE 11, e0158248. https://doi.org/10.1371/journal.pone.0158248

[8] Etemad, M., Soares Júnior, A., Matwin, S., 2018. Predicting Transportation Modes of GPS Trajectories Using Feature Engineering and Noise Removal, in: Bagheri, E., Cheung, J.C.K. (Eds.), Advances in Artificial Intelligence, Lecture Notes in Computer Science. Springer International Publishing, pp. 259–264.

[9] Gambs, S., Killijian, M.-O., del Prado Cortez, M.N., 2012. Next Place Prediction Using Mobility Markov Chains, in: Proceedings of the First Workshop on Measurement, Privacy, and Mobility, MPM ’12. ACM, New York, NY, USA, pp. 3:1–3:6. https://doi.org/10.1145/2181196.2181199

[10] Valsamis, A., Tserpes, K., Zissis, D., Anagnostopoulos, D., Varvarigou, T., 2017. Employing traditional machine learning algorithms for big data streams analysis: The case of object trajectory prediction. Journal of Systems and Software 127, 249–257. https://doi.org/10.1016/j.jss.2016.06.016

[11] Pesaranghader, Ahmad, Pesaranghader, Ali, Matwin, S., Sokolova, M., 2018. One Single Deep Bidirectional LSTM Network for Word Sense Disambiguation of Text Data, in: Bagheri, E., Cheung, J.C.K. (Eds.), Advances in Artificial Intelligence, Lecture Notes in Computer Science. Springer International Publishing, pp. 96–107.

[12] Wu, X., Zurita-Milla, R., Verdiguier, E.I., Kraak, M.-J., 2018. Triclustering Georeferenced Time Series for Analyzing Patterns of Intra-Annual Variability in Temperature. Annals of the American Association of Geographers 108, 71–87. https://doi.org/10.1080/24694452.2017.1325725

[13] Jiang, X., de Souza, E.N., Pesaranghader, A., Hu, B., Silver, D.L., Matwin, S., 2017. TrajectoryNet: An Embedded GPS Trajectory Representation for Point-based Classification Using Recurrent Neural Networks, in: Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering, CASCON ’17. IBM Corp., Riverton, NJ, USA, pp. 192–200.

[14] Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.

[15] Bouras, I., Aisopos, F., Violos, J., Kousiouris, G., Psychas, A., Varvarigou, T., 2019. Mapping of Quality of Service Requirements to Resource Demands for IaaS, in: ResearchGate. Presented at the 9th International Conference on Cloud Computing and Services Science, Crete, Greece. https://doi.org/10.5220/0007676902630270

[16] Bisgaard, S., Kulahci, M., 2011. Time Series Analysis and Forecasting by Example. John Wiley & Sons.

[17] Chollet, F., 2017. Deep Learning with Python, 1st ed. Manning Publications Co., Greenwich, CT, USA.

[18] Violos, J., Pelekis, S., Berdelis, A., Tsanakas, S., Tserpes, K., Varvarigou, T., 2019. Predicting Visitor Distribution for Large Events in Smart Cities, in: 2019 IEEE International Conference on Big Data and Smart Computing (BigComp). Presented at the 2019 IEEE International Conference on Big Data and Smart Computing (BigComp), pp. 1–8. https://doi.org/10.1109/BIGCOMP.2019.8679181

[19] Cerqueira, V., Torgo, L., Mozetic, I., 2019. Evaluating time series forecasting models: An empirical study on performance estimation methods. arXiv:1905.11744 [cs, stat].

[20] Zaremba, W., Sutskever, I., and Vinyals, O., “Recurrent Neural Network Regularization,” arXiv:1409.2329 [cs], Feb. 2015, Accessed: May 28, 2020. [Online]. Available: http://arxiv.org/abs/1409.2329.

[21] Feurer, M., Klein, A., Eggensperger, K. , Springenberg, J., Blum, M. , and Hutter, F., “Efficient and Robust Automated Machine Learning,” in Advances in Neural Information Processing Systems 28, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2962–2970.

[22] Chen, T., and Guestrin, C., “XGBoost: A Scalable Tree Boosting System,” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794, Aug. 2016, doi: 10.1145/2939672.2939785.