International Journal of Geo-Information

MDPI

Article

A Deep Learning Streaming Methodology for Trajectory Classification

Ioannis Kontopoulos [1,*], Antonios Makris [1] and Konstantinos Tserpes [1,2]

[1] Department of Informatics and Telematics, Harokopio University of Athens, 9 Omirou Str., 17778 Athens, Greece; amakris@hua.gr (A.M.); tserpes@hua.gr (K.T.)
[2] Department of Electrical and Computer Engineering, National Technical University of Athens, 9 Heroon Polytechniou Str., 15773 Athens, Greece
* Correspondence: kontopoulos@hua.gr

Abstract: Due to the vast amount of available tracking sensors in recent years, high-frequency and high-volume streams of data are generated every day. The maritime domain is no different as all larger vessels are obliged to be equipped with a vessel tracking system that transmits their location periodically. Consequently, automated methodologies able to extract meaningful information from high-frequency, large volumes of vessel tracking data need to be developed. The automatic identification of vessel mobility patterns from such data in real time is of utmost importance since it can reveal abnormal or illegal vessel activities in due time. Therefore, in this work, we present a novel approach that transforms streaming vessel trajectory patterns into images and employs deep learning algorithms to accurately classify vessel activities in near real time tackling the Big Data challenges of volume and velocity. Two real-world data sets collected from terrestrial, vessel-tracking receivers were used to evaluate the proposed methodology in terms of both classification and streaming execution performance. Experimental results demonstrated that the vessel activity classification performance can reach an accuracy of over 96% while achieving sub-second latencies in streaming execution performance.

Keywords: trajectory classification; deep learning; neural networks; computer vision; distributed processing; stream processing; real-time vessel monitoring; trajectory compression; AIS

Citation: Kontopoulos, I.; Makris, A.; Tserpes, K. A Deep Learning Streaming Methodology for Trajectory Classification. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 250. https://doi.org/10.3390/ijgi10040250

Academic Editor: Christophe Claramunt and Wolfgang Kainz

Received: 10 March 2021
Accepted: 5 April 2021
Published: 8 April 2021

1. Introduction

The sudden increase in mobility data volume and velocity has gained researchers' attention in event-based, distributed, and streaming knowledge extraction methodologies. The automatic identification of patterns in voluminous data is of uttermost importance. Currently, vessels over 300 gross tonnage worldwide are required to carry an Automatic Identification System (AIS) transponder on board, a vessel tracking system that allows vessels to report their position periodically. Although the AIS was initially developed to ensure safety at sea by aiding vessels in collision avoidance situations and assist officers on board, it did not take long for maritime authorities to realize that vessels' mobility data can provide useful information. Mobility patterns formed by vessels through the AIS can reveal behaviors able to explain suspicious or illegal activities at sea, making the early identification of such events a prominent way for Maritime Situational Awareness (MSA).

Several distinct use cases that demonstrate the need for increased MSA include the detection of anomalous vessel behavior due to the damage of the ship, Search And Rescue operations (SAR) and collision detections, and piracy and illegal fishing activities. Regarding piracy, several armed piracy attacks happen every year (https://www.statista.com/topics/1290/pirate-attacks/, accessed on 7 April 2021, https://www.bbc.com/news/business-53426890, accessed on 7 April 2021, https://www.maritime-executive.com/piracy-news, accessed on 7 April 2021), which endanger the lives of passengers

and crew members alike. For that reason, several services are dedicated to immediately reporting such illegal behaviors (https://www.icc-ccs.org/, accessed on 7 April 2021, https://safety4sea.com/tag/piracy-attack/, accessed on 7 April 2021). Regarding the fishing activities, the immediate identification of Illegal, Unreported, and Unregulated (IUU) fishing activities can provide authorities a means of safer maritime surveillance. It is estimated that approximately 640,000 tonnes of ghost gear is left in the world's oceans each year, which ends up entangling and killing birds and other sea animals (https://www.worldanimalprotection.org/illegal-fishing-threatens-wildlife, accessed on 7 April 2021). This led to the creation of services such as Global Fishing Watch (https://globalfishingwatch.org/, accessed on 7 April 2021), which promote ocean sustainability and transparency. To promote MSA, several supervised or unsupervised learning techniques were developed that exploit the kinematic and spatiotemporal characteristics of vessels' trajectories.

Approaches that take advantage of such characteristics include trajectory clustering, classification, anomaly detection, and event prediction. Specifically, trajectory classification is a widely used technique with which normality and behavioral models are created able to identify anomalous patterns or events of interest. On the other hand, trajectory clustering approaches are often employed to form groups of AIS positions with similar spatiotemporal behaviors, uncovering behaviors that are harder to predefine. Although there is an abundance of studies in the literature regarding offline trajectory classification and clustering [1–5], fewer works have focused on steam processing of events in the maritime domain [6–10]. Event processing methodologies are faced with significant challenges when employed on streaming data where the requirements for such applications demand low memory consumption and decreased latencies.

In all those studies, the context of the analysis is typically the physical world and the geography. Latitude and longitude are the basic features in a multi-dimensional space (speed, direction, etc.). However, experts rely heavily on the visualization of trajectories to manually identify parts of the trajectory that are of some importance. This provides the intuition to move the analysis into a different domain, leveraging computer vision techniques in classification. In computer vision, the most commonly used techniques include convolutional neural networks (CNNs) [11–13]. Each layer of a CNN identifies a different feature of the image, including but not limited to shape and color. In order to increase the performance of CNNs, researchers have employed deep learning [12,13], which increases the complexity of the networks in terms of number of hidden layers and nodes. One of the most common goals of such networks is to classify a set of images into a predefined set of labels that are of interest.

Similarly, the main concept of our proposed methodology is to classify in near real time the mobility patterns of vessels to a set of predefined (possibly illegal) activities for the purposes of promoting MSA at sea. The novelty of our approach lies in the usage of computer vision techniques for the classification of trajectories and the detection of activities in mobility data. Specifically, the proposed approach leverages image classification techniques by visually representing vessel trajectories as images. The use of image classification for the problem at hand and its main contributions are:

- Mobility patterns in the maritime domain are visually distinct. This visual distinctiveness allows the increase of trajectory classification performance;
- Image classification allows for the trajectory classification even when data are not transmitted at fixed intervals (e.g., hourly) such as the AIS protocol (https://help.marinetraffic.com/hc/en-us/articles/217631867-How-often-do-the-positions-of-the-vessels-get-updated-on-MarineTraffic-, accessed on 7 April 2021). This is in contrast to time-series methodologies [2] that are inherently unsuitable for such tasks and require data points at fixed time points;
- Trajectory classification approaches found in the literature [4,10,14,15] require a preprocessing step such as the understanding and analysis of data and the selection of features suitable only for the mobility patterns to be classified. This means that

features selected for a certain mobility pattern cannot be applied to other patterns as well [14]. An image classification approach for trajectory classification entirely skips the aforementioned pre-processing step; the same technique for classifying an image (e.g., CNNs) can be applied for the classification of all of the mobility patterns since they are converted into images. Therefore, an image classification approach for trajectory classification yields a promising universal approach for the classification of mobility patterns;

- Approximately 16,000 AIS messages are generated each second from 200,000 vessels worldwide, resulting in 46GB of data per day. In the maritime domain, only in recent years have researchers started tackling the problem of real-time stream processing with the use of AIS messages [6–10]. To the best of our knowledge, this is the first time in the maritime domain literature that computer vision techniques have been used in real time to classify trajectories. Deep learning algorithms such as VGG16 [16], InceptionV3 [17], NASNetLarge[18], and DenseNet201 [19] are used in a distributed and streaming fashion, enabling low response times and early event detection of suspicious activities at sea;
- The clustering of trajectories often constitutes an initial step when dealing with trajectory classification. Most of the well-established clustering algorithms require input parameters that are hard to determine (Optics [20], Traclus [21], DBSCAN [22]) and that eventually have a significant impact on the clustering results. As our method skips this step entirely, as stated above, we eliminated the need for arbitrary or empirical user-defined parameters, making our approach scalable and robust;
- Due to these unparalleled quantities of trajectory data, which in turn can overwhelm human analysis approaches, several compression techniques were applied in order to minimize the size of the trajectory data, while at the same time minimizing the impact on the trajectory analysis methods. Thus, we performed some preliminary experiments in order to demonstrate the effect of trajectory compression on the classification accuracy of mobility patterns.

The rest of the paper is organized as follows. Section 2 serves as a literature review in the fields of deep learning, image classification, trajectory classification, and stream processing. Section 3 describes in detail the proposed methodology of the deep learning stream processing of mobility patterns, while Section 4 evaluates the proposed approach in terms of classification and execution performance. Moreover, Section 5 discusses the experimental results and the benefits of the proposed approach for the problem at hand. Finally, Section 6 summarizes the merits of our work and highlights some perspectives that require further attention in the future.

2. Related Work

To address the problems of anomaly detection and trajectory classification, several data mining techniques have been developed over the years. These two problems are often similar as both of them typically require a pre-trained classification model to evaluate new instances of trajectories. The classification model is trained on trajectories that correspond to instances of normal behavior or instances with a predefined label. The proposed approach offers a method that can perform trajectory classification in real time by employing a computer vision approach. Therefore, our methodology is comparable to other studies that perform trajectory classification or image classification individually since this is the first time in the literature trajectories have been considered as a set of images.

2.1. Trajectory Classification

Trajectory classification is a field of research where analysis is performed on the behavior of moving objects in order to create a classifier able to distinguish different mobility patterns. In the literature, only in recent years, researchers shifted their focus towards the maritime domain. Thus, new research challenges have emerged [23], and more researchers are now engaged in trying to address these. Souza et al. [14] employed three

different classifiers for the detection of three different fishing vessel activities over AIS data, namely trawlers, longliners, and purse seiners, achieving an accuracy of 83%, 84%, and 97% for each activity, respectively. Jiang et al. [15] took advantage of autoencoders to detect fishing activities based on the transmitted AIS messages. The proposed approach divides vessel trajectories into a sequence of discrete window segments. Subsequently, a supervised classifier can predict if a window of AIS data is fishing or not using binary classification. The performance of autoencoders was compared with SVM and random forests resulting in an accuracy of 85%. The authors in [24] used the DBSCAN algorithm for the extraction of Points Of Interest (POI) in fishing vessel trajectories in order to create features that were used for the training of a classification model. The authors in [25], combined General Hidden Markov Models (GHMMs) and Structural Hidden Markov Models (SHMMs) with a Genetic Algorithm (GA) for trajectory classification. They tested their approach on two surveillance data sets, MIT car [26] and T15[27], yielding promising results. Kapadais et al. [2] treated the problem of trajectory classification as a time-series or shapelet classification task. The main problem though with time-series classification is that values or features such as the speed of the vessel need to be reported at fixed time intervals, which is not the case of the AIS protocol – vessels traveling at higher speeds report their position more frequently (the same thing happens when the change of rate of a turn is also higher). Therefore, missing values need to be calculated or existing ones need to be removed, which leads to a potential distortion of the initial data, yielding wrong classification results.

Another approach for trajectory classification is through the use of neural networks. Jiang et al. [28] employed Recurrent Neural Networks (RNNs) for point-based trajectory classification into four different transportation modes. The proposed method uses embedding of GPS data to map the original low-dimensional and heterogeneous feature space into distributed vector representations. The feature space was enriched with segment-based information, and maxout activations were employed by RNNs for an increased performance. The proposed method achieved 98% accuracy, outperforming several well-known classifiers such as decision trees, support vector machines, naive Bayes, and conditional random field. In [29], a deep multi-scale learning model was used to model grid data under different space and time granularities, thus capturing the impact of space and time on the classification results. Furthermore, an attention dense module was designed by combining an attention mechanism, which was able to select major features, and DenseNet, which was able to enhance the propagation of local and spatial features throughout the network. The resulting feature representations were employed as the final classification results. The low-dimensional feature space and the limited feature compositions introduced by the heterogeneous AIS data pose challenges related to the development of deep learning models. The authors in [30] employed deep recurrent neural networks and specifically a novel partition-wise Gated Recurrent Unit (pGRU) architecture for point-based trajectory classification on detecting trawler fishing activities. The proposed method maps low-dimensional features into another space with the use of a partition-wise activation function applied before the linear transformation and receives different parameters for distinct partitions to model them jointly in a nonlinear hierarchical deep structure. Thus, the proposed method is less sensitive to the quality of partitions and achieves better predictive results.

Although many methodologies have been developed for trajectory classification, fewer studies have focused on real-time stream processing of events in the maritime domain [10,31,32]. Lin et al. [31] extracted features from AIS messages, which were then fed to a deep neural network for the prediction of the Estimated Time of Arrival (ETA) of vessels. Chatzikokolakis et al. [32] developed a real-time anomaly detection service, which was focused on identifying a wide range of events of interest in the maritime domain either through the use of machine learning techniques or with rule-based approaches. The authors in [10] presented an online feature extraction process for the classification of fishing trajectories in real time with the use of random forests.

Compared to the aforementioned state-of-the-art methodologies, our approach can not only achieve better classification performance in the identification of mobility patterns, but it can act as a universal multi-class classification and streaming methodology for trajectory classification.

2.2. Image Classification

The most commonly used technique regarding computer vision includes convolutional neural networks [11], where each layer of the network identifies a different aspect of the image. In an attempt to improve the performance of neural networks, deep learning [12,13,33] has been adopted, which increases the complexity of the neural networks by adding more hidden layers and nodes. Deep learning can be used for both classification and feature learning in various fields and is rapidly becoming the state-of-the-art technique, leading to enhanced performance [34]. The Convolutional Neural Network (CNN) is a kind of deep learning technique that has achieved remarkable performance in the field of computer vision [35], as well as in various medical applications [36–38]. According to [39], CNNs can handle four different aspects: training the weights from scratch, fine-tuning, feature extraction, and unsupervised pre-training. Deep convolutional neural networks are among the powerful deep learning architectures and have been widely applied in a broad range of machine learning tasks. The first CNN to create a standard "architectural template" was LeNet-5 [40], which uses two convolutional layers and three fully connected ones. Ever since, the same idea has been applied to more architectures by adding more convolutions and pooling layers, ending with one or more fully connected layers such as AlexNet [41] and VGG-16 [16]. AlexNet was the first CNN architecture that implemented Rectified Linear Units (ReLUs) as an activation function [42]. On the other hand, VGG-16 used 13 convolutional layers and three fully connected ones, keeping the ReLUs from AlexNet as an activation function. In the years that followed, architectures became deeper and more complex by introducing several techniques inside the layers of the networks. Inception-v1 [43] utilizes 22 layers and a "network inside a network" approach by using "Inception" modules. The idea behind this approach was motivated by Arora et al. [44], who suggested an architecture that analyzes the correlation statistics of the last layer and clusters them into groups of high-correlation units. These Inception modules use parallel towers of convolutions with different filters, each filter capturing different features, and then cluster these features together. Inception-v3 [17] and Inception-v4 [45], which constitute the successors of the initial Inception network, added more Inception modules and made some modifications to improve the training speed. In [45], the Inception modules were converted to residual Inception blocks, introducing a new deeper architecture called Inception-ResNet-v2. ResNet-50 [46] along with Inception-v3 were the first networks to use batch normalization and furthermore contained two even deeper architectures, ResNet101 and ResNet152 with 101 and 152 layers, respectively. Xception [47] replaced the Inception modules with depthwise separable convolutions. This means that it performed 1×1 convolutions on every channel and then performed a 3×3 convolution on each output. In DenseNet [19], each layer obtains as the input the feature maps of all preceding layers. Thus, each layer receives a "collective knowledge" from all preceding layers. Finally, NASNet [18] contains two architectures, NASNetMobile and NASNetLarge, which differ in the input size of the training images. The main components of NASNet are two types of convolutional cells: normal cells, which return a feature map of the same dimension, and reduction cells, which return a feature map where the height and width are reduced by a factor of two.

A more recent study of Chen et al. [48] aimed at transportation mode classification of vessel trips from raw trajectory data. The proposed method integrates mobility modes' identification and discovery by utilizing origin and destination point clustering, in order to discover route patterns from historical trajectory data. AIS trajectories are converted into a mobility-based trajectory structure that resembles a low-resolution image. The bounding box of the entire vessel trajectory is converted into a grid that contains multiple pixels. If

the trajectory of the vessel falls within a pixel, then the pixel is colored by using a range of grey colors with each hue representing a different speed value. Furthermore, for mobility modes' classification, a CNN model is utilized by taking advantage of labeled historical trajectory data, resulting in a maximum F1-score of 84.7%. Another CNN architecture for vessel movement classification was employed in [49]. AIS raw trajectories were converted into trajectories images that contained three different vessel movement patterns, static, cruise, and maneuvering. Then, a supervised learning method was used to train the CNN, and the results showed an accuracy of about 75%. Finally, a similar structure was also used in [50] for the classification of the transportation mode of civilians (e.g., walking, bike, bus, taxi), achieving a maximum accuracy of 83.2%. These approaches do not capture the behavior of the trajectory in its entirety, since the surveillance space is segmented into large cells, especially in cases where the moving objects perform micro-movements that eventually form a different mobility pattern. The main difference of our approach and its novelty lie in the fact that high-resolution images can be generated by plotting the trajectory positions, capturing every aspect of the trajectory behavior during a mobility pattern, which can then be used by any image classifier.

3. Methodology

In this section, we present our proposed approach for the classification of vessel activities in near real time. Specifically, we first demonstrate in Section 3.1 the mobility patterns or vessel activities that were used for the experimental evaluation. Despite the fact that four mobility patterns were used for demonstration purposes, our approach can be extended to use more patterns and classify more classes such as piracy if a ground truth is provided. Then, in Section 3.2, the transformation of the trajectories or mobility patterns into images is presented, while in Section 3.3, the way these images are exploited for their classification through deep learning techniques is explained in detail. Furthermore, the classification of images in real-time is presented in Section 3.4. Finally, in Section 3.5, several trajectory compression algorithms are presented, which serve as a means of reducing the size of the AIS data while at the same time having a minimum effect on the classification accuracy.

3.1. Maritime Patterns

In this research study, five different vessel mobility patterns were studied:

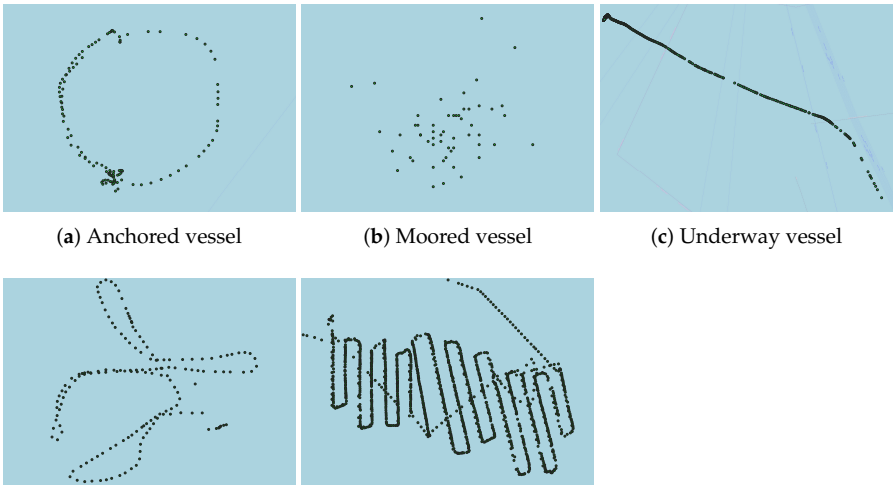
Anchored: Anchoring is a critical operation as it constitutes the key to avoiding accidents related to damage/loss of the vessel or other nearby vessels, by preventing drifting away from a desired position. Anchoring can be related to loading/unloading cargo, maintenance, waiting for a berth, and refueling, as mentioned in [51]. When anchoring, factors such as sea and wind conditions (direction and strength), sea currents, prohibited areas, underwater pipelines, shallow waters, and other vessels in the nearby area must be taken into account. During this type of activity, vessels are anchored offshore in an anchorage area. The vessel tends to move around the anchor and forms circular or semi-circular patterns in different orientations, as shown in Figure 1a, in which the anchor is located approximately in the middle of the circle. The circular movement of the vessel around the anchor can be caused by the effects of the wind, the tide, or sea currents, and as a result, the speed over ground is minimized to approximately 0.5–3.0 knots depending on the vessel type.

Moored: During this type of activity, vessels are anchored inside a port. Mooring can be related to lassoing, tethering, tying, or any permanent structure to which a vessel may be secured such as quays, wharfs, jetties, piers, anchor buoys, and mooring buoys. Vessel's motion is more limited compared to an anchored vessel because the latter is constrained not only by the anchor, but also by the mooring buoys. As shown in Figure 1b, vessel position appears scattered and "close" to the mooring. The slight movement is due to the effects of the wind and the current of the sea, and as a result, the speed over ground does not exceed 1 or 2 knots.

Underway: During this type of activity, a vessel is traveling from a departure point to a destination point. A vessel is considered underway when it is not aground, anchored, or has not been made fast to a dock, the shore, or some other stationary object. An underway vessel is not necessarily propelled or pushed by an instrument or a device, but it may be underway because of the wind or the sea current [52,53]. Figure 1c shows the trajectory of an underway vessel. The movement patterns of an underway vessel typically form straight, curved, or zigzag lines, which can occur when a vessel avoids islands in the middle of the sea.

Trawling: There are different kinds of fishing activities such as trawling and longlining. Trawling vessels typically keep their speed steady at approximately 2.5 knots in order to stabilize the fishing net, which is dragged by the boat. Moreover, trawling vessels do not travel in a straight line, but they tend to frequently change their course around the fishing area of interest (Figure 1d). The trawling activity can last from several hours to several days.

Longlining: On the other hand, vessels engaged in longlining activity set fishing lines with baited hooks attached to them. While setting the lines, the vessels travel at their steaming speed, and they maintain a constant speed. When all lines are set, they are left in the water, and the vessels drift slowly with them. While drifting, the speed of the vessel can be approximately 2.5 knots due to the wind or the current of the water. Finally, the longlining activity has a similar duration to the trawling activity and can last several days. Although the two fishing activities have some similarities such as frequent turns and similar speeds, their mobility pattern can differ visually (Figure 1e). It can be observed that long straight-line sub-trajectories are formed, which correspond to the part of the trajectory during which longlining vessels set the lines with the baited hooks.



(**a**) Anchored vessel (**b**) Moored vessel (**c**) Underway vessel

(**d**) Vessel trawling (**e**) Vessel longlining

Figure 1. The different vessels' mobility patterns.

3.2. Image Representation

This section describes the image representation approach of the trajectories. These trajectories are the movement patterns indicative of the vessel activities as defined in the previous sub-section. In order to visualize and efficiently classify the movement patterns of the vessels, two key features are captured that characterize the trajectory patterns in the maritime domain: (i) the shape of the trajectory, which indicates the way the vessel moves in space taking into account the rate of change of the Course Over Ground (COG) or heading, and (ii) the speed, which indicates how fast the vessel moves in space.

3.2.1. Shape of the Trajectory

Trajectories of the same vessel activity form similar patterns. However, as the distance each vessel travels through space is different (e.g., a fishing vessel in the Atlantic Ocean travels greater distances compared to a fishing vessel in the Irish Sea), the bounding box or the surveillance area in which the vessel moves needs to be normalized. Therefore, to efficiently capture and place the shape of the trajectory inside a normalized bounding box, the total distance of both the x and the y axis in which the vessel moves must be defined first. For this reason, we calculated the total horizontal distance (x—Equation (1)) and the total vertical distance (y—Equation (2)) the vessel travels based on the minimum and maximum longitudes and latitudes, respectively. The total horizontal distance is defined as:

$$distance_x = longitude_{max} - longitude_{min} \quad (1)$$

Similarly, the total vertical distance the vessel has traveled is defined as:

$$distance_y = latitude_{max} - latitude_{min} \quad (2)$$

Then, the distance each AIS position m has traveled from the minimum longitude and latitude can be calculated from Equations (3) and (4), respectively, as follows:

$$distance(m_x) = longitude_m - longitude_{min} \quad (3)$$

and:

$$distance(m_y) = latitude_m - latitude_{min} \quad (4)$$

From Equations (1)–(4), we can calculate the percentage of the total distance each AIS position m has traveled so far from the minimum coordinate in both the x and y axes:

$$normalized(m_x) = distance(m_x) \div distance_x \quad (5)$$

and:

$$normalized(m_y) = distance(m_y) \div distance_y \quad (6)$$

Given a predefined image size of $N \times N$, the exact position of m inside an image can be calculated as follows:

$$pixel_x = normalized(m_x) \times N \quad (7)$$

and:

$$pixel_y = normalized(m_y) \times N \quad (8)$$

Therefore, each AIS position is placed inside a normalized bounding box or a surveillance space of size $N \times N$, which is essentially an image representation. Figure 2 demonstrates an example of the surveillance space normalization for $N = 10$. Each green circle corresponds to an AIS position where the corresponding longitude and latitude are denoted by the blue arrows. The total horizontal distance is $distance_x = 28.75 - 2.875 = 25.875$, while the total vertical distance is $distance_y = 92.5 - 9.25 = 83.25$. Based on Equations (3) and (4), $distance(m_x)$ of the upper right AIS message is equal to 20.125, and $distance(m_y)$ is equal to 0. According to Equations (5) and (6), $normalized(m_x)$ is equal to 0.7, and $normalized(m_y)$ is equal to 0. Multiplying each normalized value by $N = 10$ results in $pixel_x = 7$ and $pixel_y = 0$. In order to fit boundary AIS positions into the normalized surveillance space, pixel values smaller than N are transformed into 1, and pixel values larger than N are transformed into N Therefore, $pixel_y$ is transformed into 1, and the final pixel position inside the 10×10 image is $x = 7$ and $y = 1$, as shown in Figure 2.

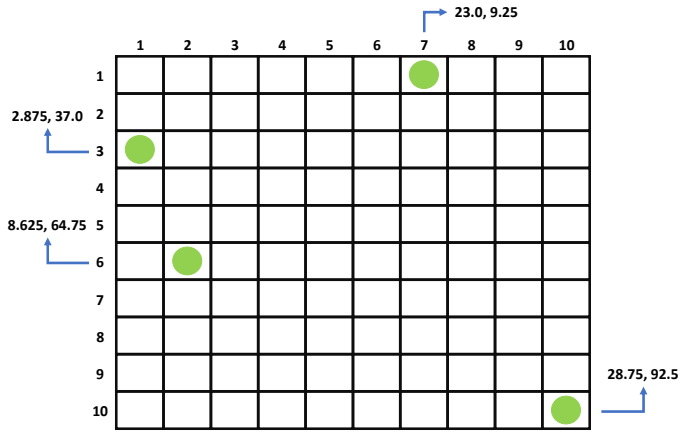


Figure 2. Example of space normalization.

Finally, a straight line between each temporally consecutive $pixel(pixel_x, pixel_y)$ or AIS position m is drawn using Bresenham's line algorithm [54], in order to make the pattern created by each trajectory more distinctive. Bresenham's line algorithm is a line drawing algorithm that generates points that form a close approximation to a straight line between two points of an N-dimensional raster.

3.2.2. Speed

The range of speed regarding the most common vessel types such as passenger, cargo, or fishing varies between 0 and 22 knots, $R = [0, 22]$. In order to represent the speed values of each AIS position m, the range R was segmented into 2-knot increments with each increment corresponding to a different RGB color value in the final image. The 2-knot increment was chosen because we wanted a reasonable amount of color values (11 distinct color values against 22 color values with 1-knot increments) while maintaining a relatively high number of increments. Therefore, the color values were different for dissimilar speed values of AIS positions. Furthermore, the speed of the fishing vessels varies between 2 and 4 knots [10,14]. which corresponds to a 2-knot increment. Sometimes, speeds exceed the R range; thus, speed values greater than 22 knots have the same color as the last increment ($20 \leq s < 22$). Moreover, pixels that do not contain an AIS position or a line drawn by Bresenham's line algorithm are colored white, and pixels that contain the lines between the AIS positions use the color of the first increment. As a result, there are 12 color values in a trajectory image. An example of a trawling trajectory is shown in Figure 3. Finally, it is worth noting that the transformation of a trajectory into an image adds a negligible overhead.

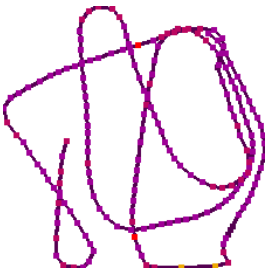


Figure 3. Example of a trawling trajectory that has been transformed into an image.

3.3. Deep Learning for Vessel Pattern Classification

This section describes the deep learning approach for vessel mobility pattern classification from trajectory images, based on Convolutional Neural Networks (CNNs).

CNNs do not require hand-created feature extraction, and deep features of trajectories such as shape and color (e.g., speed) can be learned by the CNN automatically. As stated in [48], neurons in the CNN receive signals from other neurons in the local area

in the preceding layer; thus, the CNN is able to capture more local spatial correlations. Furthermore, the weight-sharing characteristic in the connection of adjacent layers can significantly reduce the number of variables. The main disadvantage of deep learning approaches regarding image classification is that they require a large amount of data in order to perform accurate feature extraction and classification. In order to overcome this limitation, transfer learning was adopted. Transfer learning is a common and effective method, which aims at training a network with fewer samples, as the knowledge extracted by a pre-trained model is then reused and applied to the given task of interest. The intuition behind transfer learning is that generic features learned on a general large data set can be shared among seemingly disparate data sets. The learned features can be used to solve a different, but related task [55].

In general, there are two ways of transfer learning utilization in the context of deep learning [56]: (a) feature extraction [57] and (b) fine-tuning [58,59]. In feature extraction, representations learned from a pre-trained model are treated as an arbitrary feature extractor and employed in order to extract meaningful features from new samples. As the base convolutional network already contains generically useful features for classification, there is no need for retraining the entire model. In fine-tuning, the fully connected layers of the pre-trained model are replaced with a new set of fully connected layers. These new layers are trained on a given data set, and the weights of the top layers of the pre-trained model along with the newly-added layers are "fine-tuned" by means of backpropagation. Thus, the weights are tuned from generic feature maps to features associated specifically with the provided data set. With fine-tuning, specialized features are adapted to a given task. Fine-tuned learning experiments have been presented to be much faster and more accurate in comparison with models trained from scratch [60].

In this research work, fine-tuning was employed as the standard method for transfer learning. The weights were pre-trained on the ImageNet [61] data set for all the deep CNNs. Figure 4 illustrates the fine-tuning process on the VGG16 network. The same process was applied to all the examined deep learning models. The VGG16 model contained 13 convolutional ($CONV$) and 3 fully connected (FC) layers. The final set of layers that contained the FC layers along with the $softmax$ activation function is called the "head". The network was instantiated with weights pre-trained on ImageNet, as shown at top of the figure. Afterwards, the FC layers were truncated, and the final $POOL$ layer was treated as a feature extractor, as depicted in the middle of the figure. Finally, the truncated layers were replaced by a new FC head layer, which was randomly initialized and placed on top of the original architecture (bottom of the figure). Then, the model was trained through a backpropagation process. The body of the network, i.e., the weights of the $CONV$ layers of the pre-trained network, were frozen such that only the FC head layer was trained. This was because the $CONV$ layers had already learned discriminative filters and captured universal features like curves and edges; thus, these weights had to remain intact. On the other hand, the FC head layer was randomly initialized from scratch and focused on learning data set-specific features; thus, random values were able to destroy the learned features. Early-layer features appeared more generic, whereas later features progressively became more specific to a task [62].

Historical trajectory data were transformed into images and labeled based on concrete mobility patterns' annotation into four classes ($anchored$, $moored$, $underway$, $fishing$). These images were fed as the input into the different deep learning models for training. The labels were encoded into one-hot vectors, and then, each model was trained through a backpropagation operation until the optimization process of the objective function converged. All the examined CNNs shared some common hyperparameters. Input images were scaled to the fixed size of 224×224 pixels. Training was conducted for 25 epochs for all pre-trained models with a learning rate of (1×10^{-3}) and a batch size of 8. The output

of the convolution layers was activated by the non-linear activation function called the Rectified Linear Unit (ReLU), which computes the function:

$$ReLU(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{9}$$

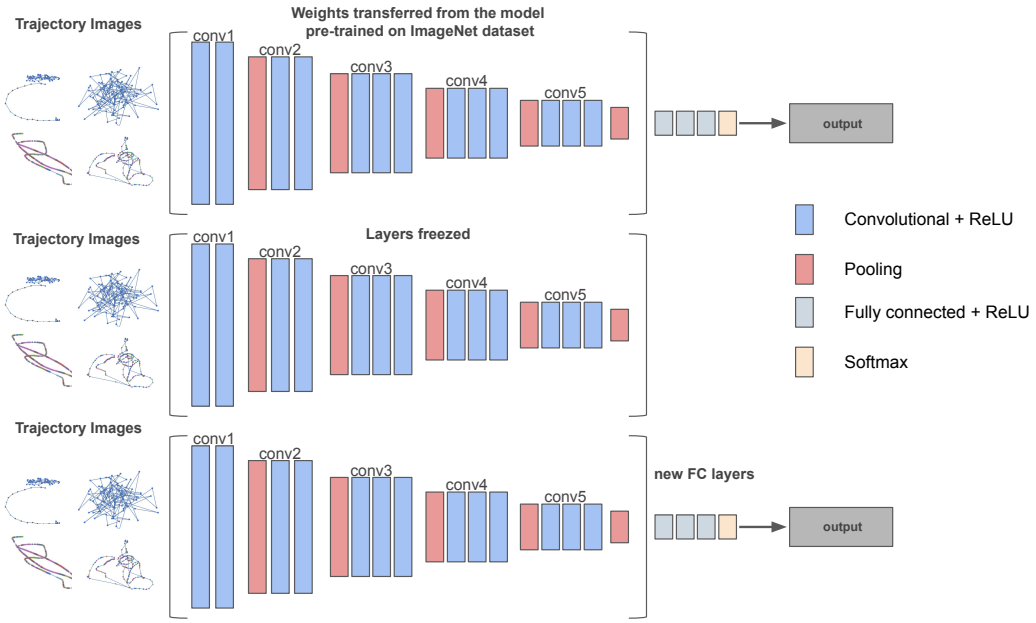


Figure 4. Fine-tuning on the VGG16 network architecture.

ReLU was chosen as the activation function due its reduced likelihood of vanishing gradients and its efficient computation. After each convolution layer, the pooling layer was introduced to carry out downsampling operations, which reduced the in-plane dimensionality of the feature maps. Downsample operations after convolutional layers introduce a translation invariance to small shifts and distortions and decrease the number of subsequent learnable parameters. Average pooling [63] was employed as the pooling strategy, which performed an extreme type of dimensionality reduction, where a tensor with dimensions $h \times w \times d$ was downsampled into a $1 \times 1 \times d$ array by simply taking the average of all the elements in each $h \times w$ feature map, whereas the depth of feature maps was retained. The number of learnable parameters was reduced, preventing overfitting. The output feature maps of the final layer were flattened and connected to the fully connected layers, in which every input was connected to every output by a learnable weight. The output layer generated a probability distribution over the classification labels by resorting the softmax function:

$$softmax(x)_i = \frac{exp(x_i)}{\sum_{j=1}^{I} exp(x_j))} \tag{10}$$

where x is the output vector of the network. The softmax function calculates the probabilities of each class $i = 1, 2, \ldots, I$ over all possible target classes.

The error was calculated between the actual output per class t_i and the estimated output derived from $f(s)_i$ using the "categorical_crossentropy" as the loss function:

$$CE = -\sum_{i}^{C} t_i \log(f(s)_i) \tag{11}$$

where $f(s)$ refers to Equation (10). "categorical_crossentropy" compares the distribution of the predictions with the true distribution. The true class was represented as a one-hot encoded vector, and the closer the model's outputs were to that vector, the lower the loss was.

Examined CNNs were compiled utilizing the optimization method called Adam [64], which finds the minimum of the objective (error) function making use of its gradient. Furthermore, a dropout layer [65] of 0.5 was applied, which means that 50% of neurons were randomly set to zero during each training epoch, thus avoiding overfitting on the training data set. Dropout is one of the most popular regularization techniques, which forces the weights in the network to receive only small values. making the distribution of weight values more regular. As a result, this technique can reduce overfitting on small training examples [66]. Another efficient way to prevent model overfitting is data augmentation, which increases the amount of training data [67]. Thus, data augmentation was performed during training, leveraging several multi-processing techniques. Specifically, the transformations employed included random rotation of the images (the maximum rotation angle was 30 degrees), horizontal flips, shearing, zooming, cropping, and small random noise perturbations. Data augmentation improved the generalization and enhanced the learning capability of the model. Table 1 summarizes the parameters employed by the different deep CNNs.

Table 1. Configuration of the CNN architectures.

Parameter	Value
Epochs	25
Learning rate	1×10^{-3}
Batch size	8
Activation function	*ReLU*
Pooling	*Average2D*
Optimizer	*Adam*
Output activation function	*Softmax*
Loss function	*categorical_crossentropy*
Dropout probability	0.5

3.4. Streaming Vessel Classification

Low latency and high throughput are two key characteristics of streaming systems that support fast decision-making. In such systems, events must be processed in real time, e.g., after the consumption of an event, the system must output a result as soon as possible. In a real-world scenario, approximately 200,000 vessels globally transmit more than 16,000 AIS messages per second, totaling 46 GB a day, with each AIS receiver being flooded with 5 to 8 AIS messages per second. Therefore, a streaming system that balances latency and throughput needs to be developed for the classification of vessel activities in near real time. To this end, we proposed a deep learning streaming methodology for the mobility patterns' identification, which consisted of two phases, the offline model training and the real-time vessel activity classification.

Offline model training: In this phase, the deep learning model was created as described in Section 3.3. For the training of the model, representative trajectories of the mobility patterns were required to be used as the ground truth. Thus, already labeled trajectories from historical AIS data, which were annotated as "anchored", "moored", "underway", and "fishing", were used. These trajectories were converted into images, which in turn were used as training instances of the deep learning model. For the implementation, the Keras (https://keras.io/, accessed on 7 April 2021) library with a TensorFlow (https://www.tensorflow.org/, accessed on 7 April 2021) backend was used, which consisted of not only APIs to create neural networks, but pre-trained CNN models as well. These pre-trained models were employed and fine-tuned to classify images of mobility patterns in the next phase.

Real-time vessel activity classification: There are several frameworks for distributed stream processing such as Apache Spark (https://spark.apache.org/streaming/, accessed on 7 April 2021), Apache Flink (https://flink.apache.org/, accessed on 7 April 2021), and Kafka streams (https://kafka.apache.org/documentation/streams/, accessed on 7 April 2021), out of which only Apache Spark has support for the Python programming language, which was needed for the implementation of the neural networks and the creation of the images. Apache Spark is not preferred since it performs micro-batching over streams of events, and a system is needed that can handle event-processing in real time. Therefore, to balance event-processing with low latency and high throughput, the Apache Kafka (https://kafka.apache.org/, accessed on 7 April 2021) framework was used in this phase, a distributed publish-subscribe and message-exchange platform similar to a message queue able to process streams of events as they occur. Three major concepts exist in the Apache Kafka ecosystem, namely topics, producers, and consumers. A kafka topic is a category/feed name to which messages are stored and published. A producer is an application that continuously publishes or stores messages in a topic. A consumer is an application that is subscribed to a topic and continuously reads or consumes messages. A kafka topic can be divided into n partitions with each partition storing different messages. Specifically, messages with the same key will be stored in the same partition. n consumers can be subscribed to the partitioned topic with each consumer consuming from a different partition, thus enabling high throughput. A producer can store messages to the partitioned topic, and Apache Kafka will handle the load balancing of the messages among the partitions internally. In our use case, the vessel identifier can be considered as the message key, the AIS receiver as the producer and the prediction modules as the consumers. An even distribution of the load within the nodes of the system reduced the probability that a node would turn into a hotspot, and this property also acted as a safeguard to the system reliability [68,69].

The prediction modules were the main components of our methodology. Each prediction module was responsible for consuming AIS messages from a set of vessels and classifying parts of their trajectories based on the deep learning model created in the previous phase. To classify parts of the vessels' trajectories, the module used a temporal sliding window W of user-defined length L and step S. Every S AIS messages, the module took into account all of the AIS messages of the corresponding vessel that belonged to the current window W and converted them to an image. Next, the deep learning model read the image and output for each of the predefined vessel activities a probability. The vessel activity with the highest probability was the final prediction of the module. Figure 5 illustrates the sliding windows of the prediction modules. A window of length L and a step of $S = 2$ events is illustrated, which slides from left to right (W_1 to W_3). The upper temporal limit of the window is the time t the incoming message was consumed, and the lower temporal limit of the window is L hours/minutes/seconds before t, $t - L$. Messages that fall within the window W were used for the prediction.

Finally, Figure 6 visualizes the entire system architecture of our proposed methodology. Initially, historical AIS messages that were collected from AIS receivers were transformed into images. These images were then used to train a deep learning model offline. For the real-time classification, messages received through the AIS receivers were stored in a Kafka topic. These messages were then consumed by the prediction modules and transformed into images, which were fed to the already trained deep learning model for the prediction of the vessel activity. During the real-time vessel activity classification, more than one prediction module can be employed to increase the throughput of the system.

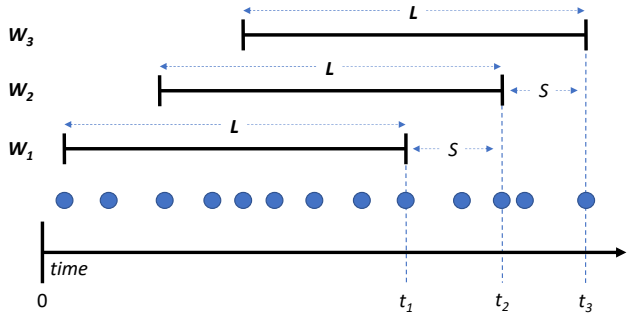


Figure 5. Example of sliding windows with a sliding step of $S = 2$ events.

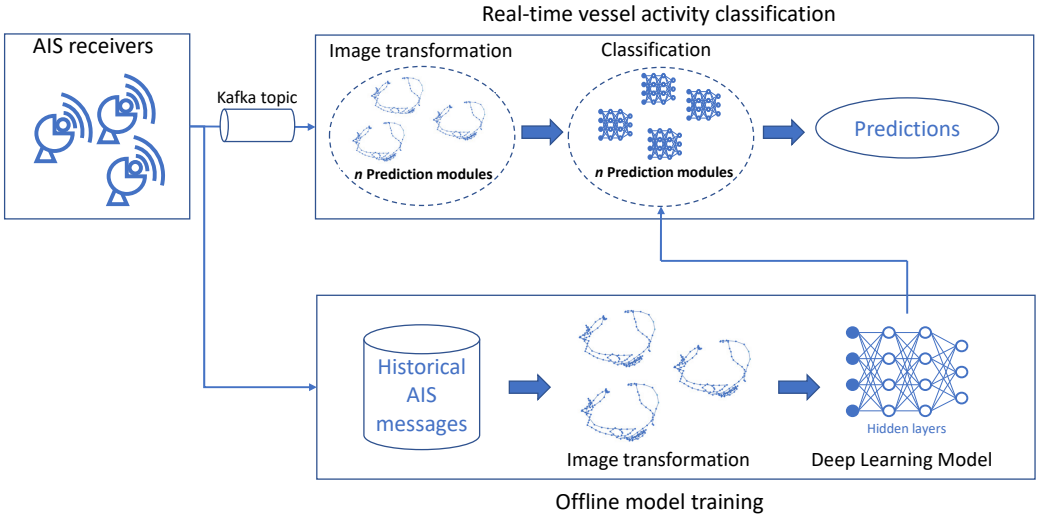


Figure 6. System architecture.

3.5. Trajectory Compression

The amount of spatiotemporal data being constantly generated by modern-day systems such as GPS devices, AIS receiving stations, location-based services, and satellites has seen staggering growth. Managing and analyzing these data are becoming increasingly important, enabling applications that may transform science and society [70,71]. Spatiotemporal database management systems constitute the basic components in dealing with the challenges posed by spatiotemporal applications [72]. While more points are collected, the more accurate a trajectory representation becomes, the enormous volumes of trajectory data can quickly overwhelm available data storage systems, and the redundant information often contained in these data can overwhelm human analysis.

A typical approach to the above challenges is the reduction of trajectory data by employing compression techniques. Trajectory compression aims at substantial reductions in the amount of data while minimizing the loss of information (error) and at the same time preserving the quality of the trajectory. Thus, it is becomes clear that there is a trade-off between the compression rate and the quality of trajectory achieved after compression. The quality is strongly related to the information loss and measured though metrics that calculate the applied error over a trajectory if a certain point is discarded. The motivation behind trajectory compression in terms of data points representing a trajectory is that only a small portion of carefully selected points is needed in order to conduct the analysis task without significant loss in its performance metrics.

Several studies exist in the literature regarding compression that aim to balance the trade-off between the achieved compression rate and the acceptable error. Meratnia et al. [73] was among the first research studies considering three-dimensional mobility data, thus trajectories of mobility objects, where the temporal factor was taken into consideration in the compression techniques. Leichsenring et al. [74] presented an evaluation of seven

lossy compression algorithms with a view toward identifying the most important aspects in selecting the appropriate compression algorithm, while Muckell et al. [75] also presented a performance comparison of seven compression algorithms when utilizing two different errors, the Synchronized Euclidean Distance (SED) and the median difference in speed. Potamias et al. [76] proposed two online compression techniques called STTraceand Thresholds. STTrace exploits spatiotemporal features that characterize movement and detects changes in speed and orientation, thus minimizing the SED in each step, while in Thresholds, the choice of appending a point to the sample is based on a velocity threshold. In SQUISH [77], the most important points of a trajectory are prioritized, and new incoming points require the removal of another point that is already stored in a buffer. The removed point is that with the minimum estimated SED error. Dead reckoning [78] estimates the successor point through the object's current position and velocity. Finally, an alternative compression technique was presented in [79], which was based on the dual transformation of moving objects. Each raw trajectory is approximated by a discrete number of linear sub-trajectories, which are subjected to dual transformation. The duality transformation of line segments is based on Hough-X and Hough-Y. In essence, Hough-X constitutes a linear transformation on spatial coordinates. As the results suggested, the Hough space represents better mobility patterns than the original trajectory spatial data, thus leading to more homogeneous clusters, and provides storage efficiency as well.

Our interest is focused on offline lossy compression algorithms, which remove the less significant data in an attempt to preserve the major characteristics of the original trajectory while maintaining an acceptable degree of error. In this research work, five lossy offline top-down compression algorithms were evaluated. In top-down algorithms, a trajectory is recursively split until a halting condition is met [80]. The algorithms were as follows:

- Douglas–Peucker (DP)
- Time Ratio (TR)
- Speed based (SP)
- Heading based (HD)
- Speed-Heading based (SP_HD)

Douglas–Peucker [81] is a line generalization algorithm that recursively selects points from the original set. The process of the algorithm is illustrated in Figure 7. Initially, the first (P_a) and the last point (P_b) are selected as the anchor and float point, respectively, and these two points form a line segment $LS\ a \rightarrow b$, as shown in Figure 7a. The starting curve is a set of points and a threshold (distance dimension) $\varepsilon > 0$. Subsequently, the point with the maximum perpendicular distance ($\perp$) from $LS\ a \rightarrow b$ is selected (P_1). If this point is closer than ε to the $LS\ a \rightarrow b$, all the points between P_a and P_b are discarded. Otherwise, the newly added point is included in the resulting set and becomes the new float point for the first segment and the anchor point for the second segment, as shown in Figure 7b. This process is repeated using recursion on each line segment (Figure 7c). When the recursion is completed, a new simplified trajectory is generated of only those points that have been marked as kept (Figure 7d).

Despite the popularity of line generalization algorithms in the fields of cartography and computer graphics, the main drawback when dealing with trajectory data of moving objects is that they do not consider the temporal aspect. The spatial error finds the closest point on the compressed representation of a trajectory to each point on the original trajectory. Each trajectory is treated as a line in two-dimensional space. However, a trajectory incorporates an important extra dimension: time. Meratnia and By [80] proposed the time ratio algorithm, which computes the distances between pairs of estimated temporally synchronized positions, one on the original and one on the corresponding approximated trajectory, as illustrated in Figure 8.

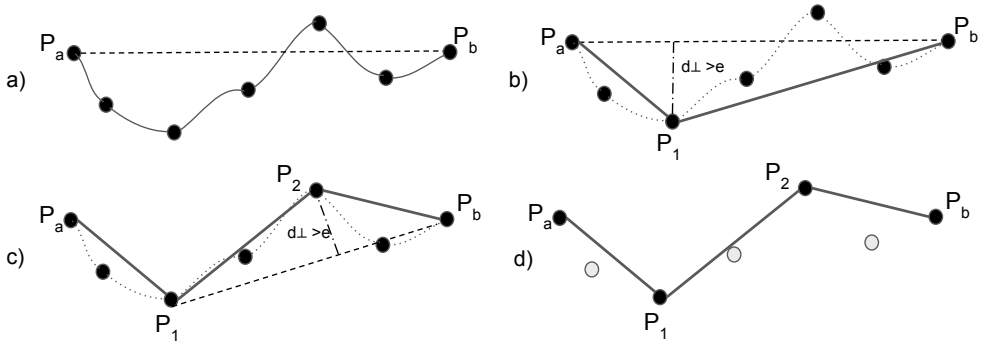


Figure 7. Douglas–Peucker algorithm. (**a**) Line segment formed by anchor and float points. (**b**) The point (P_1) with the maximum perpendicular distance from line segment is selected. (**c**) The process is repeated using recursion on each line segment. (**d**) New simplified trajectory.

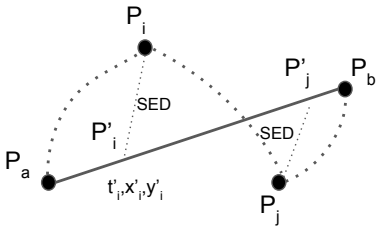


Figure 8. Time ratio algorithm.

For each point on the original trajectory such as P_i, the temporally synchronized point $P_{i'}$ is located on the approximated trajectory Tr_{Pa-Pb}, and the coordinates (x'_i,y'_i) of Pi' can be calculated using linear interpolation as:

$$\begin{aligned} x'_i &= x_a + \frac{t_i - t_a}{t_b - t_a}(x_b - x_a) \\ y'_i &= y_a + \frac{t_i - t_a}{t_b - t_a}(y_b - y_a) \end{aligned} \tag{12}$$

After the temporally synchronized points are determined, the synchronized Euclidean distance is calculated as $SED(P_i, P'_i) = \sqrt{(x_i - x'_i)^2 + (y_i - y'_i)^2}$. (x_i, y_i) and (x'_i, y'_i) represent the coordinates of a moving object at time t_i and synchronized time t'_i in the uncompressed and compressed traces, respectively. Finally, if the distance between $P_{i'}$ and P_i is greater than a user-defined threshold, the particular point is included in the resulting set, otherwise discarded. By including the temporal factor, the algorithm is able to provide more accurate results.

The speed-based algorithm exploits the speeds from subsequent segments of a trajectory. If the absolute value of the speed difference between two subsequent segments of a trajectory, for example $|v_{i+1} - v_i|$, is greater than a user-defined threshold, the point in the middle is retained, otherwise discarded. The algorithm is illustrated in Figure 9.

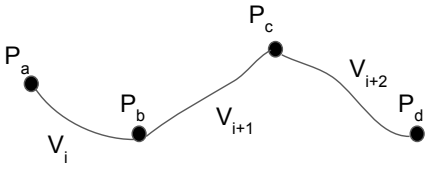


Figure 9. Speed-based algorithm.

The heading-based algorithm exploits the angle formed by subsequent segments of a trajectory. Initially, the distances between continuous points are calculated. Then, for each triangle formed by three continuous points like $P_aP_bP_c$, the distance length of the opposite

side of the examined angle is calculated, e.g., P_aP_c, and the law of cosines is used in order to find the angle, e.g., A_1 as $c^2 = a^2 + b^2 - 2ab\cos\gamma$, where γ denotes the angle contained between the sides of length a (P_aP_b) and b (P_bP_c) and opposite the side of length c (P_aP_c). If the angle of two subsequent segments is greater than a user-defined threshold, the point in the middle is retained, otherwise discarded. The algorithm is illustrated in Figure 10.

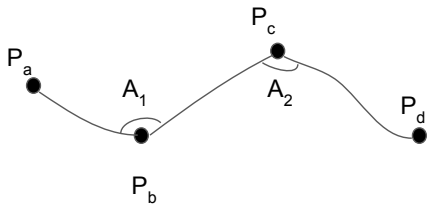


Figure 10. Heading-based algorithm.

By combining the speed-based with the heading-based algorithm, a new algorithm can be derived that exploits the concept of both algorithms simultaneously. The algorithm, namely speed-heading based, retains or discards a point based on two thresholds. If the speed or heading are greater than a speed and heading threshold, respectively, the point is retained, otherwise discarded.

One of the challenges is to define the thresholds to be employed by the compression algorithms. Setting the proper threshold is an application-dependent process and can significantly affect the compression results in terms of the compression ratio and achieved quality. In general, the steps that each algorithm follows are: (i) Group the points, and create a trajectory of each object based on an identifier. This practically means that the number of trajectories in a data set is as large as the number of objects. (ii) Compress the whole trajectory of each identifier. (iii) Write the points remaining after compression to a file grouped by identifiers. As we grouped the points for each identifier (object) in the data set, we extracted the trajectories (one trajectory for each object). In order to determine the threshold that each algorithm will use, a dynamic process was proposed: for each individual trajectory, a different threshold was automatically defined based on an average. Specifically, we calculated the average epsilon (*DP*), average SED (*TR*), average speed (*SP*), average heading (*HD*), and average speed_heading (*SP_HD*) for each trajectory before any of the respective algorithms were applied. Then, we used this Average value (AVG) as a reference point to define a common rule for the threshold calculation. This practically means that in every trajectory of each data set, a different threshold was applied in the corresponding algorithm, which depended on the actual features and peculiarities of this trajectory. Thus, we eliminated the need for arbitrary user-defined thresholds.

4. Experimental Evaluation

This section presents the experimental evaluation of the proposed methodology. To properly evaluate the approach presented in Section 3, we present experiments that:

- demonstrate the achieved classification accuracy and the overall classification performance of the deep learning methodology (Section 4.2),
- demonstrate the achieved latency and throughput and the overall execution performance (Section 4.3),
- demonstrate the effect the trajectory compression algorithms have on the classification performance of the task at hand (Section 4.4).

The real-world data sets used to evaluate the proposed trajectory classification approach are presented in the following (Section 4.1).

4.1. Data Set Description

The first data set used contained AIS messages collected from a Terrestrial AIS receiver (T-AIS) that covers the Saronic Gulf (Greece) including the port of Piraeus and contained high-quality AIS information without gaps of information. The data set provided infor-

mation for 1229 unique vessels and contained 11,769,237 AIS records in total. The vessels were monitored for almost a one and a half month period starting 18 February 2020 and ending 31 March 2020. A small sample of the data set can be found here [82].

AIS, besides the positional information, also transmits its navigational status (https://help.marinetraffic.com/hc/en-us/articles/205426887-What-kind-of-information-is-AIS-transmitted-2020-04-06, accessed on 7 April 2021). Navigational status is manually inserted by the vessel's crew and constitutes an identifier regarding the vessel's activity (e.g., five indicates that the vessel was moored when the message was received). The AIS messages used for our ground truth data set contained activities that were extracted from vessels that set their navigational status to 0, 1, and 5—hereafter, Pattern A:

- 0: underway
- 1: anchored
- 5: moored

Since the status is reported manually, it can often be observed that the vessel's crew may forget to change the status, resulting in false annotations. To tackle the problem of false annotations and to provide good-quality representations of vessel activities to the classifiers, one-hundred fifty representative images from each class (450 in total) were selected.

The second data set that was used was provided by MarineTraffic (https://www.marinetraffic.com, accessed on 7 April 2021) and contained AIS messages from 1 January 2018 to 28 February 2018 in the seas of Northern Europe. The total number of AIS messages of this data set summed up to 61,050. Similar to the first data set, the AIS messages used for our ground truth data set contained fishing activities that were extracted from fishing vessels that set their navigational status to seven, which indicates "fishing activity", and by vessels that set their destination to trawling or longlining for the corresponding activity. These fishing trajectories were then segmented into:

- fishing (trawling and longlining)
- moored
- underway

Vessels are typically moored to a port, then they travel to the fishing area, and finally, they are engaged in a fishing activity (trawling or longlining)—hereafter Pattern B. Again, one-hundred fifty representative images from each class (450 in total) were used.

In both data sets, each AIS record consisted of 10 attributes, as described in Table 2.

Table 2. Data set attributes.

Feature	Description
ship_id	A unique identifier for each ship
lat, lon	The longitude and the latitude of the current ship position
ship_type	AIS-reported ship type
speed	Speed over ground in knots
course	Course over ground in degrees with 0 corresponding to north
heading	Ship's heading in degrees with 0 corresponding to north
destination	AIS-reported destination
navigational_status	The identifier regarding the vessel's activity
timestamp	The time at which the message was received (UTC)

4.2. Deep Learning Evaluation

In order to evaluate the classification performance, several commonly used classification metrics were adopted: Accuracy (ACC), precision, recall (sensitivity), and F1-score. *Accuracy* indicates how well a classification algorithm can discriminate the classes of the

trajectories in the test set. As shown in Equation (13), ACC can be defined as the proportion of the predicted correct labels (TP + TN) to the total number of labels (N).

$$Accuracy(ACC) = \frac{|Correctly\ Labeled\ Examples|}{N} \tag{13}$$

*Precision*is the proportion of predicted correct labels to the total number of actual labels, as shown in Equation (14), while *Recall*is the proportion of predicted correct labels to the total number of predicted labels, as shown in Equation (15). Recall is also known as sensitivity or the True Positive Rate (TPR).

$$Precision(P) = \frac{TP}{TP + FP} \tag{14}$$

$$Recall(Sensitivity) = \frac{TP}{TP + FN} \tag{15}$$

The $F1$-score consists of the harmonic mean of precision and recall, where precision is the average precision in each class and recall is the average recall in each class, as illustrated in Equation (16).

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{16}$$

TP, TN, FP, and FN refer to the True Positive, True Negative, False Positive, and False Negative samples for each class (anchored, moored, underway, and fishing).

The classification performance of several pre-trained CNNs, namely VGG16, InceptionV3, NASNetLarge, and DenseNet201, was evaluated on the effectiveness of vessel pattern recognition. For each CNN, we performed a five-fold cross-validation on the 800 samples (200 samples per class) for both data sets, keeping at each fold 80% of the data for training and 20% of the data for validation, and reported the macro-average results. The trajectories were segmented into equally-sized temporal-window trajectories of 8 h in length. This particular temporal-window was selected because the mobility patterns of the vessels require some time to form [10]. During a one-hour window, for instance, the anchored pattern as described in Section 3.1 will most probably not have fully formed.

Table 3 illustrates the classification results of each model of the first data set (Pattern A).

The Keras package and a TensorFlow backend were employed along with the Python programming language for training the deep learning models. Keras is a simple-to-use neural network library built on top of Theano or TensorFlow.

Table 3. Classification performance obtained from different pre-trained CNN models for Pattern A.

Model	Labels	Precision	Recall	F1-Score	Overall Accuracy
VGG16	anchored	0.96	1.00	0.98	0.9888
	moored	1.00	1.00	1.00	
	underway	1.00	0.97	0.99	
InceptionV3	anchored	0.77	1.00	0.87	0.9222
	moored	1.00	1.00	1.00	
	underway	1.00	0.80	0.89	
NASNetLarge	anchored	0.76	0.96	0.85	0.9158
	moored	1.00	1.00	1.00	
	underway	0.97	0.80	0.88	
DenseNet201	anchored	0.68	1.00	0.81	0.8777
	moored	1.00	0.97	0.98	
	underway	1.00	0.71	0.83	

The results suggested that the VGG16 model achieved the best classification accuracy of almost 99% followed by InceptionV3 and NASNetLarge with almost the same accuracy of 92%. DenseNet201 presented the worst results with an accuracy of 88%. A sensitivity of 100% for anchored and moored and 97% for the underway class can be observed for VGG16. This practically means that confirmed anchored or moored vessel mobility patterns would be accurately identified as such 100% of the time, while confirmed underway patterns would be misclassified only 3% of cases. Furthermore, VGG16 showed high precision values for all classes, specifically 100% for moored and underway and 96% for anchored. This implies that there were no classes incorrectly classified as moored or underway from another class, while only one anchored class was incorrectly classified as underway. This can be seen in the confusion matrix of the VGG16 model with the best classification performance out of the five folds (Figure 11a). Regarding the F1-score, the moored class achieved 100% in all performance metrics, which means that the FPs and FNs were minimized at zero.

InceptionV3 and NASNetLarge presented very similar results. A precision of 100% in the moored class was observed, which means that there were no anchored or underway classes falsely misclassified as moored. On the other hand, the precision and F1-score regarding the anchored class depicted low results. Indeed, these low values were justified by a large number of FPs. Figure 11b,c depicts that the anchored class had seven misclassified cases as underway in both models. Furthermore, the models presented high sensitivity values for the anchored and moored class, while the underway class depicted a moderate value of 80%. Again, the moored class achieved 100% in all performance metrics.

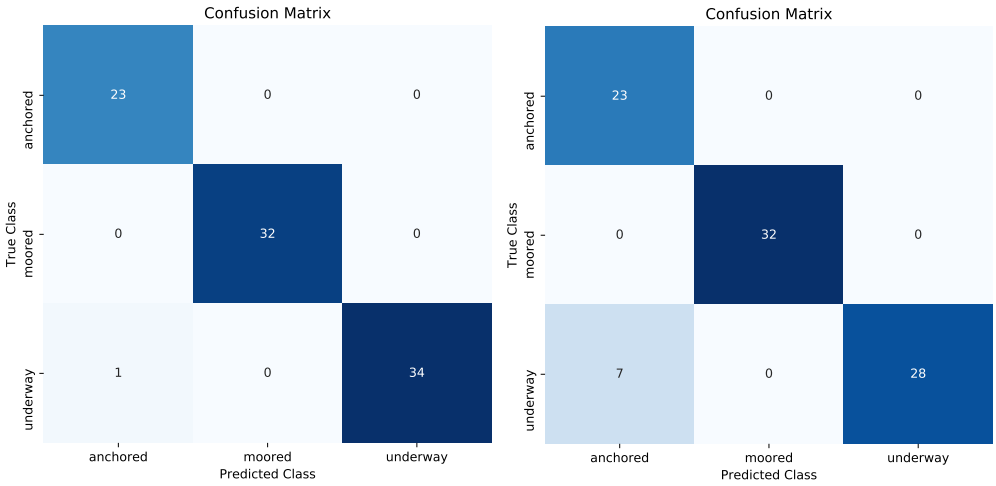
Although DenseNet201 presented the worst results in terms of accuracy, the moored class achieved again high values for all performance metrics. As the results suggest, the precision regarding the anchored class was very low at about 68% and followed the same trend as the InceptionV3 and NASNetLarge models for the particular class. However, the model was able to correctly identify confirmed anchored and moored classes as such, 100% and 97% of the time, respectively, while only 71% of confirmed underway cases would accurately be identified correctly. The anchored class presented 11 FPs, as illustrated in Figure 11d, which was strongly associated with a moderate value of the F1-score.

In order to visualize how well the examined models distinguished the classes in terms of the predicted probability, for each CNN, we visualize the ROC curves of the model with the best classification performance out of the five folds in Figure 12. Specifically, Figure 12a,b demonstrates the ROC curves for the models with the highest and lowest accuracy, VGG16 and DenseNet201, respectively. As expected, the area covered (AUC) by VGG16 was bigger as the model was able to discriminate the classes with a much higher probability in comparison with DenseNet201.

The same patterns can be observed between the different models for particular performance metrics. The precision of the anchored class followed a downward trend as the overall accuracy decreased. The same behavior was presented in the sensitivity of the underway class. On the other hand, moored class achieved high values in all performance metrics even with a moderate accuracy in the case of DenseNet201. Furthermore, all the CNN models presented quite high precision values for moored and underway classes regardless of the accuracy.

Subsequently, for each CNN, we visualize the loss and accuracy of the model with the best classification performance out of the five folds during their training in Figure 13. VGG16 demonstrated a smooth training process, as shown in Figure 13a, during which the loss gradually decreased and the accuracy increased. Moreover, it can be observed that the accuracy of both training and validation did not deviate much from one another, a phenomenon that can also be observed for the training and validation loss, indicating that the model did not overfit. The same behavior was presented by the NASNetLarge model, as shown in Figure 13c. On the other hand, in the case of InceptionV3 and DenseNet201, their validation loss was either increasing or fluctuating, as shown in Figure 13b,c, respectively, which means that the models most probably overfit. An interesting fact is that the

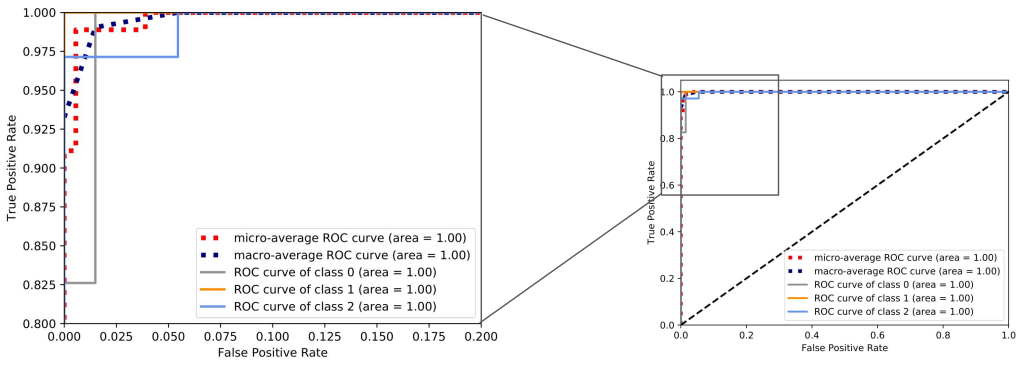
VGG16 model, which contained the least number of layers, achieved a better classification performance. This can be explained by the fact that neural networks with more hidden layers require more training data; thus, an even larger number of mobility pattern samples was required as an input in these networks.



(**a**) VGG16 (**b**) InceptionV3

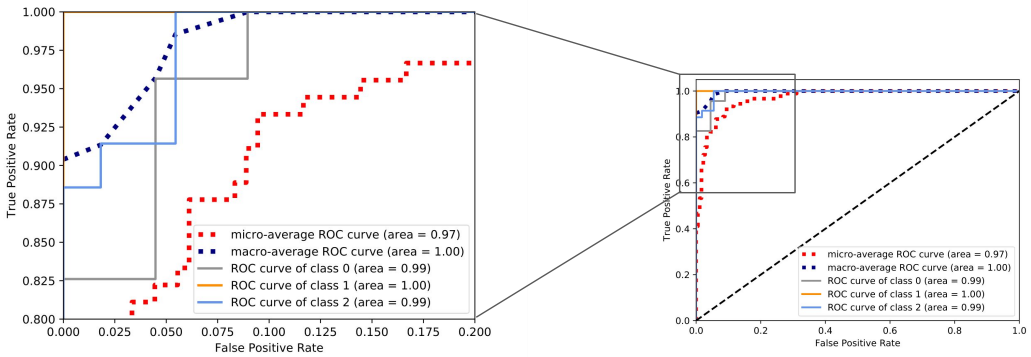
(**c**) NASNetLarge (**d**) DenseNet201

Figure 11. Confusion matrix of deep learning models for Pattern A with the best classification performance out of the five folds.



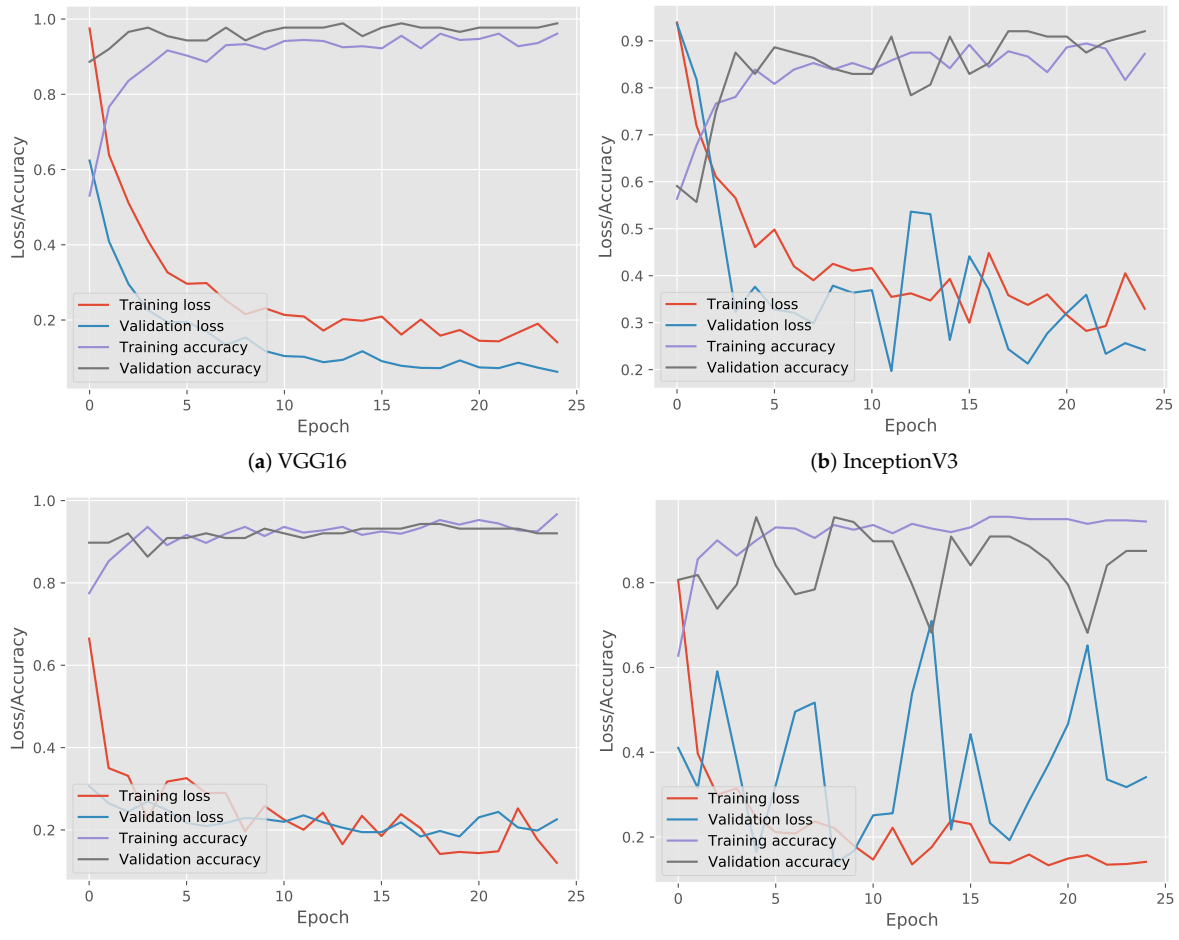
(**a**) VGG16

Figure 12. *Cont.*



(**b**) DenseNet201

Figure 12. ROC curve of deep learning models for Pattern A.



(**a**) VGG16 (**b**) InceptionV3

(**c**) NASNetLarge (**d**) DenseNet201

Figure 13. Accuracy and loss (train and test) of deep learning models for Pattern A.

Additionally, we compared our results with other methodologies for trajectory classification, which were illustrated in the recent survey of Wang et al. [5]. Well-known classifiers such as Random Forests (RFs) and Support Vector Machines (SVMs) are employed by state-of-the-art methodologies for trajectory classification, on features extracted from the AIS messages or data points of the trajectories. Thus, three features from the trajectories during each type of activity (anchored, moored, and underway) were extracted: (i) the average speed of the vessel, (ii) the standard deviation of its speed, and (iii) the haversine distance between the first and the last vessel position. These features were selected because they demonstrated distinct differences between these specific activities. A moored vessel

typically has a zero speed, while an anchored vessel moves with slightly greater speeds on average due to the effects of the wind and the currents of the sea. A vessel underway presents much higher speeds, and the distance between the first and the last position is greater compared to the distance of anchored or moored vessels. Subsequently, these aforementioned features were then fed to two classifiers, random forests and support vector machines. The methodologies employed in the comparison are shown below:

- M1: the proposed deep learning methodology of this paper (image classification by employing the VGG16 model, as it presented the best classification accuracy results).
- M2: the random forests classifier with the features extracted from the AIS messages of the trajectories.
- M3: the support vector machines classifier with the features extracted from the AIS messages of the trajectories.

Table 4 illustrates the cross-validation macro average results of the comparison between the different approaches. Results show that our proposed methodology (M1) surpassed the other methodologies (M2, M3) in terms of classification performance. M1 achieved an F1-score of 99.32%, and M2 and M3 achieved an F1-score of 96.18% and 91.1%, respectively.

Table 4. Comparison of the methodologies on Pattern A.

Methodology	Cross-Validation		
	Precision	Recall	F1-Score
M1	0.9905	0.9912	0.9932
M2	0.966	0.9603	0.9618
M3	0.924	0.9138	0.911

To further evaluate our methodology on vessel activities of utmost importance to the maritime authorities, we performed the same deep learning classifiers employed on Pattern A on the second data set (Pattern B), which contained fishing vessels engaged in trawling, longlining, moored at the port, or traveling towards a fishing area (underway). A fishing vessel is typically moored at a port, then travels towards a fishing area and is finally engaged in the fishing activity. After the fishing ends, the vessel travels back to the port and moors. Table 5 illustrates the classification results of each model for Pattern B.

Table 5. Classification performance obtained from different pre-trained CNN models for Pattern B.

Model	Labels	Precision	Recall	F1-Score	Overall Accuracy
VGG16	fishing	0.97	0.88	0.92	0.9473
	moored	1.00	1.00	1.00	
	underway	0.86	0.96	0.91	
InceptionV3	fishing	0.74	1.00	0.85	0.8736
	moored	1.00	0.97	0.99	
	underway	1.00	0.58	0.73	
NASNetLarge	fishing	0.6	0.97	0.74	0.7578
	moored	0.97	0.94	0.96	
	underway	1.00	0.23	0.38	
DenseNet201	fishing	0.63	1.00	0.77	0.7894
	moored	1.00	0.94	0.97	
	underway	1.00	0.31	0.47	

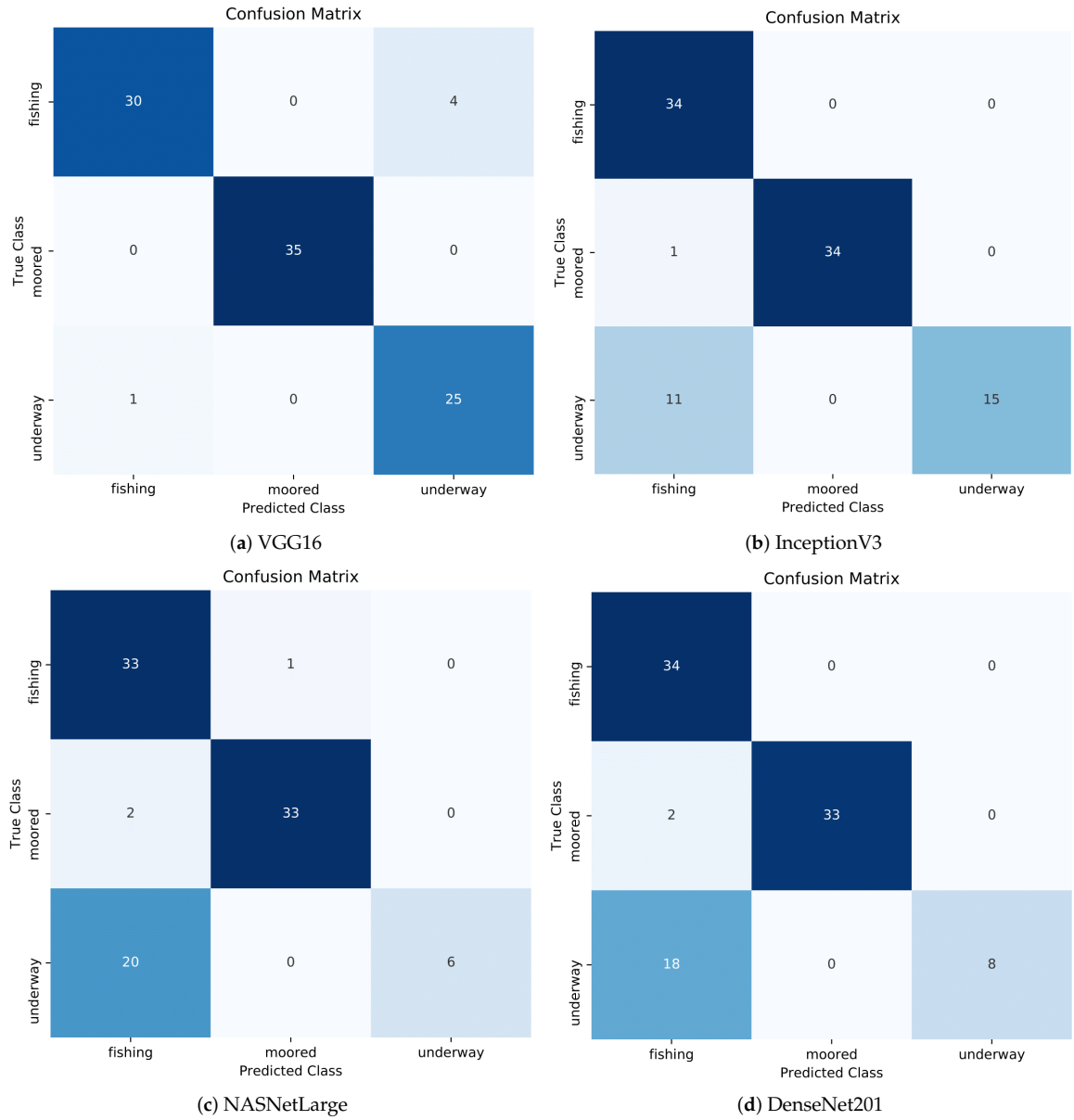
As the results suggest, VGG16 model achieved again the best classification accuracy of almost 95% followed by InceptionV3 with an accuracy of 87%. NASNetLarge exhibited the worst results with an accuracy of 76%, while DenseNet201 presented slightly better results with an overall accuracy of 79%. A recall (sensitivity) of 100% and 96% for the moored and underway class, respectively, can be observed for VGG16. This practically means that confirmed moored vessel mobility patterns would be accurately identified as such 100% of the time, while confirmed underway patterns would be misclassified for only 4% of cases. In contrast, fishing presented a moderate recall of 88%. Furthermore, VGG16 presented high precision values for fishing and moored and a moderate precision for underway. This implies that there were no classes incorrectly classified as moored from another class, while only one fishing class was incorrectly classified as underway. The moderate precision of underway was justified by a moderate number of FPs. Indeed, four underway classes were incorrectly classified as fishing, as shown in Figure 14a, which represents the confusion matrix of the VGG16 model with the best classification performance out of the five folds. The moored class achieved an F1-score of 100%, while the remaining classes exhibited a high percentage of about 92% for fishing and 91% for underway.

In the InceptionV3 model, the moored class presented high values for all performance metrics. Furthermore, a precision of 100% in both underway and moored was observed. On the other hand, the precision of the fishing class presented a low value of 74% and justified by the large number of FPs (12 in total), as shown in Figure 14b. Furthermore, the recall of the underway class was low, 58%, as well as the F1-score, which in turn justified the large number of FNs (11 in total). The other classes exhibited large values regarding recall, 100% for fishing and 97% for moored.

NASNetLarge and DenseNet201 presented several similarities. The fishing class had a low value of precision, almost 60% in both models. Figure 14c,d depicts that the fishing class had 22 and 20 misclassified cases, respectively. On the other hand, high precision values were observed for the moored and underway classes in both models. The moored class presented yet again high values in all performance metrics. Furthermore, recall regarding the underway class was very low at about 23% in NASNetLarge and 31% in DenseNet201 and followed the same trend as the InceptionV3 model for the particular class. In combination with the low values of the F1-score, only a small number of confirmed underway classes were identified correctly.

Figure 15 demonstrates the ROC curves for the models with the highest and lowest accuracy for Pattern B, VGG16 in Figure 15a and NASNetLarge in Figure 15b, respectively. For these two CNNs, we visualized the ROC curves of the model with the best classification performance out of the five folds.

Subsequently, for each CNN, we visualized the loss and accuracy of the model with the best classification performance out of the five folds during their training for Pattern B, as shown in Figure 16. VGG16 demonstrated a smooth training process for Pattern B, as shown in Figure 16a, during which, the loss gradually decreased and the accuracy increased. Almost the same behavior was presented by the InceptionV3 model, as shown in Figure 16b, except for the last epochs where the loss increased. On the other hand, in the case of NASNetLarge and DenseNet201, their validation loss was either increasing or fluctuating, as shown in Figure 16c,d, respectively, which means that the models most probably overfit. Indeed, the growth rate of validation loss was extremely high in the case of NASNetLarge.

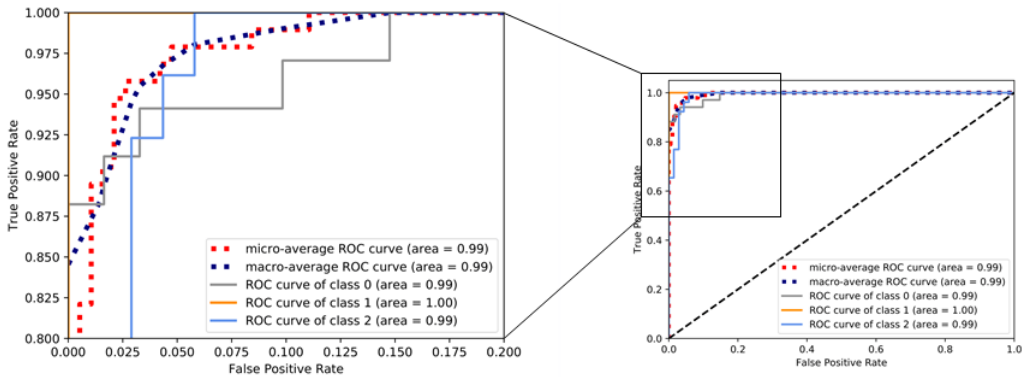


(**a**) VGG16 (**b**) InceptionV3

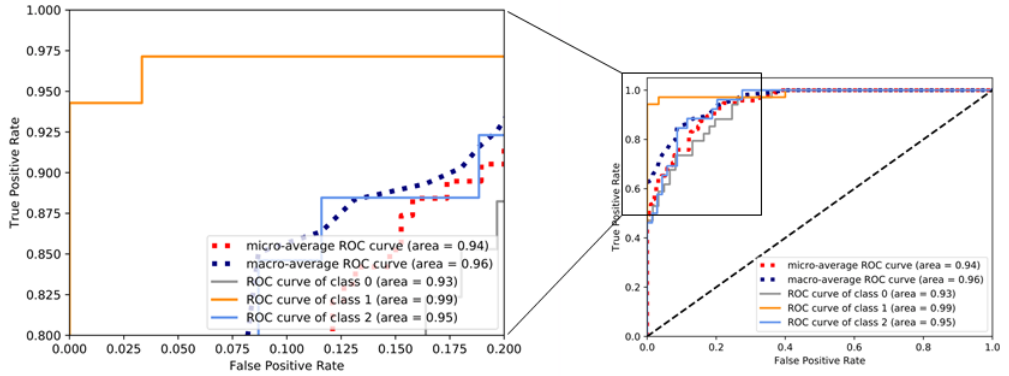
(**c**) NASNetLarge (**d**) DenseNet201

Figure 14. Confusion matrix of deep learning models for Pattern B with the best classification performance out of the five folds.

Finally, we compared our classification methodology results to the approach proposed in [10] for the fishing vessels (M4). The authors in [10] used a set of features suitable for the fishing activities, which were then used by random forests to classify the vessel activities. Moreover, we also used the methodology of the previous experiment (M2), which uses features that were suitable for Pattern A. Similar to the previous set of experiments, we employed only the VGG16 model as it presented the best overall classification accuracy. The macro average results are reported in Table 6. The results again validate that the proposed approach (M1) was by far more accurate than the set of features selected specifically for the classification of fishing activities (M4). Furthermore, the results demonstrate that the methodology M2 did not perform well (F1-score of 76.2%). This is explained by the fact that the features used for Pattern A were not able to discriminate Pattern B. On the other hand, the proposed methodology (M1) was able to perform well on both sets of patterns due to the fact that the patterns had distinct visual differences.

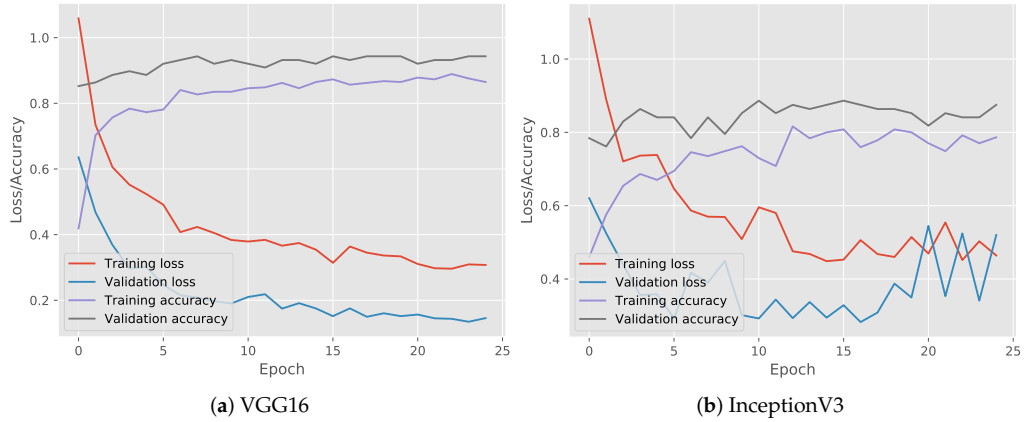


(**a**) VGG16



(**b**) NASNetLarge

Figure 15. ROC curve of deep learning models for Pattern B.



(**a**) VGG16 (**b**) InceptionV3

(**c**) NASNetLarge (**d**) DenseNet201

Figure 16. Accuracy and loss (train and test) of deep learning models for Pattern B.

Table 6. Comparison of methodologies on Pattern B.

Methodology	Cross-Validation		
	Precision	Recall	F1-Score
M1	0.9452	0.9528	0.9465
M2	0.7523	0.772	0.7620
M4	0.8455	0.8436	0.8446

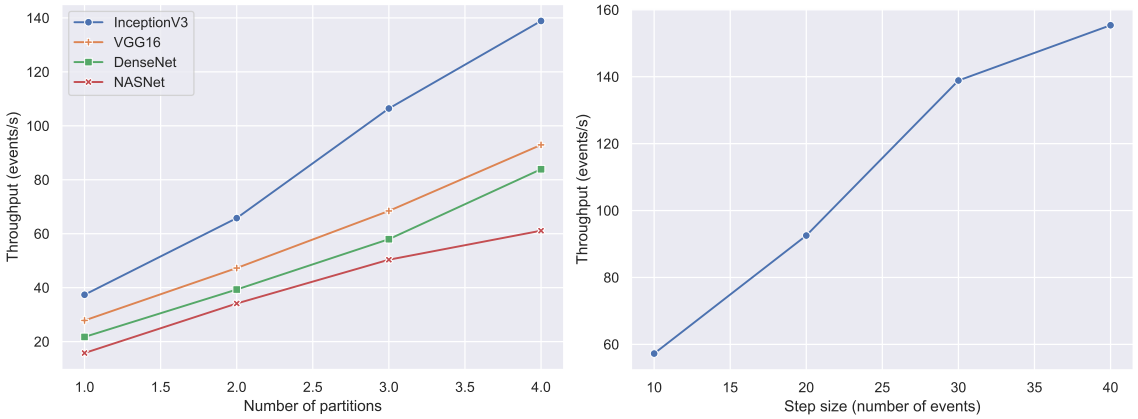
4.3. Streaming Evaluation

In this section, we provide experimental results of the execution performance of the proposed streaming methodology. To this end, different sets of experiments were conducted in order to evaluate the streaming application's:

- scalability,
- throughput,
- latency,
- execution performance with a GPU.

The experimental evaluation presented below was performed on a cluster consisting of four machines with four cores (Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00 GHz) and 8 GB of RAM each, totaling 16 cores and 32 GB of RAM, running Ubuntu 18.04 LTS with Python 3.6 and Apache Kafka 2.4.0.

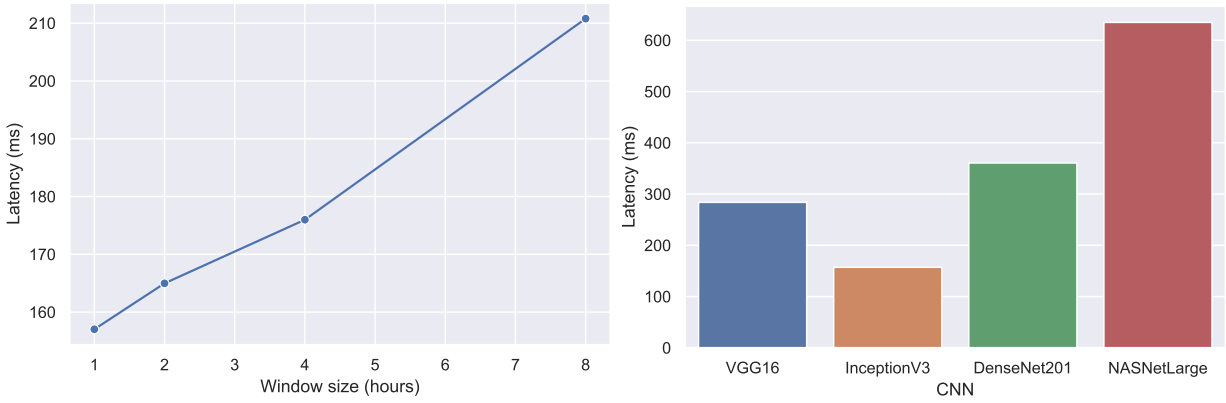
To evaluate the scalability of our approach, we ran experiments using 1, 2, 3, and 4 machines incrementally. Each machine employed a prediction module and kept the window size and window step constant equal to 4 h and 30 events, respectively. As mentioned in Section 3.4, the streaming methodology used a sliding window over the streams of AIS messages to perform the classification. A window size of 8 h was sufficient to have a fully formed pattern per vessel [10]. Since the highest transmission rate of the AIS protocol was one message every 2 s, a window step of 30 events or messages implies that a classification will be triggered by the prediction modules every one minute, a time period during which the pattern would not have changed significantly. Furthermore, we repeated each experiment four times to evaluate the four different CNN architectures (VGG16, InceptionV3, DenseNet, NASNet). Figure 17a illustrates the throughput of each CNN architecture with a different number of machines or partitions. It can be observed that the proposed streaming methodology can benefit when more machines and prediction modules were used regardless of the employed CNN architecture. Moreover, it is apparent that InceptionV3 was the fastest CNN architecture achieving a maximum throughput of 140 events per second, outperforming all of its contestants. The next CNN architecture was VGG16 with a throughput of 90 events per second, which is similar to DenseNet's throughput of 80 events per second. The slowest CNN architecture was NASNet, which achieved a maximum throughput of 60 events per second. The performance of the two last architectures, which presented the worst results, can be explained by the fact that these architectures are more complex than the former two, consisting of more hidden layers and pooling layers. To further evaluate the throughput of the proposed methodology, we selected InceptionV3, which was the fastest CNN architecture, and performed experiments using four machines and a varying number of window steps. Figure 17b visualizes the results of the aforementioned experiment using window steps that range from 10 events to 40 events. The value of the window step denotes how often the classification will be triggered; thus, the smaller the number of steps is, the larger the amount of classifications that are triggered, resulting in a lower throughput. Indeed, this was confirmed by the experiments in Figure 17b, where the throughput was 155 events per second when using a step of 40 and 57 events per second when using a step of 10.



(**a**) (**b**)

Figure 17. Throughput experiments. (**a**) Throughput of each CNN architecture with different numbers of partitions. (**b**) Throughput of InceptionV3 with different window steps.

The first experiment, which yielded the results of Figure 17a, also provided the average processing latency of each CNN architecture in Figure 18b. Processing latency is defined as the amount of time required by a CNN to perform the image classification. Again, it can be observed that InceptionV3 was the fastest CNN architecture with a processing latency of 157 ms. Next was the VGG16 network with a latency of 283 ms, followed by DenseNet (360 ms) and NASNet (634 ms). To evaluate the image latency, which is defined as the amount of time required to create an image from the AIS messages, we kept the same configurations of the first experiment (four machines and a window step of 30 events), but changed the window size. Figure 18a illustrates the average image latency with a window size of 1, 2, 4, and 8 h. Since a larger time window contains more AIS messages, it is expected that the creation of the image will be greater in larger window settings. As shown and confirmed in Figure 18a, the average image latency in a one-hour window was 157 ms, and in an eight-hour window, the latency could reach up to 210 ms.



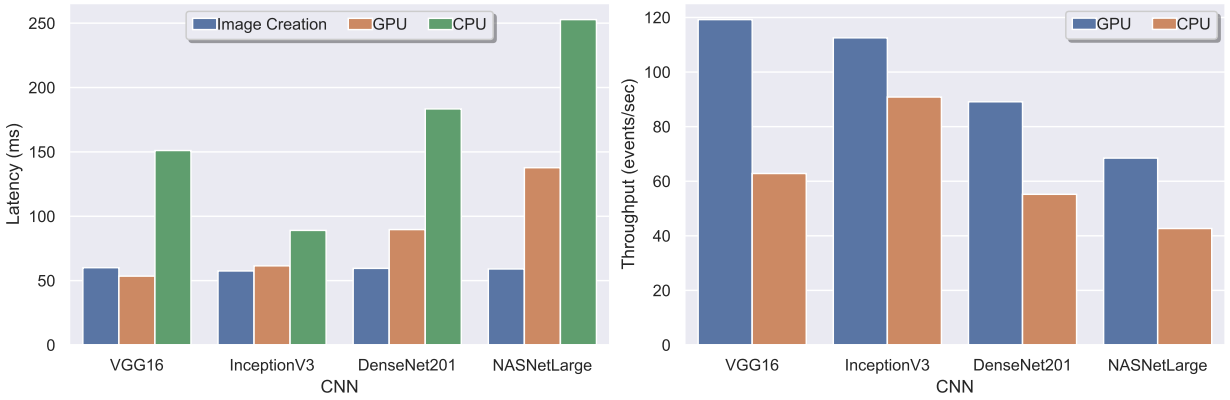
(**a**) (**b**)

Figure 18. Latency experiments. (**a**) Image latency with different window sizes. (**b**) Latency of each CNN architecture.

To further evaluate our approach, we demonstrate experimental results with and without the use of a GPU. The following experiments presented below were performed on a commodity machine with eight logical cores (Intel(R) Core(TM) i7-7700HQ CPU @ 2.8 GHz), 32 GB of RAM, and a GPU (NVIDIA 1050 with 4 GB of VRAM), running Windows 10. To demonstrate the performance boost by the GPU, we ran an experiment with a window size and step of 4 h and 30 events, respectively, twice for each CNN architecture, with and without the use of the GPU. The experimental results are presented in Figure 19. Specifically, Figure 19a illustrates the image latency and the latency of each

CNN architecture. The image latency in each experiment was approximately 60 ms (at least two-times faster than the same experiment with a lower frequency CPU in Figure 18a). On the other hand, a tremendous performance gain can be observed in the processing latency of each CNN. The processing latency of VGG16, InceptionV3, DenseNet, and NASNet was 150 ms, 80 ms, 180 ms, and 250 ms, respectively, without the use of GPU, while when a GPU was employed, the processing latency reduced to 50 ms, 60 ms, 80 ms, and 140 ms correspondingly. The same pattern can also be observed in the throughput of our approach in Figure 19b, where the throughput of each CNN architecture increased. An indicative example is VGG16, where the throughput from 60 events per second was increased to 120 events per second.

Finally, it is worth noting that all of the experiments presented in this section achieved a throughput higher than the transmission rate of AIS messages per receiver (five to eight messages per second), as stated in Section 3.4, making the prediction module suitable to be employed on a terrestrial receiver basis.



(**a**) Latency results (**b**) Throughput results

Figure 19. Execution performance experiments with and without the GPU.

4.4. Compression Evaluation

In order to evaluate the trajectory compression algorithms, some validation criteria were employed. The main validation solutions include performance and accuracy metrics. For performance, the following metrics were considered: (i) compression ratio and (ii) compression time. The compression ratio refers to the ratio of the points remaining after compression (a) compared to the original points (n), being defined as $\frac{a}{n}$, while compression time refers to the total time required for the compression execution. On the other hand, accuracy metrics are used to measure information loss and therefore the accuracy of the compression algorithms.

At this point, it should be mentioned that in order to evaluate the compression results, we employed a subset of the first data set, which is mentioned in Section 4.1. Specifically, this sub-data set contained information for 472 unique vessels that were monitored for a three-day period starting 1 March 2020 and ending 3 March 2020. Our goal was to perform some preliminary experiments in order to demonstrate the effect of trajectory compression over the classification accuracy of mobility patterns.

Figure 20 illustrates the compression ratio achieved by the different algorithms. The highest compression was achieved by *DP* with a ratio of 94%. Compression was slightly lower in *TR*, but nevertheless remained extremely high at about 92%. Algorithms that considered time information to decide whether or not to discard a point tended to preserve more points in a trajectory. *DP* possessed the highest compression rates as it treated a trajectory as a line in two-dimensional space and did not consider the temporal aspect. The compression ratio was decreased in the case of *SP*, but maintained at a high level of about 80%. *HB* and *SP_HD* held the lowest compression rates of 59% and 54%, respectively. *SP_HD* incorporated two thresholds simultaneously, and it was expected to present the

lowest compression ratio. Generally, the less the compression ratio is, the more points are included in the resulting set.

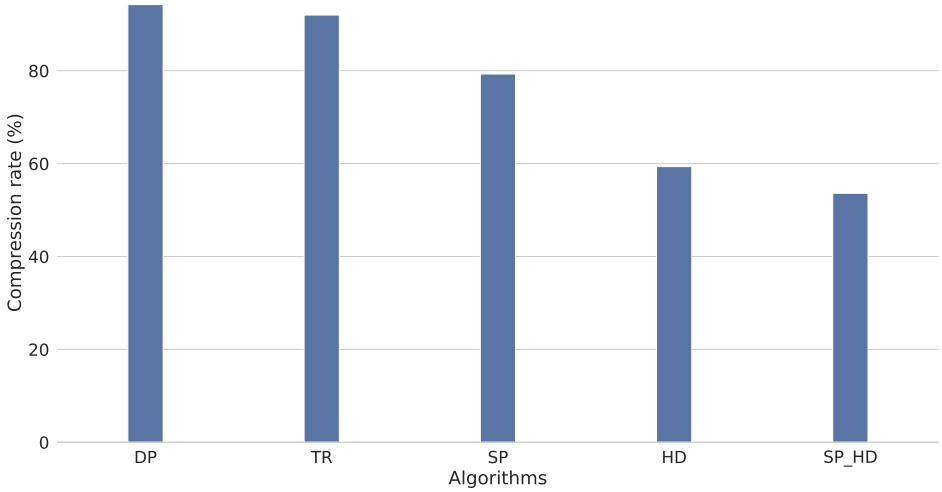


Figure 20. Compression ratio achieved by different algorithms.

Besides the compression ratio, it is important to assess the execution time of each algorithm as it constitutes an important index to measure efficiency. Figure 21 presents the execution times of all algorithms. *DP* and *TR* presented the lowest execution among all algorithms. However, the compression presented the highest rates for these algorithms. In fact, *DP* was executed in only 180 s, almost four-times faster than *TR*. *SP* spent considerably more time than *DP* and *TR*. Specifically, *SP* was 27- and seven-times slower than *DP* and *TR*, respectively. The execution time of *HD* and *SP_HD* was quite high, more than two hours for both algorithms. As expected, *SP_HD* had the worst performance due to its exponential complexity.

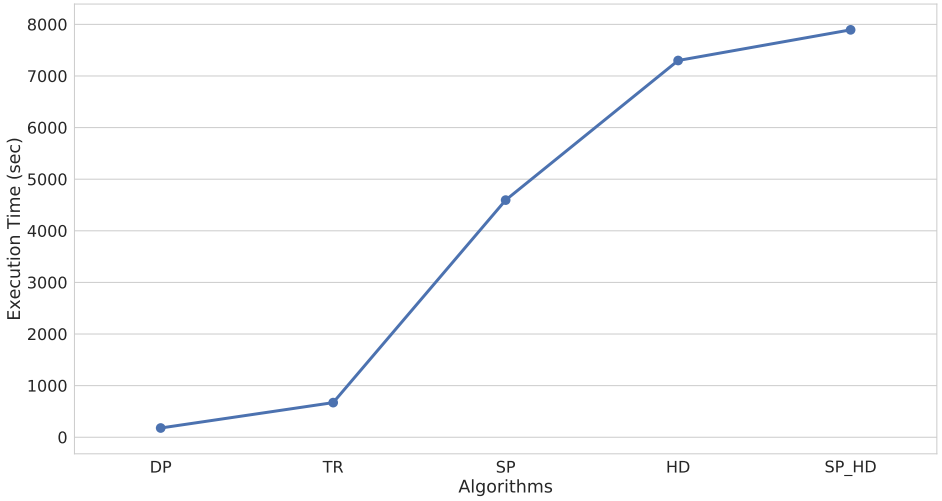


Figure 21. Execution times of the compression algorithms.

As stated in Section 3.5, the examined algorithms are lossy, which means that they present some information loss as compressed data cannot be recovered or reconstructed exactly [83]. Thus, it is important to measure the compression performance by evaluating the quality of the compressed data. Existing techniques evaluate the trajectory compression algorithms using various error metrics such as spatial distance, synchronized Euclidean distance, time-distance ratio, speed, heading, acceleration, and enclosed area. Muckell et al. [77] employed the average SED and average speed error, while in [75], the

same metrics with the addition of heading error were applied. Furthermore, three assessment criteria were used in [84], average SED, spatial, and speed error, while Liu et al. [85] evaluated several compression algorithms through the average PD, SED, and enclosed area. Additionally, in [80], the error was provided by a formula that measured the mean distance error between the original and compressed trajectory.

As there is a trade-off between the achieved compression ratio and the acceptable error, the effectiveness of the trajectory compression algorithms was evaluated in terms of "information" quality obtained after compression using deep learning. Deep learning was applied over the compressed trajectory data in order to evaluate their accuracy. To the best of our knowledge, this is the first research work that has investigated and evaluated the impact of trajectory compression techniques over vessel mobility pattern classification through neural networks.

Each unique vessel inside the examined sub-data set consisted of a different number of initial points, which ranged from one to 25,000. In order to evaluate the compression results, forty-seven unique vessels (10%) were selected from the total of 472 out of a normal distribution of the points per vessel. Then, the sub-trajectories of these different vessels were extracted and visualized as trajectory images, as described in Section 3.2, for both the original (uncompressed) and compressed data. Finally, we performed vessel pattern classification on both trajectory data (compressed and uncompressed), by employing the VGG16 model, as it achieved the highest accuracy. Table 7 illustrates the percentage $\frac{m}{n}$ of correctly classified mobility patterns extracted from the sub-trajectories' images for each compression algorithm (m) compared to the total number of examined mobility patterns (n). In fact, the higher the percentage, the better the quality of the data obtained after compression was.

Table 7. Percentage of correctly classified mobility patterns for each compression algorithm. DP, Douglas–Peucker; TR, Time Ratio; SP, Speed based; HD, Heading based.

Compression Algorithms				
DP	**TR**	**SP**	**HD**	**SP_HD**
65%	69%	74%	93%	86%

The results demonstrate that accuracy was strongly related to the compression ratio. *DP* presented the highest compression, but exhibited the worst accuracy, about 65%. Similarly, *TR* achieved extremely high compression, but the accuracy was low (69%), slightly better than *DP*. *SP* neither had the worst, nor the best results and presented a moderate accuracy of 74%. Nevertheless, the compression obtained in the case of *SP* was 12% lower than *TR*, while the accuracy between these algorithms differed only 5%. *HD* achieved the highest accuracy as it presented the most correctly classified vessel patterns from the sub-trajectory images with a ratio of 93%. Accuracy was slightly lower for *SP_HD*, but nevertheless remained high at about 86%. The compression achieved in the case of *HD* and *SP_HD* was significantly lower in comparison with the other algorithms, but it was accompanied by extremely high accuracy, especially in the case of *HD*.

5. Discussion

The proposed work provided a novel approach for the classification of trajectories in near real time, a task that is of utmost importance in the surveillance of vessels in the maritime domain. The novelty behind our approach lies in the fact that trajectories were converted into a space normalization format that resembled an image, which were then classified by a convolutional neural network, a methodology that to the best of our knowledge has received limited study in the literature. This fusion of the fields of trajectory classification and computer vision can deliver results comparable to or even better than the state-of-the-art (see Section 4.2).

Classification techniques may involve a number of pre-processing steps in order to develop a classifier that yields optimal results. Initially, a thorough data analysis and cleaning are typically performed. Then, a clustering technique may be employed, which often requires some form of user-defined parameters to group similar trajectories together. Afterwards, a careful selection and analysis of optimal features for the label or labels to be identified followed. Finally, the analysis and selection of several classifiers were performed to achieve high-precision results [5]. Therefore, the overall process of feature selection and classification is not an easy task. This difficulty of finding the best classifier can be observed in the fishing patterns where vessels pose a similar behavior; thus, different features and classifiers need to be employed for each pattern [14,15]. Although mobility patterns can be similar (e.g., fishing patterns have similar behaviors in terms of the speed and frequency of turns [10,14]), they differ visually (see Figure 1d,e). This distinct visual difference of mobility patterns is used by computer vision techniques for their classification. The main advantage of image classification for mobility patterns over other techniques is (1) the decrease of the complexity of the aforementioned steps, (2) the removal of user-defined parameters, and (3) the distinct visual difference of mobility patterns. Although the data cleaning process cannot be avoided, the parameter selection, feature selection, analysis, and the search for an optimal classifier can be entirely skipped. For the classification of an image, only one classifier is needed (e.g., CNNs) whether this image illustrates a fishing pattern or a Search And Rescue (SAR) pattern.

Due to the similarity of fishing patterns (trawling, longlining, purse-seiners), Souza et al. [14] proposed a different classification scheme for each pattern (three classifiers with three different sets of features), achieving an F1-score of 97%. Each classifier was trained to identify trajectories between fishing or underway, where fishing corresponded to the fishing activity of the respective vessel type. This classification performance was comparable to the performance of the proposed methodology (≈95%), which used a convolutional neural network for the classification of four patterns in total (Pattern A and B). The authors in [10] presented a set of features that can be extracted in real time and classified the same fishing activities that this work did, achieving a lower classification performance (87%). The main disadvantage is that these features cannot be used to classify other mobility patterns, and they are only suitable for the fishing patterns in the study. Kapadais et al. [2] proposed a time-series shapelet classification technique to identify SAR patterns. The main drawback in such techniques is that they require data to be available at fixed intervals (e.g., hourly, weekly, monthly, yearly), a characteristic that is not present in AIS messages (see Section 4.1). Therefore, techniques need to be devised that can fill in the gaps or remove data points in order to create a time-series suitable for classification, a problem that can be avoided by our proposed approach.

Compared to Chen et al. [48] in which the authors converted AIS trajectories into a Mobility-Based trajectory structure (MB) that resembled a low-resolution image, our approach can create high-resolution images able to capture a trajectory's behavior in its entirety. Since the surveillance space in [48] was segmented into large cells, cases where the vessels perform micro-movements that eventually form a different mobility pattern such as fishing activities will be misclassified. The authors in [48] aimed at the classification of the transportation mode (e.g., carrying cargo, tankers, towing, or fishing) of vessel trips when traveling from an origin port to a destination port and achieved an F1-score of 84.7%. A similar approach was employed in [50] for the classification of the transportation mode of civilians (e.g., walking, bike, bus, taxi), achieving a maximum accuracy of 83.2%.

Despite the fact that several classification methodologies have been proposed in the literature, each one has its limitations and possible drawbacks. Similarly, although the proposed methodology tackled most of the issues found in the previous literature, it can be observed that there is a cold start problem. This means that the proposed approach can accurately classify vessel activities only when enough AIS messages are accumulated for a predefined period of time (e.g., four-hour time window) and not when the vessel first appears in the surveillance area. Furthermore, deep neural networks are computationally

expensive, not only during the training process, but in the classification process as well. From the experimental results, it can be seen that the maximum throughput for data originating from a single AIS receiver was approximately 155 events per second without a GPU. At a global scale where the frequency can reach up to 16,000 events per second, a huge cluster is required to handle the volume and the velocity of the data. Therefore, the solution to address the high-frequency and high-volume streams of AIS data is to deploy one prediction module (see Section 3.4) per AIS receiver, thus creating a global distributed network of prediction modules.

6. Conclusions and Future Work

In this work, we presented a novel and high-accuracy trajectory classification approach over streams of AIS messages. The goal of our methodology was to provide an efficient and alternative way to treat the problem of streaming vessel activity classification as an image classification task using state-of-the-art deep learning algorithms and to create a universal approach for the classification of vessel activities. This universal approach can be achieved through the distinct visual difference that most of the mobility patterns, if not all, have in the maritime domain. Therefore, a common classifier can be used for all patterns after they have been converted to images.

To demonstrate the applicability of our work in real-world conditions, two real-world data sets of AIS messages were used, one collected from a terrestrial, vessel-tracking receiver installed on our premises and another one extracted from the MarineTraffic database. As deep learning approaches require a large amount of data in order to perform an accurate classification, transfer learning was employed. Experimental evaluation showed that the proposed approach achieved state-of-the-art classification accuracy. The proposed approach was evaluated on five mobility patterns of the maritime domain in total, namely anchored, moored, underway, trawling, and longlining. The achieved classification performance exceeded 89% for all CNN models in the first data set (Pattern A), with VGG16 yielding the best accuracy of 99%. In order to evaluate the methodology in a real-world use case of interest, we demonstrated its classification performance on four mobility patterns of fishing vessels: trawling, longlining, moored, and underway. The VGG16 model presented again the best classification results with an accuracy of 95%.

Scalability experiments demonstrated that the proposed methodology can deal with the event rate of real-world streams of AIS messages and can scale up to multiple machines, allowing a near real-time execution performance. InceptionV3 achieved a maximum throughput of 140 events per second, outperforming all of its contestants, followed by VGG16 with a throughput of 90 events per second. Regarding latency, InceptionV3 was again the fastest CNN architecture with a processing latency of 157 ms followed by VGG16's latency of 283 ms. In addition, the superiority of GPU versus CPU was verified through experiments regarding throughput and latency.

Preliminary experiments with compression algorithms demonstrated that streams of AIS messages can compress more than 50% of the original number of messages with a relatively low impact on the classification performance. The heading-based compression algorithm achieved the best compression ratio, while the accuracy remained at a fairly high level; only 7% of cases were misclassified.

As future work, we plan on collecting more images to further improve the performance of the deep neural networks and identify more vessel mobility patterns to enrich our deep learning models with more labels. Finally, online compression algorithms will be developed and evaluated to decrease the system's latency.

Author Contributions: Ioannis Kontopoulos, Antonios Makris and Konstantinos Tserpes worked on the conceptualization of the work. Ioannis Kontopoulos and Antonios Makris worked on the methodology and, with Konstantinos Tserpes, developed the final solution. Ioannis Kontopoulos and Antonios Makris worked on the experiments and the original draft preparation. Konstantinos Tserpes helped with the writing—review and editing—of the final manuscript and performed the overall project supervision. All authors read and agreed to the published version of the manuscript.

Funding: This research was co-financed by Greece and the European Union (European Social Fund (ESF)) through the Operational Programme "Human Resources Development, Education and Lifelong Learning 2014–2020" in the context of the project "A global, distributed surveillance system for early detection and analysis of anomalous vessel trajectories (GLASSEAS)" (MIS 5049026). Furthermore, this work was supported by the MASTER and SmartShip Projects through the European Union's Horizon 2020 research and innovation program under Marie-Sklodowska Curie Grant Agreement Nos. 777695 and 823916, respectively. The work reflects only the authors' view, and the EU Agency is not responsible for any use that may be made of the information it contains.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Restrictions apply to the availability of these data. Data was obtained from MarineTraffic and part of them are available at https://doi.org/10.5281/zenodo.3754481 (accessed on 5 April 2021) with the permission of MarineTraffic.

Acknowledgments: This work was supported in part by MarineTraffic, which provided data access for research purposes.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lee, J.; Han, J.; Li, X.; Gonzalez, H. *TraClass*: Trajectory classification using hierarchical region-based and trajectory-based clustering. *Proc. VLDB Endow.* **2008**, *1*, 1081–1094. [CrossRef]
2. Kapadais, K.; Varlamis, I.; Sardianos, C.; Tserpes, K. A Framework for the Detection of Search and Rescue Patterns Using Shapelet Classification. *Future Internet* **2019**, *11*, 192. [CrossRef]
3. Mazzarella, F.; Vespe, M.; Damalas, D.; Osio, G. Discovering vessel activities at sea using AIS data: Mapping of fishing footprints. In Proceedings of the 17th International Conference on Information Fusion, FUSION 2014, Salamanca, Spain, 7–10 July 2014; pp. 1–7.
4. da Silva, C.L.; Petry, L.M.; Bogorny, V. A Survey and Comparison of Trajectory Classification Methods. In Proceedings of the 2019 8th Brazilian Conference on Intelligent Systems (BRACIS), Salvador, Brazil, 15–18 October 2019; pp. 788–793.
5. Wang, D.; Miwa, T.; Morikawa, T. Big Trajectory Data Mining: A Survey of Methods, Applications, and Services. *Sensors* **2020**, *20*, 4571. [CrossRef] [PubMed]
6. Bachar, M.; Elimelech, G.; Gat, I.; Sobol, G.; Rivetti, N.; Gal, A. Venilia, On-line Learning and Prediction of Vessel Destination. In Proceedings of the 12th ACM International Conference on Distributed and Event-Based Systems, DEBS 2018, Hamilton, New Zealand, 25–29 June 2018; Hinze, A., Eyers, D.M., Hirzel, M., Weidlich, M., Bhowmik, S., Eds.; ACM: New York, NY, USA, 2018; pp. 209–212. [CrossRef]
7. Bodunov, O.; Schmidt, F.; Martin, A.; Brito, A.; Fetzer, C. Real-time Destination and ETA Prediction for Maritime Traffic. In Proceedings of the 12th ACM International Conference on Distributed and Event-based Systems, DEBS 2018, Hamilton, New Zealand, 25–29 June 2018; Hinze, A., Eyers, D.M., Hirzel, M., Weidlich, M., Bhowmik, S., Eds.; ACM: New York, NY, USA, 2018; pp. 198–201. [CrossRef]
8. Kontopoulos, I.; Spiliopoulos, G.; Zissis, D.; Chatzikokolakis, K.; Artikis, A. Countering Real-Time Stream Poisoning: An Architecture for Detecting Vessel Spoofing in Streams of AIS Data. In Proceedings of the 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, DASC/PiCom/DataCom/CyberSciTech 2018, Athens, Greece, 12–15 August 2018; IEEE Computer Society: Washington, DC, USA, 2018; pp. 981–986.
9. Kontopoulos, I.; Chatzikokolakis, K.; Zissis, D.; Tserpes, K.; Spiliopoulos, G. Real-time maritime anomaly detection: Detecting intentional AIS switch-off. *Int. J. Big Data Intell.* **2020**, *7*, 85–96. [CrossRef]
10. Kontopoulos, I.; Chatzikokolakis, K.; Tserpes, K.; Zissis, D. Classification of vessel activity in streaming data. In Proceedings of the DEBS '20: The 14th ACM International Conference on Distributed and Event-Based Systems, Montreal, QC, Canada, 13–17 July 2020; Gascon-Samson, J., Zhang, K., Daudjee, K., Kemme, B., Eds.; ACM: New York, NY, USA, 2020; pp. 153–164. [CrossRef]
11. Rawat, W.; Wang, Z. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput.* **2017**, *29*, 2352–2449. [CrossRef] [PubMed]
12. Zhong, S.; Liu, Y.; Liu, Y. Bilinear deep learning for image classification. In Proceedings of the 19th ACM International Conference on Multimedia, Scottsdale, AZ, USA, 28 November–1 December 2011; pp. 343–352. [CrossRef]
13. Wu, J.; Yu, Y.; Huang, C.; Yu, K. Deep multiple instance learning for image classification and auto-annotation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3460–3469. [CrossRef]
14. Souza, E.; Boerder, K.; Worm, B. Improving Fishing Pattern Detection from Satellite AIS Using Data Mining and Machine Learning. *PLoS ONE* **2016**, *11*, e0158248. [CrossRef]

15. Jiang, X.; Silver, D.L.; Hu, B.; de Souza, E.N.; Matwin, S. Fishing Activity Detection from AIS Data Using Autoencoders. In *Advances in Artificial Intelligence—29th Canadian Conference on Artificial Intelligence, Canadian AI 2016, Victoria, BC, Canada, 31 May–3 June 2016*; Khoury, R., Drummond, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9673, pp. 33–39, [CrossRef]
16. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, 7–9 May 2015.
17. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826. [CrossRef]
18. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning Transferable Architectures for Scalable Image Recognition. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8697–8710. [CrossRef]
19. Huang, G.; Liu, Z.; van der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269. [CrossRef]
20. Ankerst, M.; Breunig, M.M.; Kriegel, H.P.; Sander, J. OPTICS: Ordering points to identify the clustering structure. *ACM Sigmod Rec.* **1999**, *28*, 49–60. [CrossRef]
21. Lee, J.; Han, J.; Whang, K. Trajectory clustering: A partition-and-group framework. In Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, 12–14 June 2007; pp. 593–604. [CrossRef]
22. Kontopoulos, I.; Varlamis, I.; Tserpes, K. A distributed framework for extracting maritime traffic patterns. *Int. J. Geogr. Inf. Sci.* **2020**, 1–26. [CrossRef]
23. Vouros, G.A.; Vlachou, A.; Santipantakis, G.M.; Doulkeridis, C.; Pelekis, N.; Georgiou, H.V.; Theodoridis, Y.; Patroumpas, K.; Alevizos, E.; Artikis, A.; et al. Big Data Analytics for Time Critical Mobility Forecasting: Recent Progress and Research Challenges. In Proceedings of the 21st International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, 26–29 March 2018; pp. 612–623. [CrossRef]
24. Chuaysi, B.; Kiattisin, S. Fishing Vessels Behavior Identification for Combating IUU Fishing: Enable Traceability at Sea. *Wirel. Pers. Commun.* **2020**, *115*, 2971–2993. [CrossRef]
25. Saini, R.; Roy, P.; Dogra, D. A Segmental HMM based Trajectory Classification using Genetic Algorithm. *Expert Syst. Appl.* **2017**, *93*, 169–181. [CrossRef]
26. Wang, X.; Ma, K.T.; Ng, G.W.; Grimson, W.E.L. Trajectory analysis and semantic region modeling using a nonparametric Bayesian model. In Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), Anchorage, AK, USA, 24–26 June 2008. [CrossRef]
27. Hu, W.; Li, X.; Tian, G.; Maybank, S.J.; Zhang, Z. An Incremental DPMM-Based Method for Trajectory Clustering, Modeling, and Retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1051–1065. [PubMed]
28. Jiang, X.; de Souza, E.N.; Pesaranghader, A.; Hu, B.; Silver, D.L.; Matwin, S. 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, Markham, ON, Canada, 6–8 November 2017; pp. 192–200.
29. Zhang, R.; Xie, P.; Wang, C.; Liu, G.; Wan, S. Classifying transportation mode and speed from trajectory data via deep multi-scale learning. *Comput. Netw.* **2019**, *162*, 106861. [CrossRef]
30. Jiang, X.; Liu, X.; de Souza, E.N.; Hu, B.; Silver, D.L.; Matwin, S. Improving point-based AIS trajectory classification with partition-wise gated recurrent units. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 4044–4051.
31. Lin, C.X.; Huang, T.W.; Guo, G.; Wong, M.D.F. MtDetector: A High-Performance Marine Traffic Detector at Stream Scale. In Proceedings of the 12th ACM International Conference on Distributed and Event-Based Systems, DEBS '18, Hamilton, New Zealand, 25–29 June 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 205–208. [CrossRef]
32. Chatzikokolakis, K.; Zissis, D.; Vodas, M.; Spiliopoulos, G.; Kontopoulos, I. A distributed lightning fast maritime anomaly detection service. In Proceedings of the OCEANS 2019, Marseille, France, 17–20 June 2019; pp. 1–8. [CrossRef]
33. Boiarov, A.; Tyantov, E. Large scale landmark recognition via deep metric learning. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Turin, Italy, 22–26 October 2019; pp. 169–178.
34. Shen, D.; Wu, G.; Suk, H.I. Deep learning in medical image analysis. *Annu. Rev. Biomed. Eng.* **2017**, *19*, 221–248. [CrossRef]
35. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
36. Makris, A.; Kontopoulos, I.; Tserpes, K. COVID-19 detection from chest X-Ray images using Deep Learning and Convolutional Neural Networks. In Proceedings of the 11th Hellenic Conference on Artificial Intelligence, Athens, Greece, 2–4 September 2020; pp. 60–66.
37. Liu, S.; Liu, S.; Cai, W.; Pujol, S.; Kikinis, R.; Feng, D. Early diagnosis of Alzheimer's disease with deep learning. In Proceedings of the 2014 IEEE 11th international symposium on biomedical imaging (ISBI), Beijing, China, 29 April–2 May 2014; pp. 1015–1018.
38. Wang, D.; Khosla, A.; Gargeya, R.; Irshad, H.; Beck, A.H. Deep learning for identifying metastatic breast cancer. *arXiv* **2016**, arXiv:1606.05718.

39. Ho, T.K.K.; Gwak, J.; Prakash, O.; Song, J.I.; Park, C.M. Utilizing Pretrained Deep Learning Models for Automated Pulmonary Tuberculosis Detection Using Chest Radiography. In Proceedings of the Asian Conference on Intelligent Information and Database Systems, Yogyakarta, Indonesia, 8–11 April 2019; pp. 395–403.
40. Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
41. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. *Commun. ACM* **2017**, *60*, 84–90. [CrossRef]
42. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
43. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9.
44. Arora, S.; Bhaskara, A.; Ge, R.; Ma, T. Provable Bounds for Learning Some Deep Representations. In Proceedings of the 31st International Conference on Machine Learning, 21–26 June 2014; Xing, E.P., Jebara, T., Eds.; PMLR: Bejing, China, 2014; Volume 32, pp. 584–592.
45. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A.A. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI'17, San Francisco, CA, USA, 4–9 February 2017; AAAI Press: Palo Alto, CA, USA, 2017; pp. 4278–4284.
46. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
47. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; pp. 1800–1807. [CrossRef]
48. Chen, R.; Chen, M.; Li, W.; Wang, J.; Yao, X. Mobility modes awareness from trajectories based on clustering and a convolutional neural network. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 208. [CrossRef]
49. Chen, X.; Kamalasudhan, A.; Zhang, X. An application of convolutional neural network to derive vessel movement patterns. In Proceedings of the 5th International Conference on Transportation Information and Safety (ICTIS), Liverpool, UK, 14–17 July 2019; pp. 939–944. .
50. Endo, Y.; Toda, H.; Nishida, K.; Ikedo, J. Classifying spatial trajectories using representation learning. *Int. J. Data Sci. Anal.* **2016**, *2*, 107–117. [CrossRef]
51. Zhang, P.; Zhao, J. The Obligations of an Anchored Vessel to Avoid Collision at Sea. *J. Navig.* **2013**, *66*, 473–477. [CrossRef]
52. Pitsikalis, M.; Kontopoulos, I.; Artikis, A.; Alevizos, E.; Delaunay, P.; Pouessel, J.; Dreo, R.; Ray, C.; Camossi, E.; Jousselme, A.; et al. Composite Event Patterns for Maritime Monitoring. In Proceedings of the 10th Hellenic Conference on Artificial Intelligence, SETN 2018, Patras, Greece, 9–12 July 2018; pp. 29:1–29:4. [CrossRef]
53. Pitsikalis, M.; Artikis, A.; Dreo, R.; Ray, C.; Camossi, E.; Jousselme, A. Composite Event Recognition for Maritime Monitoring. In Proceedings of the 13th ACM International Conference on Distributed and Event-based Systems, DEBS 2019, Darmstadt, Germany, 24–28 June 2019; pp. 163–174. [CrossRef]
54. Gaol, F.L. Bresenham Algorithm: Implementation and Analysis in Raster Shape. *J. Comput.* **2013**, *8*, 69–78. [CrossRef]
55. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [CrossRef]
56. Yamashita, R.; Nishio, M.; Do, R.K.G.; Togashi, K. Convolutional neural networks: An overview and application in radiology. *Insights Imaging* **2018**, *9*, 611–629. [CrossRef]
57. Chen, Y.; Jiang, H.; Li, C.; Jia, X.; Ghamisi, P. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 6232–6251. [CrossRef]
58. Bengio, Y. Deep Learning of Representations for Unsupervised and Transfer Learning. In Proceedings of the Unsupervised and Transfer Learning—Workshop held at ICML 2011, Bellevue, WA, USA, 2 July 2011; Volume 27, pp. 17–36.
59. Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; Darrell, T. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21–26 June 2014; Volume 32, pp. 647–655.
60. Mohanty, S.P.; Hughes, D.P.; Salathé, M. Using deep learning for image-based plant disease detection. *Front. Plant Sci.* **2016**, *7*, 1419. [CrossRef]
61. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F.-F. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; pp. 248–255.
62. Zeiler, M.D.; Fergus, R. Visualizing and Understanding Convolutional Networks. In *Computer Vision—ECCV 2014—13th European Conference, Zurich, Switzerland, 6–12 September 2014*; Part I; Fleet, D.J., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8689, pp. 818–833. [CrossRef]
63. Lin, M.; Chen, Q.; Yan, S. Network in network. *arXiv* **2013**, arXiv:1312.4400.
64. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
65. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv* **2012**, arXiv:1207.0580.
66. Hawkins, D.M. The problem of overfitting. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 1–12. [CrossRef]

67. Perez, L.; Wang, J. The effectiveness of data augmentation in image classification using deep learning. *arXiv* **2017**, arXiv:1712.04621.
68. Makris, A.; Tserpes, K.; Anagnostopoulos, D. A novel object placement protocol for minimizing the average response time of get operations in distributed key-value stores. In Proceedings of the 2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, 11–14 December 2017; pp. 3196–3205. [CrossRef]
69. Makris, A.; Tserpes, K.; Anagnostopoulos, D.; Altmann, J. Load balancing for minimizing the average response time of get operations in distributed key-value stores. In Proceedings of the 2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC), Boston, MA, USA, 11–14 December 2017; pp. 263–269.
70. Makris, A.; Tserpes, K.; Spiliopoulos, G.; Anagnostopoulos, D. Performance Evaluation of MongoDB and PostgreSQL for Spatio-temporal Data. In Proceedings of the Workshops of the EDBT/ICDT 2019 Joint Conference, EDBT/ICDT 2019, Lisbon, Portugal, 26 March 2019; Volume 2322.
71. Makris, A.; Tserpes, K.; Spiliopoulos, G.; Zissis, D.; Anagnostopoulos, D. MongoDB Vs PostgreSQL: a comparative study on performance aspects. *GeoInformatica* **2021**, *25*, 241–242. [CrossRef]
72. Makris, A.; Tserpes, K.; Anagnostopoulos, D.; Nikolaidou, M.; de Macedo, J.A.F. Database system comparison based on spatiotemporal functionality. In Proceedings of the 23rd International Database Applications & Engineering Symposium, Athens, Greece, 10–12 June 2019; pp. 1–7.
73. Meratnia, N.; de By, R.A. A new perspective on trajectory compression techniques. In Proceedings of the ISPRS Commission II and IV, WG II/5, II/6, IV/1 and IV/2 Joint Workshop Spatial, Temporal and Multi-Dimensional Data Modelling and Analysis, Quebec city, QC, Canada, 2–3 October 2003.
74. Leichsenring, Y.E.; Baldo, F. An evaluation of compression algorithms applied to moving object trajectories. *Int. J. Geogr. Inf. Sci.* **2020**, *34*, 539–558. [CrossRef]
75. Muckell, J.; Hwang, J.; Lawson, C.T.; Ravi, S.S. Algorithms for compressing GPS trajectory data: An empirical evaluation. In Proceedings of the 18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2010, San Jose, CA, USA, 3–5 November 2010; pp. 402–405. [CrossRef]
76. Potamias, M.; Patroumpas, K.; Sellis, T. Sampling trajectory streams with spatiotemporal criteria. In Proceedings of the 18th International Conference on Scientific and Statistical Database Management, Vienna, Austria, 3–5 July 2006; pp. 275–284.
77. Muckell, J.; Hwang, J.; Patil, V.; Lawson, C.T.; Ping, F.; Ravi, S.S. SQUISH: an online approach for GPS trajectory compression. In Proceedings of the 2nd International Conference and Exhibition on Computing for Geospatial Research & Application, Washington, DC, USA, 23–25 May 2011; pp. 13:1–13:8. [CrossRef]
78. Trajcevski, G.; Cao, H.; Scheuermann, P.; Wolfson, O.; Vaccaro, D. On-line data reduction and the quality of history in moving objects databases. In Proceedings of the Fifth ACM International Workshop on Data Engineering for Wireless and Mobile Access, Mobide 2006, Chicago, IL, USA, 25 June 2006; pp. 19–26. [CrossRef]
79. Dritsas, E.; Kanavos, A.; Trigka, M.; Sioutas, S.; Tsakalidis, A. Storage efficient trajectory clustering and k-nn for robust privacy preserving spatio-temporal databases. *Algorithms* **2019**, *12*, 266. [CrossRef]
80. Meratnia, N.; de By, R.A. Spatiotemporal Compression Techniques for Moving Point Objects. In *Advances in Database Technology—EDBT 2004, 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, 14–18 March 2004*; Bertino, E., Christodoulakis, S., Plexousakis, D., Christophides, V., Koubarakis, M., Böhm, K., Ferrari, E., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; Volume 2992, pp. 765–782. [CrossRef]
81. Douglas, D.H.; Peucker, T.K. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. *Cartographica* **1973**, *10*, 112–122. [CrossRef]
82. Kontopoulos, I.; Vodas, M.; Spiliopoulos, G.; Tserpes, K.; Zissis, D. Single Ground Based AIS Receiver Vessel Tracking Dataset. 2020. Available online: https://zenodo.org/record/3754481#.YG7dGD8RWbg (accessed on 1 April 2021).
83. Sayood, K. *Introduction to Data Compression*, 3rd ed.; The Morgan Kaufmann Series in Multimedia Information and Systems; Elsevier: Amsterdam, The Netherlands, 2006.
84. Muckell, J.; Olsen, P.W.; Hwang, J.H.; Lawson, C.T.; Ravi, S. Compression of trajectory data: a comprehensive evaluation and new approach. *GeoInformatica* **2014**, *18*, 435–460. [CrossRef]
85. Liu, G.; Iwai, M.; Sezaki, K. A method for online trajectory simplification by enclosed area metric. In Proceedings of the Sixth International Conference on Mobile Computing and Ubiquitous Networking, Okinawa, Japan, 23–25 May 2012.