LSTM based Anomaly Detection in Time Series for United States exports and imports

Aggarwal, Sakshi

Indian Institute of Foreign Trade

25 April 2023

LSTM based Anomaly Detection in Time Series for United States exports and imports

Sakshi Aggarwal
Indian Institute of Foreign Trade
Email – sakshi230391@gmail.com

Abstract

This survey aims to offer a thorough and organized overview of research on anomaly detection, which is a significant problem that has been studied in various fields and application areas. Some anomaly detection techniques have been tailored for specific domains, while others are more general. Anomaly detection involves identifying unusual patterns or events in a dataset, which is important for a wide range of applications including fraud detection and medical diagnosis. Not much research on anomaly detection techniques has been conducted in the field of economic and international trade. Therefore, this study attempts to analyze the time-series data of United Nations exports and imports for the period 1992 – 2022 using LSTM based anomaly detection algorithm. Deep learning, particularly LSTM networks, are becoming increasingly popular in anomaly detection tasks due to their ability to learn complex patterns in sequential data. This paper presents a detailed explanation of LSTM architecture, including the role of input, forget, and output gates in processing input vectors and hidden states at each timestep. The LSTM based anomaly detection approach yields promising results by modelling small-term as well as long-term temporal dependencies.

Keywords Anomaly detection, LSTM, Machine learning, Artificial intelligence, economic trade

Introduction

Anomaly detection is a technique used to identify unusual patterns or events in a dataset that do not conform to the expected behavior or typical values. Anomaly detection is interesting because it involves automatically discovering interesting and rare patterns from datasets (Ahmed et al., 2014). Anomaly detection has been widely studied in statistics and machine learning. Anomalies are also referred to as outliers, novelties, or deviations. Anomalies are important because they indicate significant but rare events, and they can prompt critical actions to be taken in a wide range of application domains, including fraud detection in financial transactions, intrusion detection in network security, fault detection in manufacturing processes, and medical diagnosis in healthcare (Bolton et al., 2001; Lin et al., 2005; Chandola et al., 2009; Chalapathy and Chawla, 2019). In the broader field of machine learning, recent years have witnessed a proliferation of deep neural networks, with unprecedented results in diverse application domains. Deep learning is a subset of machine learning that achieves good performance and flexibility by learning to represent the data as a nested hierarchy of concepts within layers of the neural network. (Peng and Marculescu, 2015; Javaid et al., 2016). Deep learning outperforms traditional machine learning as the scale of data increases as illustrated in Fig. 1. The recent trends in machine learning have witnessed that deep learning-based anomaly detection algorithms are becoming increasingly popular in a diverse set of tasks. Fig. 2 provides a glimpse that deep learning completely surpasses traditional methods in various domains.

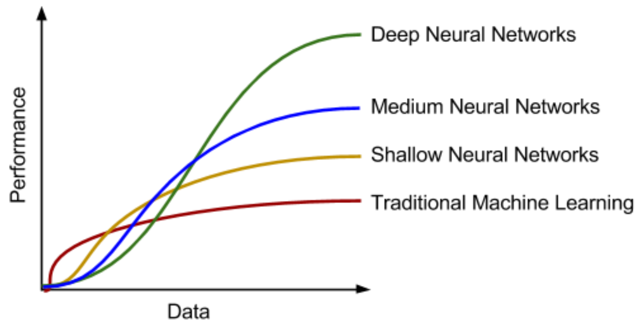

Fig. 1. Performance comparison of deep learning-based algorithms vs traditional algorithms.

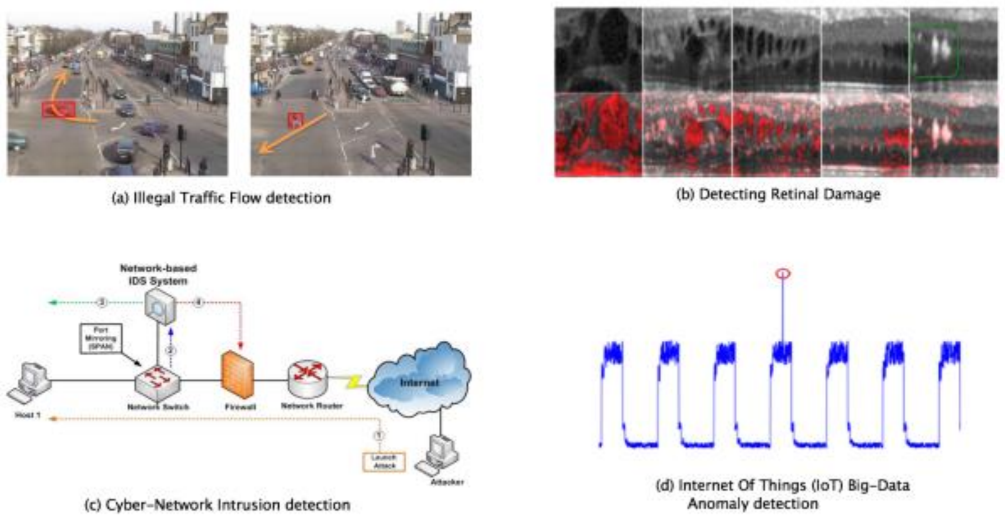
(a) Illegal Traffic Flow detection
(b) Detecting Retinal Damage
Network-based IDS System
Port Mirroring (SPAN)
Host 1
Network Switch
Firewall
Network Router
Internet
Launch Attack
Attacker
(c) Cyber-Network Intrusion detection
(d) Internet Of Things (IoT) Big-Data Anomaly detection

Fig. 2. Applications Deep learning-based anomaly detection algorithms.

In this paper, we focus on anomaly detection research in the international trade domain. Many sub-fields of economics are moving towards the use of AI in behavioral modeling, e.g., (Aggarwal and Chakraborty, 2017, 2019, 2020a, 2020b, 2020c, Ghoddusi et al., 2019) for energy and international economics. However, there is limited work in the literature that shows tailored and optimized neural network algorithms for economic applications. Neural networks offer the ability to find connections between variables that are difficult to detect using traditional machine learning methods. Even compared to other machine learning approaches, neural networks can provide a better fit in the long term as found in (Gopinath et al., 2020; Monken et al., 2021) for international trade. The 2020 U.S. National Artificial Intelligence Initiative Bill states that "Artificial intelligence is a tool that has the potential to change and possibly transform every sector of the U.S. economy" (S. 1558 – AIIA 2020). Given its obvious influence on sectoral growth, the next sector for AI to advance is international trade, and its causal effects on production, consumption, and prices.

What are anomalies?

Many authors have proposed varying definitions for an anomaly in their research; however, there has not been a universally adopted definition. Exact definitions of an anomaly depend on the problem statement, assumptions regarding the structure of the data and the application in consideration. However, definitions exist that are considered general to most if not all cases, regardless of the setting or application. Of those definitions, the most widely recognized is by Hawkins, who defines the concept of an anomaly or an outlier in this case, as follows: "*An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism*" (Hawkins, 1980). This definition refers to data from a statistics-based intuition, where normal data follows a generating mechanism and anomalies are samples or instances which deviate from this mechanism. Thus, anomalies often relay useful information about a system's abnormal characteristics that are impacting the generating mechanism (Aggarwal, 2013). Fig. 3 illustrates anomalies in a simple 2-dimensional dataset. The data has two normal regions, N_1 and N_2 since most observations lie in these two regions, whereas the region O_3, and the data points O_1 and O_2 are few data points which are located further away from the bulk of data points and hence are considered anomalies.

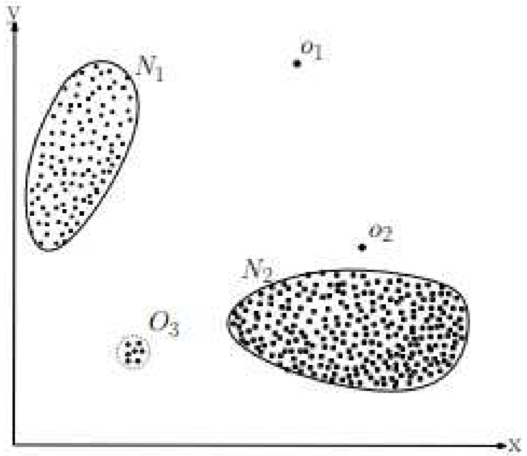

Fig. 3. Graphical visualization of anomalous data in a two-dimensional representation.

Anomalies might be induced in the data for a variety of reasons, such as malicious activity or breakdown of a system, with the common characteristic that these reasons are of interest to the analyst. Such activities include credit card fraud, cyber-intrusion, terrorist activity, breach of computer network, etc.

Types of anomalies

Anomalies can be classified into different categories, and these refer to various types of unexpected or unusual events or data points that do not conform to a well-defined characteristic of normal behavior. The type of anomaly being dealt with is a crucial aspect of consideration for any anomaly detection technique. The three main categories of anomalies are discussed below.

- **Point anomalies**: These are individual data points that are significantly different from the rest of the data in a dataset. For instance, the monthly expenditure on grocery shopping can be considered. If the usual expenditure on grocery shopping per month is USD 500 but if it becomes 5001 in any random month then it is a point anomaly.
- **Contextual anomalies**: These are data points that are not anomalous on their own but are considered unusual when considering the context or conditions in which they occur. For instance, high expenditure on credit cards during a festive period, e.g., Christmas or New Year, is usually higher than the rest of the year. Although the expenditure during a festive month can be considered high, it may not be anomalous due to the high expenses being contextually normal at that time.
- **Collective anomalies**: These are groups of data points that are anomalous when analyzed together but may not be anomalous when analyzed individually. For example, in a human Electrocardiogram (ECG) output, the existence of low values for a long period of time indicates underlying phenomenon corresponding to abnormal pre-mature contraction (Keogh et al., 2005), however, one low value by itself is not considered as anomalous.

The few other categories of anomalies are as follows:

- **Temporal anomalies**: These are data points that exhibit anomalous behavior over time, such as sudden spikes or drops in value. A common example of a temporal anomaly is credit card fraud detection. Credit card transactions typically occur at regular intervals and have a consistent pattern of behavior. Any transaction that deviates significantly from this pattern may be considered anomalous and indicative of fraud. For example, if a customer who typically makes small purchases suddenly starts making large transactions at unusual times, it could be flagged as a temporal anomaly.
- **Spatial anomalies**: These are data points that exhibit anomalous behavior in terms of their location or spatial distribution. A common example of a spatial anomaly is detecting unusual patterns in the distribution of disease outbreaks. Epidemiologists can use spatial analysis techniques to identify clusters of cases that occur near each other, indicating a potential outbreak. Any area that has a significantly higher incidence of a disease compared to its neighboring regions may be considered a spatial anomaly.
- **Rare events**: These are events that are highly unusual or unexpected, such as natural disasters or cyber-attacks.
- **Noise anomalies**: These are data points that are not actually anomalous but appear to be due to measurement errors or random fluctuations in the data. A common example of a noise anomaly is in sensor data. Sensors can sometimes produce inaccurate or random measurements due to a variety of factors, such as environmental conditions, electrical interference, or malfunctioning equipment. These measurements may not actually reflect any meaningful signal, but they can still be captured and recorded as part of the dataset, leading to false conclusions if not handled correctly. For example, if a temperature sensor in a room suddenly produces a reading that is significantly higher or lower than the surrounding readings, it could be flagged as a noise anomaly.

Anomaly detection is related to but distinct from noise removal (Chen et al., 1990) and noise accommodation (Rousseeuw and Leroy, 1987), both of which deal with unwanted noise in the data. In real-world applications, the data may be affected by a significant amount of noise, which may not be of interest to the analyst, but acts as a hindrance during the data analysis stage. It is usually only significantly interesting deviations that are of interest (Aggarwal, 2013). The detrimental effect of noise on data analysis drives the need for noise removal, as it removes any unwanted objects before data analysis (Xiong et al., 2006). Another topic related to anomaly detection is *novelty detection* (Markou and Singh 2003a; 2003b; Saunders and Gero 2000) which aims at detecting previously unobserved novel patterns in the data, e.g., a new topic of discussion in a news group. The distinction between novel patterns and anomalies is that the novel patterns are typically incorporated into the normal model after being detected.

What are novelties?

Novelty detection is the identification of a novel or unobserved patterns in the data. The novelties detected are not considered as anomalous data points; instead, the data points are applied to the regular data model (Miljković, 2010). A novelty score may be assigned for these previously unseen data points, using a decision threshold score (Pimentel et al., 2014). The points which significantly deviate from this decision threshold may be considered as anomalies or outliers. For instance, in Fig. 4 the images of white tigers among regular tigers may be considered as a novelty while the image of horse, panther, lion and cheetah are considered as anomalies. The techniques used for anomaly detection are often used for novelty detection and vice versa.

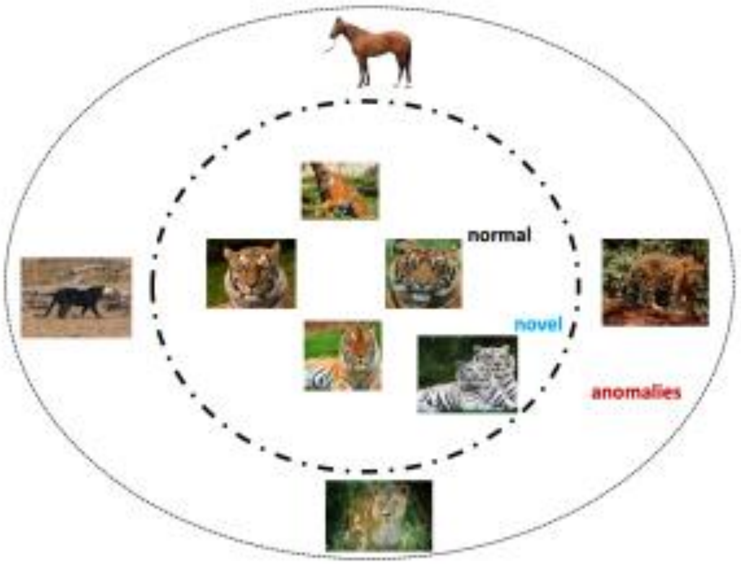

Fig. 4. Illustration of novelty in the image dataset.

Literature Survey

A substantial amount of research on outlier detection has already been conducted in statistics in the previous century (Rousseeuw and Leroy, 1987; Barnett and Lewis, 1994; Hawkins, 1980; Beckman and Cook, 1983). An extensive review of novelty detection techniques using deep learning framework and advanced statistical methodology has been presented in Markou and Singh (2003a, 2003b). A plethora of published research literature has been conducted by applying anomaly detection techniques in various applications in different domains. Anomaly detection has emerged as the topic of focus for many surveys and review papers in recent years. An extensive survey of anomaly detection techniques developed in machine learning and statistics has been provided by (Hodge and Austin, 2004; Nguyen and Armitage, 2008). These surveys provide extensive background on outliers or anomalies and the challenges associated with detecting them, thereby marked a significant impact on further research in various fields. In 2003, Noble and Cook detected unusual patterns within graph-based data using the concept of 'conditional entropy' (Noble and Cook, 2003). In 2009, Chandola et al. also surveyed different aspects of anomaly detection techniques that has been proposed in research but not covered in the literature of Hodge and Austin, provided more meaningful insight into the real-world applications they are employed in (Chandola et al., 2009).

In 2012, a survey Zimek et al. published research for high-dimensional numerical data and reviewed unsupervised anomaly detection techniques specifically. The focus of the research was to discuss the aspects of the 'curse of dimensionality' in detail (Zimek et al., 2012). The analysis involved in this research was comparison of two categorizes of specialized algorithms: ones that address the presence of irrelevant feature or attributes in the model and others that are more concerned with efficiency and effectiveness of the model. Temporal data poses another issue for detecting anomalies in the dataset. With the advances in computational capabilities enabling the availability of various forms of temporal data, the need to extensively review the anomaly detection techniques in time-series data has arisen. In this respect, Gupta et al. in 2013 conducted research and provided significant insight into various applications of temporal anomaly detection and the associated challenges in each domain.

Other exhaustive survey papers exist that focus more specifically on the techniques and applications of anomaly detection in several domains includes (Patcha and Park, 2007; Garcia-Teodoro, 2009; Callado et al., 2009; Zhang et al., 2009; Sperotto et al., 2010). Patcha and Park (2007) presented surveys of anomaly detection techniques used specifically for cyber intrusion detection. Callado et al. (2009) reported major techniques and problems identified in IP traffic analysis, with an emphasis on application detection. Zhang et al. (2009) presented a survey on anomaly detection methods in network intrusion detection. A review of flow-based intrusion detection was presented in Sperotto et al. (2010), who elaborates on the concepts of flow and classified attacks and provided a detailed discussion of detection techniques for DoS attacks. Other literature surveys have been reported in the context of wireless networks (Sun et al., 2006; Sun et al., 2007a, Sun et al., 2007b; Sun et al., 2007c). Sun et al. (2007a) presented a survey of intrusion detection techniques for mobile ad-hoc networks (MANET) and wireless sensor networks (WSN). Their analysis also highlighted the several important research issues and challenges in the context of building IDSs by integrating aspects of mobility. A systematic literature review of different graph-based anomaly detection techniques that have been studied in published literature (Pourhabibi et al., 2020).

Not much extensive research on anomaly detection techniques has been conducted in the field of economic and international trade. Understanding, modeling, and predicting international trade is a central domain in economics. Economists have implemented the Gravity equation of trade to describe trade flows (Anderson 1979, Aggarwal et al., 2022; Aggarwal et al., 2023). More recently, researchers across multidisciplinary fields have identified international trade as well suited for machine learning and network analysis (Bhattacharya et al., 2007; Aggarwal 2016, 2017a, 2017b; Aggarwal et al., 2021). By harnessing machine learning methods, researchers have been able to better predict agricultural exports trade patterns using Gradient Tree Boosting, ARIMA, and XGBoosting (Batarseh et al., 2019; Aggarwal and Chakraborty, 2021, 2022). Use of shallow neural networks also improves international trade forecasting (Wohl and Kennedy 2018). Combining neural networks with network analysis theory of international trade, (Panford-Quainoo et al., 2020; Monken et al., 2021) demonstrates the efficacy of country GDP classification using a

Graph Neural Networks (GNNs) for a single period. Our contribution to the literature is to detect anomalies in novel ways for United States trade data over the past two decades and influencing public policy for the greater good.

Deep learning anomaly detection techniques

Deep learning anomaly detection techniques have gained prominence in recent years, demonstrating significantly better performance than other techniques in addressing real-world problems. These include neural networks (NN) architectures of various types, such as convolutional neural networks (CNN), long short-term memory networks (LSTM), autoencoders (AE) and generative adversarial networks (GANs). (Agarwal et al., 2017; Fiore et al., 2019; Wang et al., 2020). Neural networks have been successfully applied to a wide range of tasks, including image and speech recognition, natural language processing, and predictive analytics. Convolutional neural networks are commonly used for image recognition tasks. LSTM networks are well-suited for tasks that involve processing sequential data, such as speech recognition, natural language processing, and time series prediction. Autoencoders have many applications, including image and video compression, anomaly detection, and feature extraction for downstream tasks such as classification or clustering. They can also be used for generative modeling by training the decoder network to generate new data points by sampling from the compressed representation. Generative adversarial networks are used for generating realistic synthetic data, such as images and videos. This section reviews the neural network algorithms and its different types that have been applied in the published literature.

(i) Neural Networks (NN)

A neural network is a type of machine learning model that is designed to mimic the structure and function of the human brain. In this section, we are primarily concerned with feedforward neural networks, also known as multilayered perceptron networks (MLP), which are simplest type of neural networks. It is made up of interconnected nodes, or neurons, which are organized into layers.

The basic building block of a neural network is the neuron. Each neuron receives one or more inputs, which are multiplied by weights that represent the strength of the connection between the neuron and its inputs. The neuron then applies an activation function to the weighted sum of the inputs, producing an output. The activation function is a non-linear function that is applied to the weighted sum of the inputs. It is used to introduce non-linearity into the network and enable it to learn complex relationships between inputs and outputs. Common activation functions include sigmoid, tanh, and ReLU (rectified linear unit). A neural network typically consists of an input layer, one or more hidden layers, and an output layer. The input layer receives the input data, which is then passed through the hidden layers before reaching the output layer (Michelucci, 2018). A graphical illustration of a densely connected NN can be seen in Fig. 5. Each hidden layer consists of multiple neurons, each of which applies an activation function to the weighted sum of its inputs. During training, the weights of the connections between neurons are adjusted in order to minimize the difference between the network's predicted outputs and the true outputs. This process is executed by feeding the network a set of training examples, each with a known input and output, and adjusting the weights based on the difference between the predicted and true outputs.

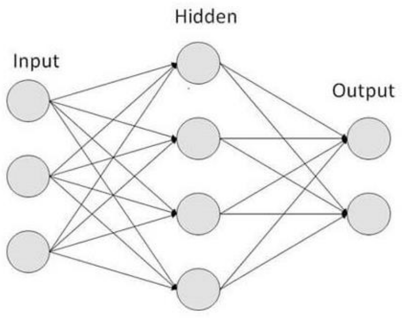

Fig. 5. Labelled visual representation of a densely connected multilayered perceptron.

Neural network algorithm

The algorithm of a neural network can be broken down into several steps, including:

1. Initialization: Initialize the weights of the connections between the neurons in the network to small random values.
2. Forward Propagation: Feed the input data into the input layer of the neural network. The data then flows through the network, layer by layer, until it reaches the output layer. At each layer, the weighted sum of the inputs is calculated, and an activation function is applied to produce the output of each neuron.
3. Loss Calculation: Compare the output of the neural network to the actual output and calculate the loss or error. This is typically done using a loss function such as mean squared error (MSE) or cross-entropy loss.
4. Backpropagation: Propagate the error backwards through the network, from the output layer to the input layer. The error is used to adjust the weights of the connections between neurons, with the goal of minimizing the loss.
5. Update Weights: Use an optimization algorithm, such as stochastic gradient descent (SGD) or Adam, to update the weights of the connections between neurons based on the error calculated during backpropagation.
6. Repeat: Repeat steps 2-5 for multiple iterations or epochs, until the network has learned to produce accurate outputs for the given inputs.
7. Prediction: Use the trained neural network to predict the output for new inputs.

This process is typically repeated multiple times with different training examples until the network can accurately predict outputs for unseen data. Additionally, hyperparameters such as the learning rate, number of layers, and number of neurons in each layer can be tuned to improve the performance of the neural network

(ii) Convolutional Neural Networks (CNN)

CNN is another type of deep learning architecture that was first introduced in the 1990s, which is characterized as a network with many layers categorized into the input, pooling, fully connected, and output layers (LeCun and Bengio, 1995). They are named after the fact that the convolutional layer performs a mathematical operation known as convolution on the input. CNN (LeCun et al., 1998) is a subclass of neural networks that takes advantage of the spatial structure of the inputs. CNN models have a standard structure consisting of alternating convolutional layers and pooling layers (often each pooling layer is placed after a convolutional layer). Fig. 6 displays a representation of the structure of a CNN's layers. CNNs are usually trained by backpropagation via Stochastic Gradient Decent (SGD) to find weights and biases that minimize certain loss function in order to map the arbitrary inputs to the targeted outputs as closely as possible. CNNs have been predominantly used for image-driven pattern recognition tasks due to the essential nature of the input, which is usually in matrix form (O'Shea and Nash, 2015). However, they have been demonstrated to be applicable in other fields and domains by manipulating the structure of the input data.

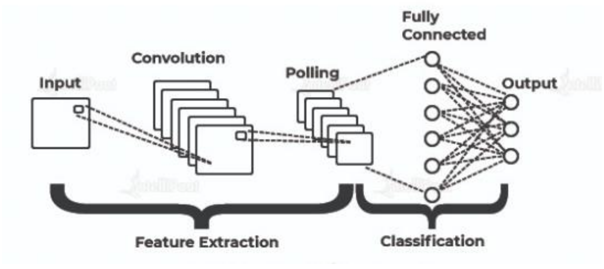

Fig. 6. Schematic representation of the layers of a CNN and the associated functions.

The various layers of CNN's architecture are responsible for carrying out different operations. As briefly mentioned, the convolution operation is performed in the convolutional layer, which preserves the spatial relationship of the input to extract derived features. Different filters are used for the convolution operation, acting as feature detectors. The pooling layer reduces the dimensionality of the feature maps produced from the convolutional layer using subsampling techniques, retaining the most critical information, and makes the network more robust to small variations in the input image. Pooling allows for the input to be more manageable, reducing the number of computations and parameters in the network, which helps to control overfitting (Krizhevsky et al., 2012; Albelwi and Mahmood, 2017). The fully connected and output layer is a traditional densely connected feedforward NN or MLP whose input is the output from the convolutional and pooling layer and serves to classify the data.

Convolutional neural network algorithm

The algorithm of a convolutional neural network (CNN) can be broken down into several steps, including:

1. Convolution: The input image is convolved with a set of learnable filters, also known as kernels or weights. Each filter is moved across the image, and the dot product of the filter and the image pixels within the filter window is calculated at each position. This produces a feature map that highlights certain patterns or features in the image.
2. Activation: An activation function such as ReLU (rectified linear unit) is applied to the output of each convolution operation. This introduces non-linearity into the network and enables it to learn complex features.
3. Pooling: A pooling operation is applied to the output of the activation function. This down samples the feature map by taking the maximum or average value within a small window. This reduces the dimensionality of the feature map and makes the network more computationally efficient.
4. Repeat: Steps 1-3 are repeated multiple times, with the output of one layer becoming the input to the next layer. This allows the network to learn increasingly complex features at higher levels of abstraction.
5. Fully Connected Layers: The output of the last convolutional layer is flattened and passed through one or more fully-connected layers, also known as dense layers. These layers perform a weighted sum of the inputs, followed by an activation function, to produce the final output of the network.

6. Loss Calculation: Compare the output of the neural network to the actual output and calculate the loss or error. This is typically done using a loss function such as cross-entropy loss.
7. Backpropagation and Weight Update: Propagate the error backwards through the network using backpropagation and update the weights of the connections between neurons using an optimization algorithm such as stochastic gradient descent (SGD) or Adam.
8. Repeat: Repeat steps 1-7 for multiple iterations or epochs, until the network has learned to produce accurate outputs for the given inputs.
9. Prediction: Use the trained CNN to predict the output for new inputs.

This process is typically repeated multiple times with different training examples until CNN can accurately predict outputs for unseen data. Additionally, hyperparameters such as the size of the filters, the number of filters in each layer, and the size of the pooling windows can be tuned to improve the performance of the CNN.

(iii) Long Short-Term Memory Networks (LSTM)

Long Short-Term Memory networks (LSTM) are an extension of recurrent neural networks (RNN), a form of deep neural network primarily used for time series data proposed by Hochreiter and Schmidhuber in 1997 (Hochreiter and Schmidhuber, 1997). Each neuron in an LSTM is a cell with ‘memory’ that can store information, maintaining its own state, in contrast to RNNs that merely take the current input from their previous hidden state to output a new hidden state. LSTM is designed to overcome the vanishing gradient problem in traditional RNNs, which occurs when gradients become too small during backpropagation and cause the network to forget long-term dependencies. LSTM is used for processing sequential data such as time series, natural language text, or speech. (Gers et al., 2000; Yu et al., 2019)

LSTM is composed of several memory cells, each containing three gates: input, forget, and output gates. The input gate controls the amount of new information that enters the cell, while the forget gate controls the amount of old information that is retained in the cell. The output gate controls the amount of information that is outputted from the cell. During each timestep, the LSTM receives an input vector and a hidden state vector from the previous timestep. The input vector is first multiplied by a weight matrix and added to the previous hidden state vector. The resulting vector is then fed through the input gate, which applies a sigmoid activation function to the weighted sum of the input and the previous hidden state. The output of the input gate is then multiplied by a candidate vector that contains new information, which is passed through a hyperbolic tangent function to normalize the values between -1 and 1. The forget gate is then applied to the previous hidden state vector and the input vector. It also applies a sigmoid activation function to the weighted sum of the previous hidden state and the input vector. The output of the forget gate is then multiplied by the previous cell state vector, which results in the removal of unnecessary information from the cell. The output gate is then applied to the current input vector and the current hidden state vector. It applies a sigmoid activation function to the weighted sum of the input vector and the current hidden state. The output of the output gate is then multiplied by the hyperbolic tangent of the current cell state vector, resulting in the output of the LSTM cell. The LSTM architecture can be stacked to form multiple layers, which allows the network to learn more complex patterns in the input sequence. The output of the final LSTM cell is typically fed through a dense layer with a softmax activation function to predict the next element in the sequence.

Long Short-Term Memory Networks algorithm

The algorithm of Long Short-Term Memory (LSTM) networks can be broken down into several steps, including:

1. Input Processing: The input sequence is fed into the LSTM network, with each element of the sequence being processed one at a time.
2. Forget Gate: The LSTM network decides which information from the previous time step to forget, by using a sigmoid activation function to decide which values to keep or discard.
3. Input Gate: The LSTM network decides which new information to add to the memory cell, by using a sigmoid activation function and a tanh activation function to compute the values to add.
4. Update Memory Cell: The LSTM network updates the memory cell with the result of the forget and input gates.
5. Output Gate: The LSTM network decides which information to output, by using a sigmoid activation function and a tanh activation function to compute the output value.
6. Output: The LSTM network produces the output value and passes it on to the next time step.
7. Repeat: Steps 2-6 are repeated for each element of the input sequence.
8. Loss Calculation: Compare the output of the LSTM network to the actual output, and calculate the loss or error. This is typically done using a loss function such as mean squared error (MSE) or binary cross-entropy.
9. Backpropagation and Weight Update: Propagate the error backwards through the network using backpropagation, and update the weights of the connections between neurons using an optimization algorithm such as stochastic gradient descent (SGD) or Adam.
10. Repeat: Repeat steps 1-9 for multiple iterations or epochs, until the network has learned to predict outputs accurately.

(iv) Autoencoders (AE)

An autoencoder (AE) is a type of unsupervised deep learning network symmetric in structure with fewer nodes in the middle layers. It has a section that encodes inputs into a lower-dimensional representation and another section that decodes or reconstructs that input again (Boukerche et al., 2020).

The basic idea behind an autoencoder is to learn a compressed representation of the input data by reducing its dimensionality and then reconstructing the original data from this compressed representation. Autoencoders are composed of two main components: an encoder network that compresses the input data into a lower-dimensional representation, and a decoder network that reconstructs the original data from this representation. The encoder network takes the input data and applies a series of transformations to it to reduce its dimensionality. This is typically done using a series of fully connected layers or convolutional layers with non-linear activation functions, such as ReLU or sigmoid. As illustrated in Fig. 7, the input layer passes the input data to the hidden layer to the output layer, where it is decoded and reconstructed as much as possible. The number of hidden layers in an AE is arbitrary, with the condition that for each part of the network, e.g., the encoder, each subsequent hidden layer must have fewer neurons than the previous layer. The output of the encoder network is a compressed representation of the input data that captures its most important features. The decoder network takes this compressed representation and reconstructs the original data from it. Like the encoder network, the decoder network is typically composed of fully connected or convolutional layers with non-linear activation functions. The output of the decoder network is a reconstruction of the original input data that has been compressed and then decompressed. During training, the autoencoder tries to minimize the difference between the original input data and the reconstructed output data. This is done by optimizing a loss function such as mean squared error or binary cross-entropy. The parameters of the encoder and decoder networks are adjusted using backpropagation to minimize this loss function (Chalapathy and Chawla, 2019; Kucharski et al., 2020).

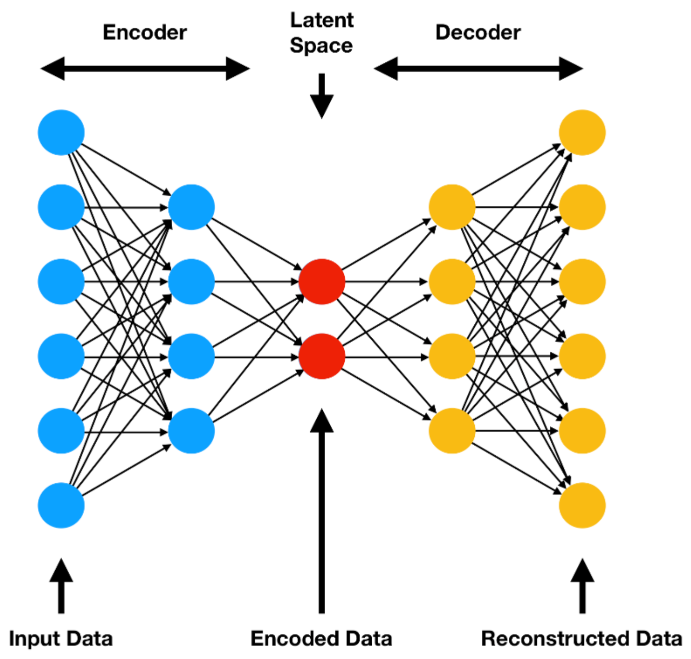

Fig. 7. Schematic of an autoencoder network's basic, symmetric architecture.

Autoencoders algorithm

The algorithm of an autoencoder in a neural network can be broken down into several steps, including:

1. Encoding: The input data is fed into the encoder network, which compresses the data into a lower-dimensional representation. This is typically done using one or more fully-connected layers with a non-linear activation function such as ReLU.
2. Decoding: The encoded data is then fed into the decoder network, which reconstructs the original input data. This is typically done using one or more fully-connected layers with a non-linear activation function such as ReLU.
3. Loss Calculation: Compare the output of the autoencoder to the actual input data, and calculate the loss or error. This is typically done using a loss function such as mean squared error (MSE) or binary cross-entropy.
4. Backpropagation and Weight Update: Propagate the error backwards through the network using backpropagation and update the weights of the connections between neurons using an optimization algorithm such as stochastic gradient descent (SGD) or Adam.
5. Repeat: Repeat steps 1-4 for multiple iterations or epochs, until the network has learned to reconstruct the input data accurately.
6. Prediction: Use the trained autoencoder to reconstruct the input data or to encode new data into a lower-dimensional representation.

The objective of an autoencoder is to learn a compressed representation of the input data that captures its most important features. The compressed representation can then be used for tasks such as data compression, data visualization, or feature extraction. Autoencoders can be further specialized for specific tasks, such as denoising autoencoders, variational autoencoders, or convolutional autoencoders, depending on the nature of the input data and the desired output.

(v) Generative Adversarial Networks (GAN)

Generative Adversarial Networks (GANs) is proposed by Goodfellow et al. in 2014, which uses a minimax game to train the generation model from the game theory perspective. These are a type of deep learning model that consists of two neural networks: a generator and a discriminator. The generator is responsible for generating new data that is similar to the training data, while the discriminator tries to distinguish between the generated data and the real data. Fig. 8 illustrates the structure of GAN, which includes two networks; one is the generator *G*. The goal of *G* is to transform the noise variable z into the generated sample ***G(z),*** which learns the distribution of real data *x*. The other is discriminator *D*, whose goal is to distinguish whether a sample is real or generated. Both *G* and *D* implement non-linear mapping by using network structures, such as multi-layer perceptron (Bengio et al., 2013; Goodfellow et al., 2014).

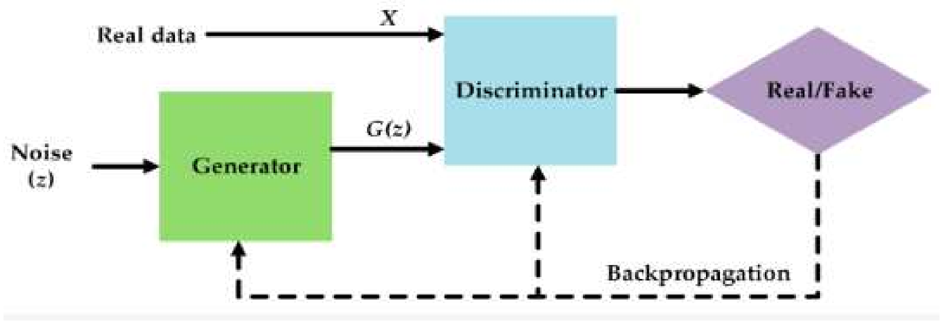

Fig. 8. Schematic of a Generative Adversarial Network (GAN) model

The basic idea behind GANs is to have the generator learn from the feedback provided by the discriminator. The generator starts by producing random noise as input, and then generates a sample of data that is similar to the training data. The discriminator is then fed both the generated sample and a sample of real data, and it is trained to classify which sample is real and which is fake. The generator is then updated based on the feedback provided by the discriminator, with the goal of generating data that is increasingly difficult for the discriminator to distinguish from the real data (Mirza and Osindero, 2014).

Generative Adversarial Networks algorithm

The algorithm of Generative Adversarial Networks (GANs) can be broken down into several steps, including:

1. Generator Network: The generator network is trained to generate synthetic data from random noise input, typically using one or more fully-connected layers and a non-linear activation function such as ReLU or tanh.
2. Discriminator Network: The discriminator network is trained to distinguish between real data and synthetic data generated by the generator network, typically using one or more fully-connected layers and a sigmoid activation function.
3. Adversarial Training: The generator and discriminator networks are trained together in an adversarial process, where the generator network tries to generate synthetic data that can fool the discriminator network, and the discriminator network tries to accurately distinguish between real and synthetic data.
4. Loss Calculation: Compare the output of the discriminator network to the actual labels (real or fake), and calculate the loss or error. This is typically done using a loss function such as binary cross-entropy.
5. Backpropagation and Weight Update: Propagate the error backwards through the discriminator network using backpropagation, and update the weights of the connections between neurons using an optimization algorithm such as stochastic gradient descent (SGD) or Adam. Then,

propagate the error backwards through the generator network using backpropagation, and update the weights of the connections between neurons using the same optimization algorithm.

6. Repeat: Steps 3-5 are repeated for multiple iterations or epochs, until the generator network has learned to generate synthetic data that can fool the discriminator network.
7. Prediction: Use the trained generator network to generate new synthetic data.

The objective of GANs is to learn a probability distribution over the input data, so that the generator network can generate synthetic data that is similar to the real data. GANs can be used for a variety of tasks, such as image and video generation, data augmentation, and style transfer. However, training GANs can be challenging due to the instability of the adversarial training process, and several techniques such as batch normalization, gradient penalty, and Wasserstein distance have been proposed to improve the stability and convergence of GANs.

The recent research work that has deployed neural network algorithms to evolve into published literature are summarized in Table 1.

Table 1. Summary of published literature using different types of Neural Network algorithms

Year	**Reference**	**Method**	**Comments**
1994	Ghosh and Reilly	MLP	MLP resulted in 20 to 40 percent decrease in economic losses.
1997	He et al.	MLP	MLP had poor accuracy on its own, improved the results when clustering implemented prior to training
2002	Maes et al.	MLP	MLP trained on preprocessed dataset, adaptive learning rate proved to be beneficial.
2009	Wiese and Omlin	LSTM	LSTM outperformed SVMs as well as the MLP proposed by Maes et al.
2014	Khan et al.	MLP	Model achieves high detection rate at cost of increased false positives and is computationally expensive.
2016	Paula et al.	AE	AE was able to detect the fraudulent cases.
2016	Fu et al.	CNN	CNN achieved F-score of 0.33 and outperformed MLPs.
2017	Heryadi and Warnars	CNN-LSTM	CNN'n short-term and LSTM long-term abilities combined to capture temporal relations. AUC score of 77% achieved.
2017	Kazemi and Zarrabi	AE	AE accuracy of 81.6% achieved in the model.
2018	Wang et al.	MLP	Model addressed the problem of NNs overfitting with F-score of 98.4 percent.
2018	Jurgovsky et al.	LSTM	LSTM with feature aggregation strategy detected fraud behavior.
2018	Pumsirirat and Yan	AE	AE performed poorly with small dataset.
2018	Chen et al.	SAE-GAN	GAN trained on SAE-learned features from majority class has improved F-score and precision, but with a decrease in recall.
2019	Tanaka and Aranha	GAN-DT	DT trained with minority class GAN-based oversampling had slightly higher precision, but lower recall than when using SMOTE or ADASYN.
2019	Fiore et al.	GAN-MLP	MLP trained with GAN-based oversampling had improved recall, and proposed model also outperformed SMOTE in terms of recall, but had slightly lower specificity.
2020	Charitou et al.	SAE-GAN	SAE features extracted from entire train set, then used to train generator of GAN to produce complementary samples.

Data and Methodology

The present analysis is based on exhibit history of United States international trade in goods and services. The foreign trade data (export of goods, export of services, import of goods and import of services) for the purpose of the analysis is drawn for the period 1992-2022 from United States Census Bureau (undated).

First, it is observed that at the composite level, U.S. exports and imports has witnessed an increasing trend over the past two decades. United nations trade flows have registered fairly strong growth over 1992-2008, accompanied by rising commodity prices. Post financial crises in 2008; trade fell steeply before rebounding strongly during 2010-2011 (WTO, 2015; Aggarwal, 2020; Nag et al., 2021; Aggarwal, 2023a, 2023b). After witnessing moderate growth over 2012-2014, an economic downturn is observed in 2015-2016, and thereafter a strong rebound is marked in 2017-2018 (UNCTAD, 2019; Aggarwal, 2020). Following the COVID-19 crises, a major slowdown impacted the U.S economy drastically in 2020 (UNCTAD, 2022), finally a remarkable progress in U.S. trade with world economy has been observed in 2021. Fig. 9 reports the year-wise trend of U.S. trade during the period 1992-2022. Exports of goods has witnessed high growth in its value from US $439 billion to US$2085 billion, import of goods have been marked with ever higher growth in its value from US $536 billion to US $3277 billion, whereas export of services has increased from US $177 billion to US $924 billion and import of services has increased from US $119 billion to US $680 billion respectively in 1992 and 2022.

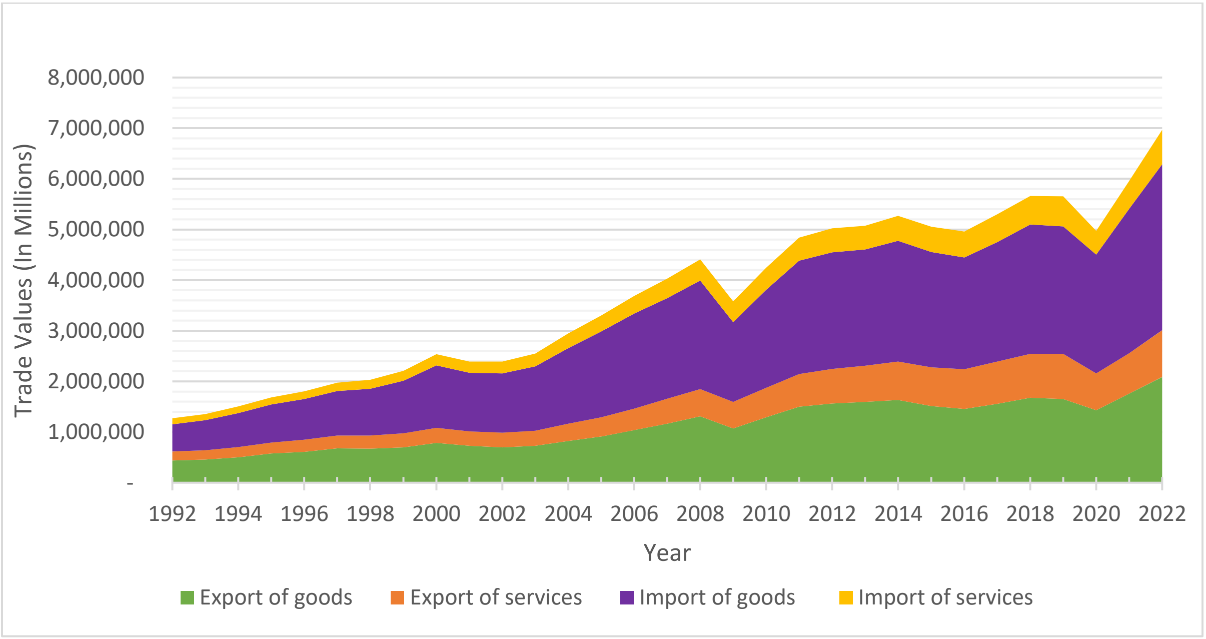

Fig 9. United States trade with ROW (1992-2022)

Second, the input data is standardized (z-score normalized) using ‘scikit-learn’ Python library before applying a machine learning algorithm. To standardize the variables, the mean of each variable is subtracted from each observation and then divided by the standard deviation of that variable (Annexure 1). This centers the variables around zero and scales them to have unit variance. The standardized variables can be represented as:

$$Z = \frac{(X - mean(X))}{std\ dev(X)}$$

where X is the original variable, mean(X) is the mean of X, and std dev(X) is the standard deviation of X. The resulting scores have a mean of 0 and a standard deviation of 1.

Third, LSTM-based anomaly detection is applied to the dataset. In this method, we have considered a time series X = {$\mathbf{x}^{(1)}$, $\mathbf{x}^{(2)}$,…,.$\mathbf{x}^{(n)}$}, where each point $\mathbf{x}^{(t)}$ € R^{m} in the time series is an m-dimensional vector {$x_1^{(t)}$, $x_2^{(t)}$,…,.$x_m^{(t)}$}, whose elements correspond to the input variables. A prediction model learns to predict the next *l* values for d of the input variables s.t. $l \leq d \leq m$. The normal sequence(s) are divided into four sets: normal train (s_N), normal validation -1 (v_{N1}), normal validation – 2 (v_{N2}), and normal test (v_A). We first learn a prediction model using stacked LSTM networks and then compute the prediction error distribution for detecting anomalies in the trade dataset.

LSTM based prediction model: We have considered the following LSTM network architecture. We take one unit in the input layer for each of the m dimensions, d * *l* units in the output layer such that there is one unit for each of the *l* future predictions for each of the d dimensions. The LSTM units in the hidden layer are fully connected through recurrent connections. We have stacked LSTM layers such that each unit in a lower LSTM hidden layer is fully connected to each unit in the LSTM layer above it through feedforward connections. The prediction model is learned using the sequence(s) in s_N. The set v_{N1} is used for early stopping while learning the network weights (Malhotra et al., 2015).

Anomaly detection using the prediction error distribution: With a prediction length of *l*, each of the selected dimensions of $\mathbf{x}^{(t)}$ € X for $l < t \leq n - l$ is predicted *l* times. We compute an error vector $\mathbf{e}^{(t)}$ for point $\mathbf{x}^{(t)}$ as $\mathbf{e}^{(t)}$ = [$e_{11}^{(t)}$,….., $e_{1l}^{(t)}$,……, $e_{d1}^{(t)}$,……, $e_{dl}^{(t)}$], where $e_{ij}^{(t)}$ is the difference between and $x_i^{(t)}$ its value as predicted at time t-j.

Anomaly Detection: Anomaly score from the Multivariate Gaussian Distribution model is calculated as

$$Anamoly\ Score = (\mathrm{e} - \mu)\Sigma^{-1}(\mathrm{e} - \mu)^T$$

Precision: Precision is calculated as the ratio of true positives (the number of correctly identified anomalies) to the total number of positives (the number of all data points identified as anomalies, including false positives). A high precision score indicates that the algorithm is accurately identifying a high proportion of true anomalies while minimizing false positives.

Precision is defined as

$$\text{Precision} = \frac{True\ Positives\ (TP)}{True\ Posiitves\ (TP) + False\ Positive\ (FP)}$$

Recall: Recall is the ratio of true positives to the sum of true positives and false negatives. In the context of anomaly detection, a true positive refers to an actual anomaly that has been correctly identified by the anomaly detection algorithm, while a false negative refers to an anomaly that has been missed by the algorithm and classified as normal.

Mathematically, recall is defined as

$$\text{Recall} = \frac{True\ Positives\ (TP)}{True\ Posiitves\ (TP) + False\ Negative\ (FN)}$$

High recall is desirable in anomaly detection because it indicates that the algorithm is correctly identifying most of the anomalies in the dataset. Anomalies can be important events that require immediate attention or investigation, such as a fraudulent transaction or a network intrusion, and missing them can have serious consequences (Agarwal et al., 2017). Therefore, it is crucial to ensure that the anomaly detection algorithm has a high recall to minimize the risk of missing important anomalies.

F1 score: The F1 score is a commonly used performance metric in anomaly detection that balances the trade-off between precision and recall. It is the harmonic mean of precision and recall and is calculated as

$$\text{F1 score} = \frac{2*(Precision * Recall)}{(Precision + Recall)}$$

The F1 score ranges between 0 and 1, where a score of 1 indicates perfect precision and recall, and a score of 0 indicates poor performance.

In anomaly detection, a high F1 score indicates that the algorithm is both accurately identifying true anomalies and minimizing the number of false positives. However, it's important to note that optimizing for F1 score alone may not always be the best approach, as it depends on the specific use case and the consequences of missing or misclassifying anomalies. Sometimes, optimizing for high recall or high precision may be more important than optimizing for F1 score. Therefore, it's important to evaluate the performance of an anomaly detection algorithm using multiple metrics, including precision, recall, F1 score, and other relevant metrics, and select the best approach based on the specific requirements and constraints of the application.

The current analysis uses monthly times series data for each of the variables (i.e., export of goods, export of services, import of goods and import of services) for the period 1992-2022 where each point in the time series in an m dimensional vector whose elements correspond to the input variables. This implies that a total of 372 points exists in the dataset. Thereafter, LSTM based prediction model is applied where we have first trained the model with a trainset which contains no anomalies, then we have used the trained model to detect anomalies in a testset, where anomalies are included. Fig. 9 highlights the steep fall in trade values *i.e.,* the economic slowdown, which is primarily witnessed in the period 2008-2009 and 2020 due to onset of economic recession and COVID-19 respectively. The model is trained with 100 epochs, weight decay of 0.0001 and window size of 3.

Results

The LSTM based prediction model is trained for 100 epochs. When the model started to learn in epoch 1, the validation loss was 1.8277. At epoch 20, the validation loss was 0.7056 and at epoch 100, the validation loss was reduced to 0.46. This can also be visualized from Fig. 10, Fig. 11 and Fig. 12. Fig. 10 depicts that model has not learned the trend of export of goods, export of services, import of goods and import of services, therefore the prediction (marked by green line) is not accurate. Fig. 11 depicts that model have learned some aspects of the growth trend of export of trade data, leading to better predictions. Fig. 12 depicts that model fully learned the trend of export of goods, export of services, import of goods and import of services, resulting in the accurate prediction of trade data.

Once the model has trained the time series data (with 100 epochs) for export and import of goods and services without anomalies (excluding data-points [197,232], [337,350]). The next step is to use this trained model to detect anomalies in the testset for export of goods, export of services, import of goods and services. The threshold value of anomaly score detected by the algorithm is 60. The first anomaly is detected near point 200 in the below graph (Fig 13 – Fig 17) which is marked by the period of 2008 economic recession, the second anomaly is detected near point 340 which is the period of economic downturn *i.e.,* challenges posed by COVID-19. Interestingly, it can be observed that anomalies are detected accurately for export of goods in Fig 13, export of services in Fig 14, import of goods in Fig 15 and import of services in Fig 16. Therefore, stacked LSTM network architecture has successfully captured the anomalies in the trade dataset.

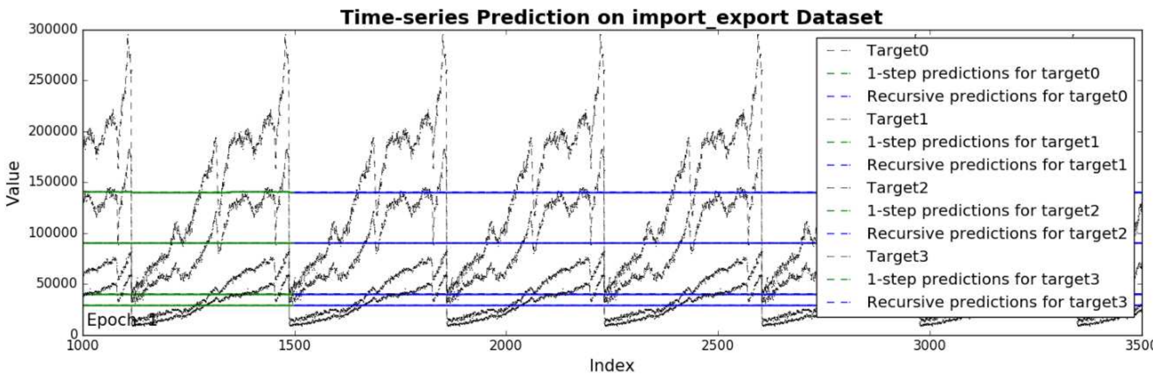

Fig 10. Prediction after Epoch 1

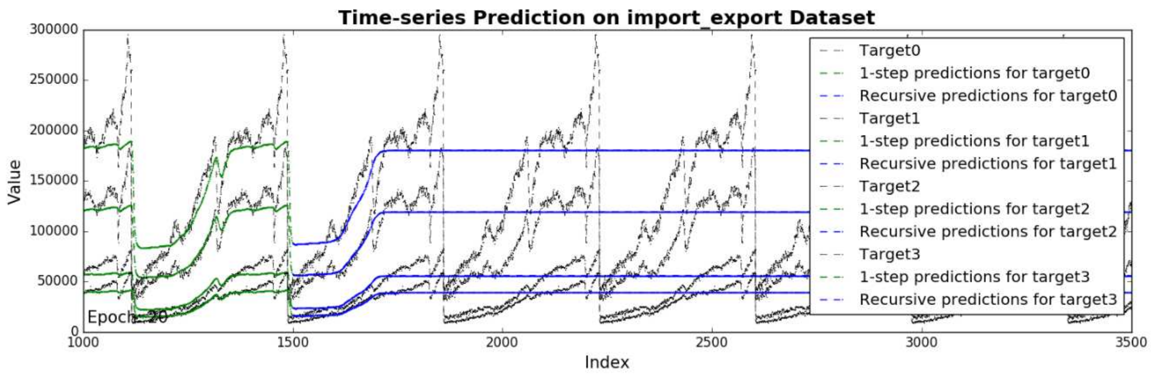

Fig 11. Prediction after Epoch 20

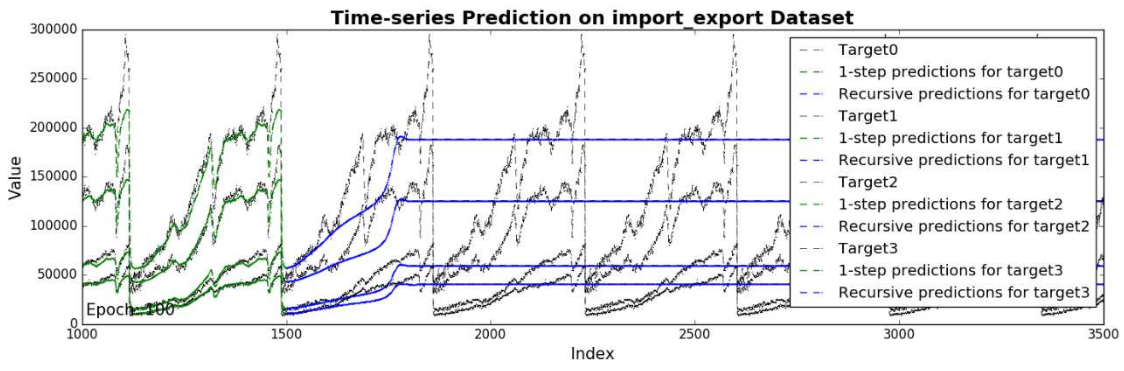

Fig 12. Prediction after Epoch 100

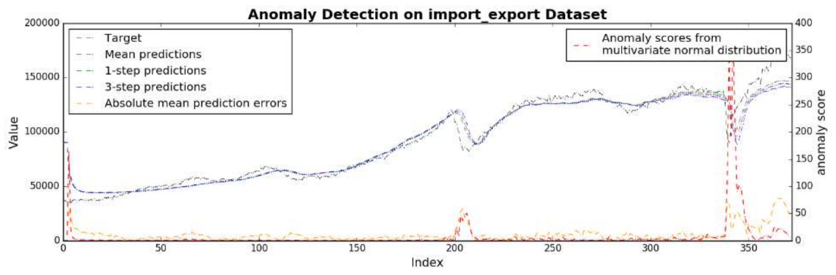

Fig 13. Anomaly detection for export of goods

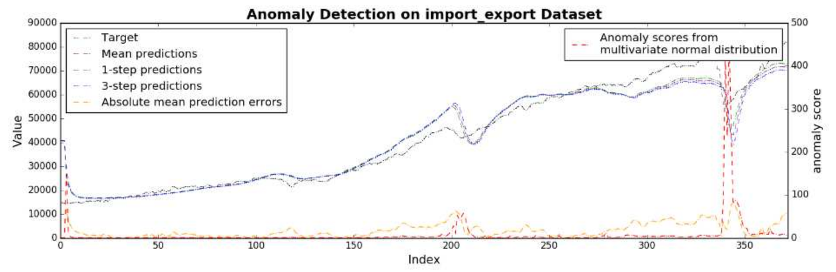

Fig 14. Anomaly detection for export of services

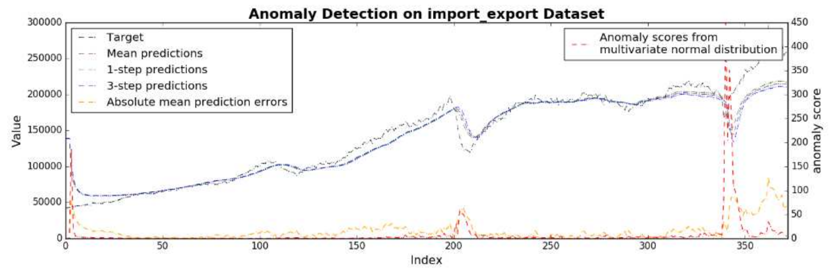

Fig 15. Anomaly detection for import of goods

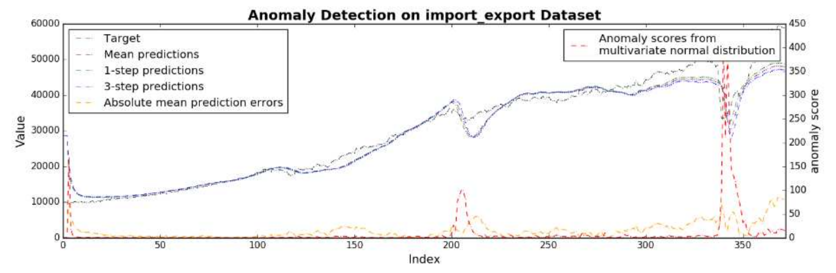

Fig 16. Anomaly detection for import of services

In anomaly detection, a marginal decrease in recall can be worse than a greater increase in precision because it can mean missing a significant number of true anomalies. As mentioned earlier, recall is the proportion of true anomalies that the anomaly detection algorithm identifies correctly. If the algorithm's recall is already high, even a small decrease can result in missing a considerable number of anomalies. Fig. 17 depicts the recall value of 0.8 at threshold anomaly score of 60.

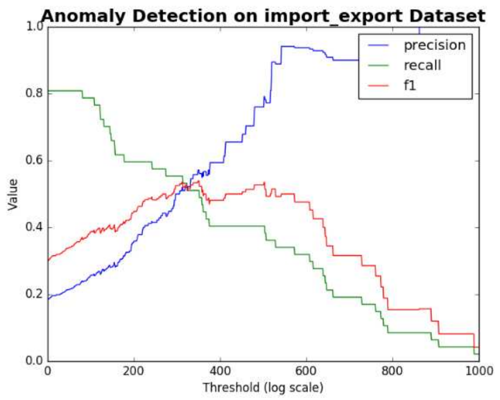

Fig 17. Precision and Recall trade-off for threshold anomaly score

Discussion

The analysis demonstrate that stacked LSTM network are able to learn higher-level temporal patterns without prior knowledge of the pattern duration and so stacked LSTM networks may be a viable technique to model normal time series behaviour, which can then be used to detect anomalies. The LSTM-AD approach deployed in the current analysis yields promising results on real-world international trade datasets which involve modelling small-term as well as long-term temporal dependencies. LSTM-AD provide better or similar results when compared with RNN-AD suggesting that LSTM based prediction models may be more robust in nature compared to RNN based models, especially when the long-term dependencies of the normal time-series behaviour is unknown.

Conclusion

In this paper, we have deployed LSTM network architecture to detect anomalies in United States trade dataset. The model is trained with 100 epochs on the time series dataset where each point $\mathbf{x}^{(t)}$ € R ^{m} in the time series is an m-dimensional vector (export and import of goods and services) whose elements correspond to the input variables. Our learned prediction model using stacked LSTM networks can compute the prediction error distribution for detecting anomalies in the trade dataset and hence calculates the anomaly score that represents the unusual data-points in comparison to normal data. The algorithm flagged the data points with the anomaly score beyond the threshold level and after the research investigation, it was found that the anomalous data-points could be attributed to the period of economic downturn (*i.e.,* financial crises in 2008 and COVID-19 in 2020).

Future Scope

Future research may focus on the introduction of more trade-specific dimensions in the LSTM framework to detect anomalies in the real-world datasets. Also, the robustness of the model can be increased by incorporating other advanced economies as well.

Appendix

Annexure 1.

```
from sklearn import preprocessing, decomposition
data_scaled = sklearn.preprocessing.scale(df1.values)
df1['scaled1'] = data_scaled[:,0]
df1['scaled2'] = data_scaled[:,1]
df1['scaled3'] = data_scaled[:,2]
```

Source: Author’s construction

References

Agarwal, A., Dawson, S., McKee, D., Eugster, P., Tancreti, M., & Sundaram, V. (2017). Detecting abnormalities in IoT program executions through control-flow-based features. In *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation* (pp. 339-340).

Aggarwal, C. C. (2013). Outlier analysis. New York, NY: Springer.

Aggarwal, S. (2016). Determinants of money demand for India in presence of structural break: An empirical analysis. *Business and Economic Horizons* (BEH), Prague Development Center (PRADEC), 12(4), 173-177.

Aggarwal, S. (2017a). Smile curve and its linkages with global value chains. *Journal of Economic Bibliography*, 4(3).

Aggarwal, S. (2017b). Sectoral Level Analysis of India's Bilateral Trade over 2001-2015. *MPRA Paper No 80099,* University Library of Munich, Germany. Retrieved from https://mpra.ub.uni-muenchen.de/80099/ (Accessed on March 10, 2023)

Aggarwal, S. (2020). Determinants of Intra-Industry Trade and Labour Market Adjustment: A Sectoral Analysis for India (Doctoral dissertation, Indian Institute of Foreign Trade).

Aggarwal, S. (2023a). Machine Learning Algorithms, Perspectives, and Real - World Application: Empirical Evidence from United States Trade Data. *International Journal of Science and Research*, 12(3), pp. 292-313. https://www.ijsr.net/getabstract.php?paperid=SR23305084601

Aggarwal, S. (2023b). The empirical measurement and determinants of intra-industry trade for a developing country. *MPRA Paper No. 117112*, University Library of Munich, Germany. Retrieved from https://mpra.ub.uni-muenchen.de/117112/ (Accessed on April 21, 2023).

Aggarwal, S., & Chakraborty, D. (2017). Determinants of India's bilateral intra-industry trade over 2001–2015: Empirical results. *South Asia Economic Journal*, 18(2), 296–313.

Aggarwal, S., & Chakraborty, D. (2019). Which factors influence India's intra-industry trade? Empirical findings for select sectors. *Global Business Review*. Retrieved from https://journals.sagepub.com/doi/10.1177/0972150919868343 (Accessed on April 23, 2020).

Aggarwal, S., & Chakraborty, D. (2020a). Labour market adjustment and intra-industry trade: Empirical results from Indian manufacturing sectors. *Journal of South Asian Development*, 15(2), 238-269.

Aggarwal, S., & Chakraborty, D. (2020b). Determinants of vertical intra-industry trade: Empirical evidence from Indian manufacturing sectors. *Prajnan: Journal of Social and Management Sciences*, 49(3), 221-252.

Aggarwal, S., & Chakraborty, D. (2020c). Is there any relationship between Marginal Intra-Industry Trade and Employment Change? Evidence from Indian Industries. *Working Paper, No. EC-20-44*, Indian Institute of Foreign Trade, Delhi.

Aggarwal, S., Chakraborty, D., & Bhattacharyya, R. (2021). Determinants of Domestic Value Added in Exports: Empirical Evidence from India's Manufacturing Sectors. *Global Business Review*. https://doi.org/10.1177/09721509211050138.

Aggarwal, S., & Chakraborty, D. (2021). Which factors influence vertical intra-industry trade in India? Empirical results from panel data analysis. *Working Paper, No. EC-21-54*, Indian Institute of Foreign Trade, Delhi. Retrieved from http://cc.iift.ac.in/research/Docs/WP/EC-21-54.pdf (Accessed on March 20, 2023)

Aggarwal, S., Chakraborty, D. (2022). Which Factors Influence India's Bilateral Intra-Industry Trade? Cross-Country Empirical Estimates. *Working Papers 2260*, Indian Institute of Foreign Trade, Delhi.

Aggarwal, S., Mondal, S., & Chakraborty, D. (2022). Efficiency Gain in Indian Manufacturing Sectors: Evidence from Domestic Value Addition in Exports. *Empirical Economics Letters*, 21(2): 69-83.

Aggarwal, S., Chakraborty, D., & Banik, N. (2023). Does Difference in Environmental Standard Influence India's Bilateral IIT Flows? Evidence from GMM Results. *Journal of Emerging Market Finance*, 22(1), 7–30. https://doi.org/10.1177/09726527221088412.

Ahmed, M., Mahmood, A. N., & Hu, J. (2014). Outlier detection. *In The state of the art in intrusion prevention and detection* (Vol. 44, pp. 3-21). New York, NY, USA: CRC Press.

Albelwi, S., & Mahmood, A. (2017). A framework for designing the architectures of deep convolutional neural networks. *Entropy*, 19(6), 242.

Anderson, J. E. (1979). A theoretical foundation for the gravity equation. The *American economic review*, 69(1), 106-116.

Barnett, V., & Lewis, T. (1994). *Outliers in statistical data*, 3(1). John Wiley and sons, New York.

Batarseh, F., Gopinath, M., Nalluru, G., & Beckman, J. (2019). Application of machine learning in forecasting international trade trends. *arXiv preprint arXiv:1910.03112*.

Beckman, R. J., & Cook, R. D. (1983). Outlier………. s. *Technometrics*, 25(2), 119-149.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8), 1798-1828.

Bhattacharya, K., Mukherjee, G., and Manna, S. S. (2007). The international trade network. *New Economic Windows* 139–147.

Bolton, R. J., & Hand, D. J. (2001). Unsupervised profiling methods for fraud detection. *Credit scoring and credit control VII*, 235-255.

Boukerche, A., Zheng, L., & Alfandi, O. (2020). Outlier detection: Methods, models, and classification. *ACM Computing Surveys (CSUR)*, 53(3), 1-37.

Callado, A., Kamienski, C., Szabó, G., Gero, B. P., Kelner, J., Fernandes, S., & Sadok, D. (2009). A survey on internet traffic identification. *IEEE communications surveys & tutorials*, 11(3), 37-52.

Chalapathy, R., & Chawla, S. (2019). Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3), 1-58.

Charitou, C., Garcez, A. D. A., & Dragicevic, S. (2020). Semi-supervised GANs for fraud detection. *International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). IEEE.

Chen, K., Lu, S. C., & Teng, H. S. (1990). Adaptive real-time anomaly detection using inductively generated sequential patterns. In *Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE Computer Society Press, 278–284.

Chen, J., Shen, Y., & Ali, R. (2018). Credit card fraud detection using sparse autoencoder and generative adversarial network. *IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)* (pp. 1054-1059). IEEE.

Fiore, U., De Santis, A., Perla, F., Zanetti, P., & Palmieri, F. (2019). Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. *Information Sciences*, 479, 448-455.

Fu, K., Cheng, D., Tu, Y., & Zhang, L. (2016). Credit card fraud detection using convolutional neural networks. In *Neural Information Processing: 23rd International Conference, ICONIP 2016, Kyoto, Japan, October 16–21, 2016, Proceedings, Part III 23* (pp. 483-490). Springer International Publishing.

Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *computers & security*, 28(1-2), 18-28.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10), 2451-2471.

Ghoddusi, H., Creamer, G. G., & Rafizadeh, N. (2019). Machine learning in energy economics and finance: A review. *Energy Economics*, *81*, 709-727.

Ghosh, S., & Reilly, D. L. (1994). Credit card fraud detection with a neural-network. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on* (Vol. 3, pp. 621-630). IEEE.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... & Bengio, Y. (2014). Generative adversarial nets. arXiv. arXiv preprint arXiv:1406.2661.

Gopinath, M., Batarseh, F. A., & Beckman, J. (2020). *Machine learning in gravity models: An application to agricultural trade* (No. w27151). National Bureau of Economic Research.

Gupta, M., Gao, J., Aggarwal, C. C., & Han, J. (2013). Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and data Engineering*, 26(9), 2250-2267.

Hawkins, D. M. (1980). *Identification of outliers* (Vol. 11). Heidelberg, Germany: Springer.

He, H., Wang, J., Graco, W., & Hawkins, S. (1997). Application of neural networks to detection of medical fraud. *Expert systems with applications*, 13(4), 329-336.

Heryadi, Y., & Warnars, H. L. H. S. (2017). Learning temporal representation of transaction amount for fraudulent transaction recognition using CNN, Stacked LSTM, and CNN-LSTM. In *2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)* (pp. 84-89). IEEE.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-1780.

Hodge, V., & Austin, J. (2004). A survey of outlier detection methodologies. *Artificial intelligence review*, 22, 85-126.

Javaid, A., Niyaz, Q., Sun, W., & Alam, M. A. (2016). A deep learning approach for network intrusion detection system, In: *proceedings of the 9th EAI International Conference on bio-inspired information and communications technologies* (formerly BIONETICS). formerly BIONETICS).

Jurgovsky, J., Granitzer, M., Ziegler, K., Calabretto, S., Portier, P. E., He-Guelton, L., & Caelen, O. (2018). Sequence classification for credit-card fraud detection. *Expert Systems with Applications*, 100, 234-245.

Kazemi, Z., & Zarrabi, H. (2017, December). Using deep networks for fraud detection in the credit card transactions. *IEEE 4th International conference on knowledge-based engineering and innovation (KBEI)* (pp. 0630-0633). Tehran, Iran.

Khan, A. U. S., Akhtar, N., & Qureshi, M. N. (2014). Real-time credit-card fraud detection using artificial neural network tuned by simulated annealing algorithm. In *Proceedings of international conference on recent trends in information, telecommunication and computing, ITC* (pp. 113-121).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Networks. *25th Advances in Neural Information Processing Systems*. Grenada, Spain.

Kucharski, D., Kleczek, P., Jaworek-Korjakowska, J., Dyduch, G., & Gorgon, M. (2020). Semi-supervised nests of melanocytes segmentation method using convolutional autoencoders. *Sensors*, 20(6), 1546.

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.

Lin, J., Keogh, E., Fu, A., & Van Herle, H. (2005). Approximations to magic: Finding unusual medical time series. In *18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)* (pp. 329-334). IEEE.

Maes, S., Tuyls, K., Vanschoenwinkel, B., & Manderick, B. (2002). Credit card fraud detection using Bayesian and neural networks. In *Proceedings of the 1st international naiso congress on neuro fuzzy technologies (Vol. 261, p. 270).*

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long Short Term Memory Networks for Anomaly Detection in Time Series. In *ESANN* (Vol. 2015, p. 89).

Markou, M., & Singh, S. (2003a). Novelty detection: a review—part 1: statistical approaches. *Signal processing*, 83(12), 2481-2497.

Markou, M., & Singh, S. (2003b). Novelty detection: a review—part 2: neural network-based approaches. *Signal processing*, 83(12), 2499-2521.

Michelucci, U. (2018). Feedforward neural networks. *Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks*, 83-136.

Miljković, D. (2010). Review of novelty detection methods. In *The 33rd International Convention MIPRO* (pp. 593-598). IEEE.

Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

Monken, A., Haberkorn, F., Gopinath, M., Freeman, L., & Batarseh, F. A. (2021). Graph neural networks for modeling causality in international trade. In *The International FLAIRS Conference Proceedings* (Vol. 34).

Nag, B., Chakraborty, D., & Aggarwal, S. (2021). India's Act East Policy: RCEP Negotiations and Beyond. *Working Paper, No. EC-21-01*, Indian Institute of Foreign Trade, Delhi.

Noble, C. C., & Cook, D. J. (2003). Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 631-636).

Nguyen, T. T., & Armitage, G. (2008). A survey of techniques for internet traffic classification using machine learning. *IEEE communications surveys & tutorials*, 10(4), 56-76.

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

Panford-Quainoo, K., Kigali, R., Bose, A. J., & Defferrard, M. (2020). Bilateral Trade Modeling with Graph Neural Networks. In *ICLR Workshop on Practical ML for Developing Countries*.

Park, J. (2018). RNN based time-series anomaly detector model implemented in Pytorch. *Published in website URL: https://github. com/chickenbestlover/RNN-Time-series-Anomaly-Detection.*

Patcha, A., & Park, J. M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 51(12), 3448-3470.

Paula, E. L., Ladeira, M., Carvalho, R. N., & Marzagao, T. (2016). Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering. In *2016 15th ieee international conference on machine learning and applications (icmla)* (pp. 954-960). IEEE.

Peng, H. K., & Marculescu, R. (2015). Multi-scale compositionality: identifying the compositional structures of social dynamics using deep learning. *PloS one*, 10(4), e0118309.

Pimentel, M. A., Clifton, D. A., Clifton, L., & Tarassenko, L. (2014). A review of novelty detection. *Signal processing*, 99, 215-249.

Pourhabibi, T., Ong, K. L., Kam, B. H., & Boo, Y. L. (2020). Fraud detection: A systematic literature review of graph-based anomaly detection approaches. *Decision Support Systems*, 133, 113303.

Pumsirirat, A., & Liu, Y. (2018). Credit card fraud detection using deep learning based on auto-encoder and restricted boltzmann machine. *International Journal of advanced computer science and applications*, 9(1), 18-25.

Rosseeuw, P. J., & Leroy, A. M. (1987). *Robust regression and outlier detection*. John Willey & Sons. Inc., New York.

S. 1558 – AIIA (2020). Artificial Intelligence Initiative Act. *U.S. Congressional Senate Committee on Commerce, Science, and Transportation*.

Saunders, R., & Gero, J. S. (2000). The importance of being emergent. In *Proceedings of Artificial Intelligence in Design*.

Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., & Stiller, B. (2010). An overview of IP flow-based intrusion detection. *IEEE communications surveys & tutorials*, 12(3), 343-356.

Sun, B., Yu, F., Wu, K., Xiao, Y., & Leung, V. C. (2006). Enhancing security using mobility-based anomaly detection in cellular mobile networks. *IEEE Transactions on Vehicular Technology*, 55(4), 1385-1396.

Sun, B., Osborne, L., Xiao, Y., & Guizani, S. (2007a). Intrusion detection techniques in mobile ad hoc and wireless sensor networks. *IEEE Wireless Communications*, 14(5), 56-63.

Sun, B., Xiao, Y., & Wang, R. (2007b). Detection of fraudulent usage in wireless networks. *IEEE Transactions on Vehicular Technology*, 56(6), 3912-3923.

Sun, B., Wu, K., Xiao, Y., & Wang, R. (2007c). Integration of mobility and intrusion detection for wireless ad hoc networks. *International Journal of Communication Systems*, 20(6), 695-721.

Tanaka, F. H., & Aranha, C. (2019). *Data Augmentation Using GANs*. Retrieved from arXiv:1904.09135.

United States Central Bureau (undated), " U.S. International Trade in Goods and Services", available at: https://www.census.gov/foreign-trade/statistics/historical/index.html (accessed March 01, 2023).

United Nations Conference on Trade and Development (2019). World Investment Report, *Geneva: WTO*.

United Nations Conference on Trade and Development (2022). Trade and Development Report, *Geneva: WTO*.

Wang, C., Wang, Y., Ye, Z., Yan, L., Cai, W., & Pan, S. (2018). Credit card fraud detection based on whale algorithm optimized BP neural network. *13th international conference on computer science & education (ICCSE)* (pp. 1-4). IEEE.

Wang, X., Du, Y., Lin, S., Cui, P., Shen, Y., & Yang, Y. (2020). adVAE: A self-adversarial variational autoencoder with Gaussian anomaly prior knowledge for anomaly detection. *Knowledge-Based Systems*, 190, 105187.

Wiese, B., & Omlin, C. (2009). *Credit card transactions, fraud detection, and machine learning: Modelling time with LSTM recurrent neural networks* (pp. 231-268). Springer Berlin Heidelberg.

Wohl, I., & Kennedy, J. (2018). Neural network analysis of international trade. *US International Trade Commission: Washington*, DC, USA.

World Trade Organisation (2011). *Trade patterns and global value chains in East Asia: From trade in goods to trade in tasks*. In collaboration with institute of developing economies (IDE) and Japan external trade organization (JETRO). WTO.

World Trade Organisation (2015). International Trade Statistics 2015, available at: www.wto.org/statistics (Accessed on September 5, 2022).

Xiong, H., Pandey, G., Steinbach, M., & Kumar, V. (2006). Enhancing data analysis with noise removal. *IEEE transactions on knowledge and data engineering*, 18(3), 304-319.

Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: LSTM cells and network architectures. *Neural computation*, 31(7), 1235-1270.

Zhang, W., Yang, Q., & Geng, Y. (2009). A survey of anomaly detection methods in networks. In 2009 *International Symposium on Computer Network and Multimedia Technology* (pp. 1-3). IEEE.

Zimek, A., Schubert, E., & Kriegel, H. P. (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 5(5), 363-387.