

Integration of PBFT and Raft Algorithms with Recurrent Neural Networks to Improve the Reliability of Distributed Systems

Alexander Bogdanov[1], Nadezhda Shchegoleva[1,2], Valery Khvatov[3], Jasur Kiyamov[1,5(✉)], Gennady Dik[1,4], Ilkhom Rakhmatullayev[5], Shakhboz Ergashev[6], and Oybek Umurzakov[5]

[1] St. Petersburg University, Saint Petersburg, Russia
{a.v.bogdanov,n.shchegoleva,z.kiyamov}@spbu.ru
[2] St. Petersburg State Marine Technical University, Saint Petersburg, Russia
[3] DGT Technologies AG, Lachine, Canada
[4] St. Petersburg LLC "System Technologies", Saint Petersburg, Russia
g.dick@systechnologies.ru
[5] Samarkand Branch of Tashkent University of Information Technologies, Samarkand, Uzbekistan
[6] Millitary Institute of Information and Communication Technologies, Tashkent, Uzbekistan

Abstract. The combined approach proposes the use of PBFT and Raft to ensure data consistency and fault tolerance in the system, and also integrates recurrent neural networks to analyze and predict the behavior of nodes in the network. RNNs can be used to detect anomalies, predict system load, and analyze time series data related to node operation. The proposed combined approach opens up new prospects for the development of distributed systems, increasing their reliability, fault tolerance and adaptability to changing conditions. Further research in this direction could lead to more efficient and secure distributed systems that can efficiently handle complex real-world scenarios.

Keywords: PBFT · Raft · Recurrent Neural Networks · Reliability Ensuring · Distributed Systems

1 Introduction

The continuous search for new methods and algorithms is an integral part of the development of distributed systems. These methods must be able to ensure stable system operation even in the face of unexpected failures or attacks. Therefore, the development and application of mechanisms to ensure fault tolerance and data reliability becomes a key task for engineers and developers.

Fault tolerance in distributed systems is achieved through a number of methods, including data replication, backup mechanisms, and consensus algorithms.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2024
O. Gervasi et al. (Eds.): ICCSA 2024 Workshops, LNCS 14815, pp. 226–237, 2024.
https://doi.org/10.1007/978-3-031-65154-0_14

Consensus algorithms such as PBFT (Practical Byzantine Fault Tolerance) and Raft play a critical role in ensuring data consistency in distributed environments [1,2].

However, recent research also shows that the use of neural networks, in particular recurrent neural networks (RNN), can significantly improve anomaly detection, system load forecasting, and time series analysis of node performance data.

Thus, the integration of consensus algorithms with recurrent neural networks represents a promising approach to improve the reliability and fault tolerance of distributed systems. This symbiosis of traditional methods and new technologies opens the door to the creation of more efficient, secure and adaptive distributed systems that can effectively cope with the challenges of the modern world [3].

2 Applications of Neural Networks for Analysis

The combined approach presented in this article combines two leading consensus algorithms - PBFT (Practical Byzantine Fault Tolerance) and Raft. PBFT is distinguished by its ability to provide data consistency in systems with a limited number of nodes, while Raft offers a simple and straightforward consensus mechanism for distributed systems (Fig. 1).

The inclusion of recurrent neural networks in this combined approach opens up new horizons for analyzing and predicting the behavior of nodes in the network. Recurrent neural networks can be used to detect anomalies, predict system load, and analyze time series data related to node activity. This innovative approach provides the ability to effectively detect and respond to changes in system operation.

In the context of distributed systems, recurrent neural networks can be used to detect anomalies in the operation of nodes. They are able to analyze time series of data related to node activity and identify deviations from normal behavior, which allows them to quickly respond to possible problems.

In addition, recurrent neural networks make it possible to predict the load on the system. By analyzing historical load and behavior data of nodes, they can predict future resource demands and temporary peaks in activity, which helps optimize system performance and warn of potential problems.

Integrating recurrent neural networks into a combination approach also provides the ability to analyze time series data, which allows us to better understand the dynamics of the system and identify hidden patterns or trends that can be useful for decision making.

The integration of PBFT, Raft and recurrent neural networks represents a promising path for the development of distributed systems. This symbiosis of traditional methods and new technologies opens the door to the creation of more reliable, secure and adaptive distributed systems that can effectively overcome the challenges of the modern world.

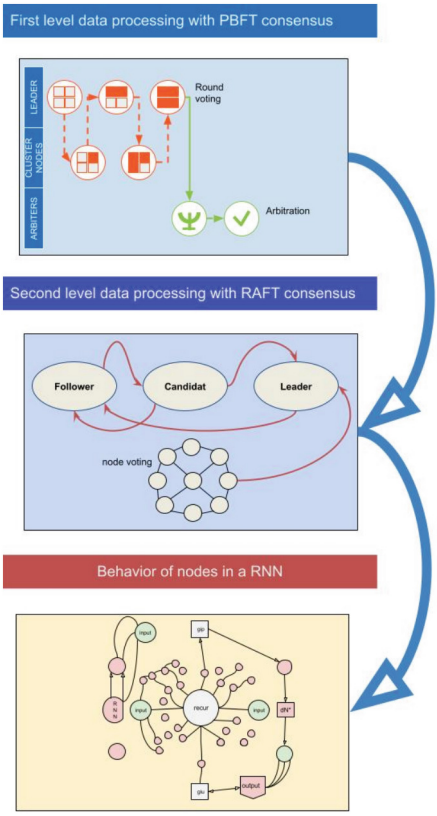

Fig. 1. Network process

3 Process of Operation of Recurrent Neural Networks

In the operation of recurrent neural networks (RNN), data is transferred through successive time steps, where each step includes input data and hidden layers [4]. RNNs have a special property called recurrence, which allows them to remember information about previous states and use this information for subsequent calculations.

Each hidden layer in an RNN has its own short-term memory, which allows the model to take into account the context and sequence of the input data [5]. This short-term memory allows the RNN to "remember" information about previous inputs and use it to make decisions in subsequent steps (Fig. 2).

When new data is input to an RNN, the model uses the current input value in combination with previous values stored in short-term memory to predict the next output or state. This process is repeated at every time step, making RNN a powerful tool for analyzing and processing sequential data such as time series, text, or audio signals.

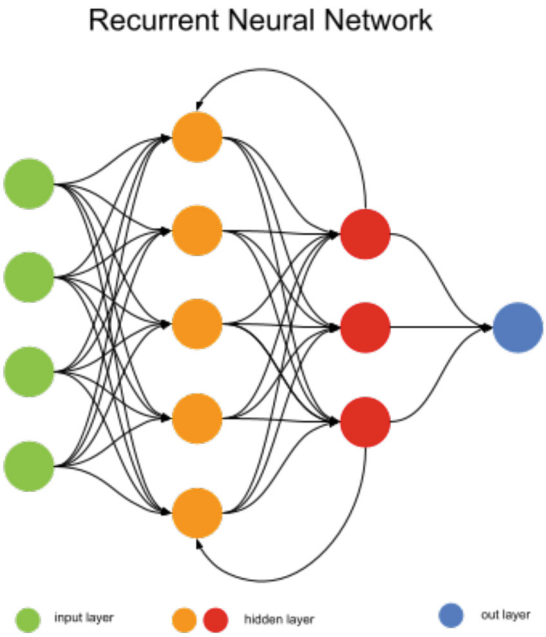

Fig. 2. RNN operation model

RNN There are different types of connections that define the interactions between nodes in the network [11,12]. Here are some of the main types of connections in RNN:

1. Unidirectional connections: Information is passed only in one direction, from input neurons to output neurons. This is a typical configuration in RNNs, where each neuron in the hidden layer is connected to the previous and subsequent layers;
2. Bidirectional connections: Information is passed both forward and backward through the neurons in the network. This allows considering context from both past and future when analyzing sequential data;
3. Self-connections: These connections allow a neuron to pass information to itself from previous time steps. They help maintain the neuron's state or detect long-term dependencies in the data.
4. Self-connections: These connections allow a neuron to pass information to itself from previous time steps. They help maintain the neuron's state or detect long-term dependencies in the data.
5. Fully connected layers: In this case, each neuron in one layer is connected to every neuron in the next layer. This is a typical configuration for fully connected layers used in neural networks.

They help determine the structure of the network and how information is transmitted and processed over time.

The introduction of RNNs into a combined approach represents a significant breakthrough in analyzing and predicting the behavior of nodes in a distributed network. RNNs have the unique ability to remember and use previous input data, making them a powerful tool for analyzing time series data.

The network analysis algorithm for a combination of the P-BFT and RAFT protocols may include the following steps:

1. Definition of network architecture:
   - Define the network structure, including the number of nodes, their types (leaders, match nodes, etc.), and the configuration of each layer (P-BFT and RAFT).
   - Determine the mechanisms for exchanging messages between nodes in the network.
2. Study of P-BFT and RAFT protocols:
   - Study the basic operating principles of each protocol, their leader selection algorithms, coordination and decision-making mechanisms.
3. Security analysis:
   - Evaluate the security level of each protocol and its suitability for a particular network.
   - Research vulnerabilities and possible attacks that could be directed at the network.
4. Determining network parameters:
   - Define network parameters such as number of nodes, timeout parameters, decision thresholds, etc.
5. Simulation and testing:
   - Create a network model using the selected parameters and protocols.
   - Conduct a series of test trials using various scenarios such as node failures, message delays, etc.
   - Evaluate network performance, its resilience to failures, and its ability to adapt to changing environmental conditions.
6. Optimization and improvement:
   - Identify weaknesses in the network and its protocols.
   - Propose and implement improvements to improve network performance, security and resiliency.
7. Monitoring and maintenance:
   - Install monitoring and tracking systems to continuously monitor the health of the network.
   - Perform periodic testing and maintenance of the network to ensure optimal operation.

This is the main network analysis algorithm for the combination of P-BFT and RAFT protocols. When implementing the algorithm, the specifics of a particular network and its goals should be taken into account.

With this ability, RNNs can effectively detect anomalies in node operation, predict system load, and analyze changes over time. This allows the system to quickly respond to changes in its operation and take the necessary measures to maintain stable and reliable operation.

Integrating RNNs into a combined approach opens up new horizons for improving the reliability, fault tolerance, and adaptability of distributed systems. Analyzing the data provided by RNN allows the system to more effectively adapt to changing conditions and successfully overcome complex scenarios encountered in the real world [6,7].

Research in this direction can lead to the creation of more efficient and reliable distributed systems that can effectively respond to changes in the environment and ensure stable operation in a wide variety of conditions. This opens up new prospects for developing technologies and improving the functioning of distributed networks in general.

4 Network Communication Model

A mathematical model of a combined approach using recurrent neural networks (RNN), PBFT and Raft in distributed systems can be represented as follows:

1. Recurrent neural networks (RNN):
   - RNNs are a class of neural networks in which information from past time steps is stored and passed on to the next steps;
   - Mathematically, RNNs are defined by the recurrence relation:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$
$$y_t = g(W_{hy}h_t + b_y)$$

   - Where h_t is the hidden state at time step t, x_t is the input vector at time step t, W_{xh}, W_{hh}, W_{hy} are weight matrices, b_h, b_y are displacement vectors, f and g are activation functions.
2. PBFT (Practical Byzantine Fault Tolerance):
   - PBFT is a consensus algorithm for distributed systems that ensures data consistency in the face of asynchrony and unreliability;
   - Mathematically, PBFT relies on the idea of quorums and signature verification mechanisms.
   - PBFT uses quorums to make decisions. Let's assume there are N nodes in the system, and to reach an agreement, it's required that a majority of nodes ($N - f$, where f is the maximum number of faulty nodes) agree on a proposed value. Let M be the set of messages received by each node. Suppose $Q(m)$ is a function returning a quorum, where m is a message. Mathematically, the quorum $Q(m)$ can be defined as:

$$Q(m) = \{m' \in M : |\{m'' \in M : \text{m}'' \text{ supports m}'\}| > N - f\}$$

3. Raft:
   - Raft is also a consensus algorithm that offers a simple mechanism for leader election and log replication to maintain consistency [8,14].
   - Mathematically, Raft is determined by the logic of leader selection and the decision-making algorithm based on the majority of votes.

– Raft is based on leader election and log replication. Let N be the number of nodes in the system. To make a decision, each node votes for a leader candidate. If a candidate receives more than half of the votes (more than $N/2$), it becomes the leader. Mathematically, the leader election condition can be expressed as:

$$\text{If votes for candidate} > \frac{N}{2}, \text{then it becomes the leader}$$

The combined approach combines these algorithms using recurrent neural networks to analyze and predict the behavior of nodes in the network [9,13]. The mathematical model of the combined approach can include sharing the output of the RNN with the results of the PBFT and Raft algorithms for decision making and adaptation to changes in the distributed system [10]. PBFT and Raft ensure data consistency and efficient consensus management, while the integration of RNN complements their capabilities with analysis and prediction of node behavior in the network.

1. Reliability and Fault Tolerance: This refers to the system's ability to operate continuously and reliably even in the presence of faults or failures. A reliable system maintains its functionality and integrity despite adverse conditions.
2. Combined Approach: By combining PBFT, Raft, and RNN, we leverage the strengths of each component to enhance the overall reliability and fault tolerance of the system. This synergistic approach ensures that the system can withstand various challenges and continue functioning optimally.
3. PBFT and Raft for Data Consistency and Consensus Management:
   – PBFT is known for its ability to achieve consensus in distributed systems, even in the presence of Byzantine faults, where nodes may behave arbitrarily. It ensures data consistency by reaching agreement among a designated quorum of nodes;
   – Raft is another consensus algorithm that simplifies distributed system design while providing strong consistency guarantees. It elects a leader among the nodes, which coordinates the replication of the log and ensures data consistency across the cluster.
4. Integration of RNN for Node Behavior Analysis and Prediction: RNNs are a class of neural networks capable of modeling sequential data. By integrating RNNs into the system, we gain the ability to analyze and predict the behavior of nodes in the network over time. RNNs can detect anomalies, forecast node workload, and analyze temporal data patterns associated with node operations.
5. Complementary Capabilities: The integration of RNN complements the consensus algorithms (PBFT and Raft) by providing additional insights into the behavior of nodes. This combination enhances the system's ability to detect and respond to anomalies or deviations from expected behavior, thereby improving overall reliability and fault tolerance.

Formalizing an RNN algorithm for network analysis of a combination of P-BFT and RAFT protocols may involve defining mathematical expressions to

evaluate the performance, security, and resilience of the network. The following are possible types of formulas that can be used in such an analysis:

1. Loss Function:
   - $L_{total} = L_{P-BFT} + L_{RAFT}$ - the total loss function, combining the loss functions for the P-BFT and RAFT protocols.
2. Performance:
   - $T^{P-BFT}_{round-trip}$ - the time taken to process one round-trip of consensus in the P-BFT protocol.
   - $T^{RAFT}_{round-trip}$ - the time taken to process one round-trip of consensus in the RAFT protocol.
   - $T^{total}_{round-trip} = T^{P-BFT}_{round-trip} + T^{RAFT}_{round-trip}$ - the total time taken to process one round-trip of consensus in the combination of protocols.
   - $Throughput_{total} = \frac{1}{T^{total}_{round-trip}}$ - the overall throughput of the network.
3. Security:
   - P^{P-BFT}_{attack} - the probability of a successful attack on the P-BFT protocol.
   - P^{RAFT}_{attack} - the probability of a successful attack on the RAFT protocol.
   - $P^{total}_{attack} = P^{P-BFT}_{attack} + P^{RAFT}_{attack}$ - the overall probability of a successful attack on the combination of protocols.
4. Resilience:
   - $T^{P-BFT}_{recovery}$ - the time taken to recover the network after a node failure in the P-BFT protocol.
   - $T^{RAFT}_{recovery}$ - the time taken to recover the network after a node failure in the RAFT protocol.
   - $T^{total}_{recovery} = T^{P-BFT}_{recovery} + T^{RAFT}_{recovery}$ - the total time taken to recover the network.

The loss function for the analysis of a network combining P-BFT and RAFT protocols depends on the specific task or objective being considered. In general, the loss function is used to quantify the discrepancy between the predicted output of the RNN model and the ground truth.

Here's a general form of the loss function for such an analysis:

Let's denote:

- $\hat{y}_i$ as the predicted output of the RNN model for the i-th example,
- y_i as the ground truth output for the i-th example.

The loss function L can be defined as the average loss across all examples in the dataset, and it can be calculated using various metrics such as mean squared error (MSE), mean absolute error (MAE), or cross-entropy loss (for classification tasks), among others.

For example, the mean squared error (MSE) loss function can be expressed as:

$$L = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

where:

- N is the total number of examples in the dataset,
- $\hat{y}_i$ is the predicted output of the RNN model for the i-th example,
- y_i is the ground truth output for the i-th example.

This loss function quantifies the average squared difference between the predicted and actual outputs across all examples in the dataset. The goal during training is to minimize this loss function, typically using optimization algorithms like stochastic gradient descent (SGD) or its variants.

The performance of an RNN in analyzing a network combining P-BFT and RAFT protocols can be evaluated using various metrics. Here are some key performance metrics that can be considered:

1. Throughput: Throughput refers to the rate at which the network can process transactions or messages. In the context of RNN analysis, throughput can be measured as the number of transactions successfully processed per unit time.
2. Latency: Latency measures the delay between the initiation of a request and the completion of the corresponding operation. Lower latency indicates faster processing and better performance.
3. Accuracy: Accuracy measures how well the RNN model can predict or classify data. It is typically defined as the percentage of correct predictions or classifications made by the model.
4. Resource Utilization: Resource utilization evaluates how efficiently the computational resources (such as CPU, memory, and bandwidth) are utilized in the network. Higher resource utilization can lead to improved performance, while excessive utilization may result in bottlenecks and decreased performance.
5. Fault Tolerance: Fault tolerance measures the network's ability to continue operating correctly in the presence of faults or failures. This includes the ability to detect and recover from node failures, network partitions, or other types of disruptions.
6. Scalability: Scalability refers to the network's ability to handle an increasing number of transactions or users while maintaining performance. A scalable network can accommodate growth without significant degradation in performance.
7. Security: Security evaluates the network's resilience to attacks, unauthorized access, and data breaches. It includes measures such as encryption, authentication, and access control mechanisms to protect sensitive information and ensure the integrity of transactions.

These performance metrics can be quantitatively evaluated through simulations, experiments, or real-world deployments of the RNN model in the network environment.

1. Cryptographic Mechanisms: Evaluate the cryptographic mechanisms used by both P-BFT and RAFT protocols to ensure data confidentiality, integrity, and authentication. This includes assessing the strength of encryption algorithms, key management practices, and digital signature schemes.

2. Consensus Protocol Security: Analyze the security properties of the P-BFT and RAFT protocols, including resistance to various attacks such as Sybil attacks, Byzantine faults, and Denial of Service (DoS) attacks. Assess the robustness of the protocols in maintaining consensus and preventing malicious behavior by network participants.
3. Node Authentication: Verify the mechanisms for authenticating nodes in the network to prevent unauthorized access and impersonation attacks. This may involve using cryptographic certificates, public-key infrastructure (PKI), or other authentication methods to verify the identity of network participants.
4. Data Confidentiality: Ensure that sensitive data transmitted over the network is protected from eavesdropping and unauthorized access. Evaluate the use of encryption techniques such as Transport Layer Security (TLS) or Secure Socket Layer (SSL) to encrypt data in transit and protect it from interception.
5. Access Control: Implement access control mechanisms to restrict access to critical network resources and functions. This includes defining roles and permissions for network participants, enforcing authentication and authorization policies, and monitoring access logs for suspicious activity.
6. Resilience to Attacks: Assess the network's resilience to various types of cyber attacks, including distributed denial-of-service (DDoS) attacks, man-in-the-middle attacks, and insider threats. Implement measures such as intrusion detection systems (IDS), firewalls, and network segmentation to detect and mitigate potential security breaches.
7. Regular Audits and Security Updates: Conduct regular security audits and vulnerability assessments to identify and address potential security weaknesses in the network. Stay informed about security advisories and updates for the P-BFT and RAFT protocols, and promptly apply patches and fixes to mitigate known vulnerabilities.

By addressing these security considerations, researchers and practitioners can enhance the security posture of a network combining P-BFT and RAFT protocols analyzed using an RNN model, reducing the risk of security breaches and ensuring the integrity and confidentiality of network transactions.

This entire system is a complex mathematical model that requires the analysis and optimization of various parameters, such as weights and hyperparameters of neural networks, parameters of consensus algorithms.

5 Conclusions

In conclusion, the key aspects of the combined approach using PBFT, Raft and recurrent neural networks (RNN) for the development of distributed systems should be highlighted.

1. Reliability and fault tolerance: The combined approach allows you to create systems with a high degree of reliability and fault tolerance. PBFT and Raft provide data consistency and efficient consensus management, and RNN integration complements their capabilities by analyzing and predicting the behavior of nodes in the network;

2. Adaptability to changing conditions: The use of RNN allows systems to adapt to dynamic changes in the environment. Analysis of time series data and forecasting allow the system to quickly respond to changes and effectively adapt to new conditions;
3. Efficiency and Security: The combined approach helps to create more efficient and secure distributed systems. PBFT and Raft algorithms ensure stable system operation, and RNNs provide tools for detecting anomalies and preventing possible security threats
4. Potential for development: Further research in this direction could lead to the creation of more innovative and efficient distributed systems. Continuous improvement of algorithms and data analysis methods opens up new prospects for developing technologies and improving the functioning of distributed networks.

Security is a critical aspect of analyzing a network that combines P-BFT and RAFT protocols using an RNN model. Here are some considerations for evaluating the security of such a network:

So, a combined approach based on the joint use of PBFT, Raft and RNN is a promising direction for the development of distributed systems. It combines reliability, adaptability and security, making it an essential tool in the quest to create efficient and reliable infrastructures for various applications and industries.

References

1. Bogdanov, A., Shchegoleva, N., Khvatov, V., Dik, G., Kiyamov, J., Dik, A.: The Combination of P-BFT and RAFT: a new approach to building networks that provide reliability and security. In: Gervasi, O., et al. (eds.) Computational Science and Its Applications – ICCSA 2023 Workshops: Athens, Greece, July 3–6, 2023, Proceedings, Part I, pp. 572–583. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-37105-9_38
2. Bogdanov, A., et al.: A multilayer approach to the security of blockchain networks of the future. In: Gervasi, O., Murgante, B., Misra, S., Rocha, A.M.A.C., Garau, C. (eds.) Computational Science and Its Applications – ICCSA 2022 Workshops: Malaga, Spain, July 4–7, 2022, Proceedings, Part I, pp. 205–216. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-10536-4_14
3. Ongaro, D., Ousterhout, J.: In search of an understandable consensus algorithm. In: Proceedings of the 2014 USENIX Annual Technical Conference (USENIX ATC), pp. 305–319 (2014)
4. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
6. Zhang, H., Juels, A., Reiter, M.K., Ristenpart, T.: Town Crier: an authenticated data feed for smart contracts. Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 270–282 (2016)
7. Chollet, F.: Deep Learning with Python. Manning Publications (2017)
8. Gilad-Bachrach, R., Navot, A., Reichental, O.: Machine Learning for Anomaly Detection. Springer (2020)

9. Bernstein, P.: Principles of Transaction Processing. Morgan Kaufmann (2017)
10. Haber, S., Stornetta, W.: How to time-stamp a digital document. J. Cryptol. **3**(2), 99–111 (1991)
11. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
12. Le, Q.V., Jaitly, N., Hinton, G.E.: A simple way to initialize recurrent networks of rectified linear units. arXiv preprint arXiv:1504.00941 (2015)
13. Rakhmatullaevich, R.I., Mardanokulovich, I.B.: Analysis of cryptanalysis methods applied to stream encryption algorithms. In: Artificial Intelligence, Blockchain, Computing and Security, vol. 1. - CRC Press, pp. 393–401 (2024)
14. Zaynalov, N.R., et al.: UNICODE for hiding information in a text document. In: 2020 IEEE 14th International Conference on Application of Information and Communication Technologies (AICT), pp. 1–5. IEEE (2020)