



You:

explain the following abstract of the paper in simple words "Abstract—Network attack is a significant security issue for modern society. From small mobile devices to large cloud platforms, almost all computing products, used in our daily life, are networked and potentially under the threat of network intrusion. With the fast-growing network users, network intrusions become more and more frequent, volatile and advanced. Being able to capture intrusions in time for such a large scale network is critical and very challenging. To this end, the machine learning (or AI) based network intrusion detection (NID), due to its intelligent capability, has drawn increasing attention in recent years. Compared to the traditional signature-based approaches, the AI-based solutions are more capable of detecting variants of advanced network attacks. However, the high detection rate achieved by the existing designs is usually accompanied by a high rate of false alarms, which may significantly discount the overall effectiveness of the intrusion detection system. In this paper, we consider the existence of spatial and temporal features in the network traffic data and propose a hierarchical CNN+RNN neural network, LuNet. In LuNet, the convolutional neural network (CNN) and the recurrent neural network (RNN) learn input traffic data in sync with a gradually increasing granularity such that both spatial and temporal features of the data can be effectively extracted. Our experiments on two network traffic datasets show that compared to the state-of-the-art network intrusion detection techniques, LuNet not only offers a high level of detection capability but also has a much low rate of false positive-alarm. "



ChatGPT:

In simple terms, this paper talks about a problem: attacks on computer networks are a big issue in today's world, affecting everything from small devices to large cloud platforms that we use every day. As more people use networks, these attacks become more frequent and sophisticated. Detecting and stopping these attacks is crucial but challenging.

The paper suggests using machine learning, specifically a type called neural networks, to help detect these network attacks. Unlike traditional methods that rely on predefined patterns of attacks (like a signature), machine learning can adapt and learn from new types of attacks. However, existing machine learning solutions often have a problem – they may detect too many false alarms, mistakenly thinking there's an attack when there isn't.

To address this issue, the paper introduces a new neural network called LuNet, which combines two types of neural networks: convolutional neural network (CNN) and recurrent neural network (RNN). These networks work together to analyze network traffic data, considering both the spatial and temporal aspects (where things are and how they change over time). The experiments conducted using two sets of network traffic data show that LuNet not only detects network attacks effectively but also has a much lower rate of false alarms compared to other advanced techniques.



You:

explain the following in the great detail in simple words "Keywords—network intrusion detection; convolutional neural network; LuNet; recurrent neural network"



Sure, let's break down those keywords in simple terms:

1. Network Intrusion Detection:

- *Network*: Think of it like a web of connected devices, like your computer, phone, or the servers that store data.
- *Intrusion*: An unwanted entry or interference. In this context, it refers to someone or something trying to compromise the security of this network.
- *Detection*: The act of identifying or noticing when there's a potential security threat.

So, "Network Intrusion Detection" means finding and recognizing when something is trying to mess with the security of a group of connected devices.

2. Convolutional Neural Network (CNN):

- *Neural Network*: Imagine a computer system that learns from examples, somewhat like how your brain learns from experiences.
- *Convolutional*: This term involves a specific way of processing information, especially useful for recognizing patterns in things like images.

Therefore, a "Convolutional Neural Network" is a type of computer system that's really good at recognizing patterns, especially in things like images or data that has a spatial structure.

3. Recurrent Neural Network (RNN):

- *Recurrent*: This term implies something that repeats or happens in a sequence, like steps in a process.
- *Neural Network*: As mentioned before, it's a computer system that learns from examples.

So, a "Recurrent Neural Network" is a type of system that's good at understanding and learning from sequences of information, where the order matters.

4. LuNet:

- This is likely a specific neural network model mentioned in the paper. The "LuNet" is a combination of a Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN). It's designed to be effective in detecting network intrusions by considering both patterns (CNN) and sequences of data (RNN).

In summary, the keywords suggest that the paper is discussing a system (LuNet) that uses a combination of advanced computer techniques (CNN and RNN) to effectively detect and recognize potential security threats in networks.



explain the following in the simple language "I. INTRODUCTION Networked computing becomes indispensable to people's life. From daily communications to commercial transactions, from small businesses to large enterprises, all activities can be or will soon be done through networked services. Any vulnerabilities in the networked devices and computing platforms can expose the whole network under various attacks and may bring about disastrous consequences. Hence, effective network intrusion detection (NID) solutions are ultimately essential to the modern society"



Certainly! In simple terms, the introduction is saying:

Our lives now heavily depend on using computers that are connected to networks. Whether it's talking with friends, buying things online, or running a business, we rely on these connected services. However, if there are weaknesses or problems in the devices (like computers and phones) that are part of these networks, it can make the entire network vulnerable to attacks. These attacks can have really bad consequences. That's why it's super important for us to have good ways of detecting and stopping these attacks, which is called Network Intrusion Detection (NID). Essentially, it's about making sure our computer networks are safe and secure for the things we do every day.



explain the following the simple language "So far, investigations on the AI-based solutions are mainly based on two schemes: anomaly detection and misuse detection. The anomaly detection identifies an attack based on its anomalies deviated from the profile of normal traffic. Nevertheless, this scheme may have a high false positive rate if the normal traffic is not well profiled, and the profile used in the detection is not fully representative [2]. Furthermore, to obtain a fully representative normal traffic profile for a dynamically evolving and expanding network is unlikely possible"



Certainly! In simple terms, the passage is explaining two ways that people have been looking into using artificial intelligence (AI) to deal with network attacks.

1. Anomaly Detection:

- Imagine your normal network traffic is like a daily routine. Anomaly detection tries to find anything unusual or different from this routine because it might be a sign of an attack.
- However, there's a problem. If we don't really understand what normal traffic looks like, or if the way we define normal is not accurate, we might end up thinking there's an attack when there isn't. This is called a "false positive." It's like a security system that thinks your friend is a burglar just because they came in through a different door.

2. Misuse Detection:

- This is like having a list of known bad things. If anything matches that list, it's considered an attack.
- But, here's the catch. If there's something new that's not on the list, the system might not recognize it as an attack. It's like having a list of known bad guys, but not being able to spot a new bad guy who's never been seen before.

The passage is saying that while these AI-based methods are useful, they have their challenges. Anomaly detection might give us too many false alarms if we don't understand what normal looks like well enough. And, getting a perfect understanding of normal in a constantly changing network is pretty hard.



You:

explain the following in the simple language "The misuse detection, on the other hand, focuses on the abnormal behaviour directly. The scheme can learn features of attacks based on a labelled training dataset, where both normal and attacked traffic data are marked. Given sufficient labelled data, a misuse-detection design can effectively generate an abstract hypothesis of the boundary between normal and malicious traffic. The hypothesis can then be used to detect attacks for future unknown traffic. Therefore the misuse detection is more feasible and effective than the anomaly detection [2], [3] and has been adopted in real-world systems, such as Google, Qihoo 360 and Aliyun."



ChatGPT:

Certainly! Let's break it down:

1. Misuse Detection:

- This method is different from anomaly detection. Instead of looking for things that are different from the normal routine, it directly focuses on abnormal behavior.
- It learns what attacks look like by using a set of data where both normal and attacked traffic are marked (labeled). This set is like a teacher showing the system examples of both good and bad behavior.
- With enough labeled data, this method can create a general idea (hypothesis) of what normal and malicious (bad) traffic looks like. It basically learns the patterns of known attacks.
- This learned hypothesis can then be used to spot attacks in future traffic that it hasn't seen before. So, it's like saying, "Hey, this looks like what I learned is a bad behavior, so it might be an attack!"

2. Advantages of Misuse Detection:

- The passage suggests that this misuse detection method is more practical and effective compared to anomaly detection. In simpler terms, it's a more realistic and better approach.
- Real-world systems used by big companies like Google, Qihoo 360, and Aliyun (Alibaba Cloud) have adopted this misuse detection method. They trust it to help keep their

networks safe from attacks.

In summary, while one method looks for things that are different from the usual (anomaly detection), the other method (misuse detection) directly learns from examples of what attacks look like and can be more practical and effective in real-world situations.



You:

explain the following in the simple language "However, the existing misuse detection designs still present a high false positive rate, which significantly limits the in-time detection efficiency, incurs large manual scrutiny workload, and potentially degrades the network-wide security. In this paper, we address this issue, and we aim for an improved misuse detection design."



ChatGPT:

Absolutely, let's simplify this:

1. Existing Misuse Detection Designs:

- There are already systems in place that try to directly spot abnormal or harmful behavior in computer networks (misuse detection).
- However, these existing systems have a problem – they often say there's a problem (a positive) when there isn't really one. This is called a "false positive."
- This high rate of false positives creates a few issues:
 - It makes it harder to quickly and accurately detect real threats in time.
 - It leads to a lot of extra work for people who have to manually check and verify whether each alert is a real problem or just a false alarm.
 - It might actually make the overall security of the network worse because it's not effectively distinguishing between real threats and harmless activity.

2. Addressing the Issue:

- The paper is saying that it wants to solve this problem. It aims to create a new and improved version of the existing misuse detection systems.
- The goal is to make a system that gives fewer false alarms, making it more efficient in quickly finding and dealing with real threats. This would reduce the extra work for people and improve the overall security of the entire network.

In a nutshell, the paper is acknowledging that the current systems meant to find abnormal or harmful behavior in computer networks often make mistakes by saying there's a problem when there isn't. The paper sets out to make a better version of these systems that will be more accurate and efficient in identifying real threats.



You:

explain the following the simple language "Our contributions are summarized as follows: • We present a hierarchical deep neural network, LuNet, that is made of multiple levels of combined convolution and recurrent sub-nets; At each level, the input data are learned by both CNN and RNN nets; As the learning progresses from the first level to the last level, the learning granularity becomes increasingly detailed. With such an arrangement, the synergy of CNN and RNN can be effectively exploited for both spatial and temporal feature extractions"



ChatGPT:

Certainly! Let's break this down:

1. Hierarchical Deep Neural Network (LuNet):

- Think of a deep neural network like a smart system that learns things from data. This one is called LuNet.
- "Hierarchical" means it's organized like a pyramid or a tree, with multiple levels or layers.
- Each level has a combination of two types of neural networks: CNN (convolutional neural network) and RNN (recurrent neural network).
- The network is designed to learn data at each level, starting from the general information and becoming more and more detailed as it goes up the levels.

2. Learning Process:

- At each level, LuNet is learning from the input data using both CNN (for recognizing patterns) and RNN (for understanding sequences or changes over time).
- As it moves from the first level to the last level, it becomes more and more detailed in its understanding. It's like going from recognizing basic shapes to understanding intricate details.

3. Synergy of CNN and RNN:

- "Synergy" here means working together to create a better result.
- The paper suggests that by combining the strengths of CNN (good at spatial features, like shapes) and RNN (good at temporal features, like changes over time), LuNet is able to effectively capture both spatial and temporal aspects of the data.
- This collaboration helps LuNet to be really good at understanding both where things are (spatial) and how they change over time (temporal).

In simple terms, the paper introduces a smart system, LuNet, organized in levels. At each level, it learns from data using two types of networks, getting more detailed as it goes. By combining the strengths of these networks, LuNet becomes really good at understanding both shapes and changes over time in the data.



You:

explain the following the simple language "We provide an in-depth analysis and discussion for the configuration of LuNet so that a high learning efficiency can be achieved."



ChatGPT:

Certainly! Let's simplify this:

"We thoroughly examine and talk about how LuNet is set up to make sure it learns really well."

In other words, the paper takes a close look at how they've arranged LuNet (the smart system) to ensure it learns things efficiently and effectively. They want to understand and explain the setup in detail.



You:

explain the following in the simple language "We test our design on two network intrusion datasets, NSLKDD and UNSW-NB15, and we demonstrate that our design offers a higher detection capability (namely, better detection rate and validation accuracy) while maintaining a significantly lower false positive rate when compared to a set of state-of-the-art machine learning based designs."



ChatGPT:

Certainly! Let's break it down:

"We tried out our system on two sets of data related to network attacks, called NSLKDD and UNSW-NB15. In our tests, we show that our system is really good at detecting attacks. Specifically, it has a better ability to spot attacks compared to other advanced systems. Plus, it makes fewer mistakes by having a much lower rate of false alarms. So, it's both more accurate at catching real threats and less likely to raise unnecessary alarms."



You:

explain the following in the simple language "The remainder of this paper is organized as follows: Section II provides a brief background of machine learning for network intrusion detection. The design

of LuNet is detailed in Section III. The threat model in the datasets used for LuNet design is presented in Section IV. The evaluation and comparison of LuNet with other state-of-the-art designs are given in Section V. The paper is concluded in Section VI."



Certainly! Let's break it down:

1. Background (Section II):

- This part gives a quick overview of how machine learning is used to detect network intrusions. It's like a brief introduction to the topic.

2. Design of LuNet (Section III):

- Here, they explain in detail how they built LuNet, the smart system they're using for detecting network intrusions. It's like describing how they put it together and what makes it special.

3. Threat Model in Datasets (Section IV):

- This section talks about the types of threats (potential attacks) they considered when creating and testing LuNet. It's like explaining the bad things they want LuNet to be able to find.

4. Evaluation and Comparison (Section V):

- They discuss how well LuNet performs by comparing it with other advanced designs. It's like a report card, showing how good LuNet is compared to its classmates (other systems).

5. Conclusion (Section VI):

- Finally, they wrap up the paper by summarizing everything they did and what they found. It's like the ending of a story, where they tell you the main points and what it all means.

So, the rest of the paper is organized into these sections, each focusing on different aspects of their work – from the background and building LuNet to testing it, comparing it with others, and finally, concluding their findings.



explain the following in the simple language "II. BACKGROUND AND RELATED WORK Many algorithms have been developed for machine learning. They can be classified into classical machine learning approaches and deep learning approaches. Some of them, relevant to network intrusion detection are briefly discussed below."



Certainly! Let's break it down:

Background and Related Work (Section II):

The authors are now going to talk about the history and other research that has been done on the topic. They're specifically looking at machine learning, which is a way to teach computers to do things without explicitly programming them.

Classical Machine Learning and Deep Learning:

1. Classical Machine Learning Approaches:

- Imagine teaching a computer using traditional methods, like giving it a set of rules or instructions. This is what they mean by classical machine learning. They're saying there are many algorithms (sets of rules) that have been created to teach computers to learn from data.

2. Deep Learning Approaches:

- This is a newer way of teaching computers, inspired by how our brains work. Instead of giving a computer explicit instructions, deep learning lets the computer figure things out by itself through a process of learning from lots and lots of examples.

Relevance to Network Intrusion Detection:

They mention that they're going to focus on the algorithms that are important for detecting network intrusions, which means identifying and stopping unauthorized or harmful activities in computer networks.

In simpler terms, the authors are diving into the background of their topic and want to discuss various ways computers are taught to learn. They're particularly interested in methods related to network intrusion detection.



You:

explain the following in the simple language "A. Classical Machine Learning Approaches Among many machine learning approaches [4], kernel machines and ensemble classifiers are two effective schemes that can be considered for NID. Support Vector Machine (SVM) that has long been used for task classification is a typical example of kernel machines, and Radial Basis Function (RBF) kernel (also called Gaussian kernel) is the most used kernel function [5]. SVM makes the data that cannot be linearly separated in the original space, separable by projecting the data to a higher-dimension feature space. Ensemble classifiers, such as Adaptive Boosting [6], Random Forest [7], are often constructed with multiple weak classifiers to avoid overfitting during training so that a more robust classification function can be achieved."



Absolutely! Let's simplify this:

Classical Machine Learning Approaches:

1. Kernel Machines:

- Think of these as smart tools that help computers make sense of information. They're like special lenses that help us see patterns in data.
- Support Vector Machine (SVM) is a specific tool in this category. It's like a pair of glasses that can make data, which is messy and mixed up, easier to understand.
- The Radial Basis Function (RBF) is a popular way these glasses (kernel function) work. It's like a special lens that helps SVM see things more clearly.

2. Ensemble Classifiers:

- Imagine having a team of different experts, each looking at a problem from a different angle. Ensemble classifiers are like these teams.
- Adaptive Boosting and Random Forest are examples. They're like teams of experts that work together to understand data better.
- These teams are good at preventing mistakes (overfitting) during their learning process. It's like making sure they don't get too focused on one thing and miss the bigger picture.

Relevance to Network Intrusion Detection (NID):

- They're saying these classical methods, like the smart lenses (kernel machines) and expert teams (ensemble classifiers), are useful for teaching computers to recognize and deal with suspicious activities in computer networks. Specifically, they mention SVM as an example, which is great for tasks like sorting things into categories.

In simple terms, the authors are talking about some smart tools (kernel machines) and expert teams (ensemble classifiers) that are helpful in teaching computers to understand and deal with strange or harmful activities in computer networks. They give examples like SVM and explain how these tools and teams work to make sense of data.



explain the following in the simple language "However, both kernel machines and ensemble classifiers are not scalable to a large data set. Their validation accuracy rarely scales with the size of training data [8]. Given a high volume of training data available from the large scale network, using these traditional machine learning approaches to train the intrusion detection system is not efficient. Furthermore, the traditional approaches learn input data only based on a given set of features; they cannot generalize features from raw data, and their learning efficiency highly relies on the features specified."



Certainly! Let's break this down:

Limitations of Kernel Machines and Ensemble Classifiers:

1. Not Scalable to Large Data Sets:

- Imagine you have a lot of information, like a huge library of books. Kernel machines and ensemble classifiers are like tools used to understand these books.
- Unfortunately, these tools don't work well when you have an enormous amount of information to go through. It's like trying to use a small magnifying glass to read a whole library of books.

2. Validation Accuracy Doesn't Keep Up:

- When you have a lot of data to train these tools, they often struggle to give accurate results. It's like trying to study for a really big exam, and your performance doesn't get better as you study more.
- In the context of network intrusion detection, where there's a huge amount of data, using these tools becomes less effective.

3. Inefficiency with Large-Scale Networks:

- If you have a massive network with tons of information flowing through, using these traditional tools to teach a system to recognize and stop intrusions isn't very efficient. It's like trying to use an old and slow computer to handle a lot of data.

4. Learning Limited to Specified Features:

- These tools learn from data based on a specific set of features, like certain characteristics of the data. They can't understand information in a broader sense. It's like teaching someone to recognize only specific types of cars but not all cars in general.
- Additionally, their ability to learn efficiently depends heavily on the features you tell them to focus on.

In simpler terms, the paragraph is saying that when you have a huge amount of data, the traditional tools they mentioned earlier (kernel machines and ensemble classifiers) don't work very well. They struggle to give accurate results, and using them becomes inefficient, especially in large-scale network situations. Additionally, they can only learn specific features that you tell them to focus on, limiting their ability to understand broader aspects of the data.



You:

explain the below in the simple language "B. Deep Learning Approaches Deep learning organizes “learning algorithms” in layers in the form of “artificial neural network” that can learn and make intelligent decisions on its own. Multi-Layer Perceptron (MLP) [9] is an early class of feedforward deep neural network that utilizes the backpropagation algorithm to minimize the error rate during training.

MLP was initially used to solve complex approximate problems for speech recognition, image recognition, and machine translation."



ChatGPT:

Absolutely, let's break it down:

Deep Learning Approaches:

1. What is Deep Learning:

- Deep learning is like a smart system that learns things on its own. It's organized in layers, kind of like building blocks. Each layer helps it understand more complex things.

2. Artificial Neural Network:

- Imagine it's like a virtual brain made of many connected parts. These connected parts are called neurons, and they work together to understand and make decisions about information.

3. Multi-Layer Perceptron (MLP):

- This is an early type of deep learning system. Think of it like a specific version of this smart system.
- It uses something called a backpropagation algorithm, which is like a method to correct mistakes and get better over time.
- MLP was first used to solve complicated problems, like understanding human speech, recognizing images, and translating languages with the help of machines.

In simpler terms, deep learning is a way to teach computers to be smart on their own. It's like a virtual brain made of connected parts (neurons), organized in layers. The Multi-Layer Perceptron (MLP) is a specific type of this system that was used to solve tricky problems, like recognizing speech, understanding images, and translating languages with machines.



You:

explain the following in the simple language "The real flourish and practical breakthrough of deep learning stems from two popular deep learning algorithms: the convolutional neural network (CNN) and recurrent neural network (RNN). CNN can automatically extract features of raw data and has gained great successes for image recognition [10], [11], [12]. However, the feature map generated by CNN often manifests the spatial relations of data. CNN does not work very well for the data of long-range dependency"



Certainly! Let's break this down:

Real Breakthroughs in Deep Learning:

1. Convolutional Neural Network (CNN):

- Think of CNN as a smart tool that can automatically figure out important features from raw data. It's really good at recognizing patterns and has been super successful in recognizing images.
- For example, if you show it a bunch of pictures of cats and dogs, it can learn to tell the difference between them.

2. Recurrent Neural Network (RNN):

- This is another smart tool that's great at understanding data sequences, like things that happen over time. It's like a system that remembers what happened before to make sense of what's happening now.
- For instance, if you're predicting the next word in a sentence, an RNN can understand the context by looking at the words that came before.

Limitation of CNN:

- While CNN is fantastic at recognizing patterns in data, especially images, it has a little issue. The features it extracts often show how things are arranged in space but might not capture long-range relationships well.
- Imagine you're trying to understand a story in a book. CNN might be really good at understanding individual sentences or paragraphs, but it might miss the broader connections between events that happen far apart in the story.

In simpler terms, the real breakthroughs in making computers smart come from two tools: CNN, which is great at recognizing patterns in things like images, and RNN, which is awesome at understanding things that happen over time. However, CNN has a bit of a weakness – it might not be as good at capturing the connections between things that are far apart in data.



explain below in the simple language "RNN, on the other hand, has an ability to extract the temporal features from the input data. Long short-term memory (LSTM) is a popular RNN [13]. It keeps a trend of the longterm relationship in the sequential data while being able to drop out short-lived temporal noises."



Certainly! Let's break this down:

1. Recurrent Neural Network (RNN):

- RNN is another smart tool that's really good at understanding things happening over time. It can figure out patterns and relationships in data sequences.
- For example, if you're trying to understand a story, RNN can help you connect events that occur one after another.

2. Long Short-Term Memory (LSTM):

- LSTM is a specific type of RNN, and it's like a smarter version. It's popular because it's excellent at keeping track of long-term patterns in sequences of data.
- Imagine you're reading a book. LSTM is like a reader who remembers important details from the beginning of the story even as new events unfold. It's good at not forgetting important stuff.

3. Ability to Extract Temporal Features:

- Temporal features are details related to how things change or evolve over time. RNN, especially with the help of LSTM, is really good at understanding these temporal features.
- Going back to the story example, it's like understanding not just what's happening now, but how it connects to what happened before.

4. Handling Temporal Noises:

- Temporal noises are like distractions or unimportant details that can mess up your understanding. LSTM is good at filtering out these short-lived distractions so that it focuses on the essential, long-term aspects of the data.

In simpler terms, RNN, especially with the help of LSTM, is great at understanding things that happen over time. It can pick up on patterns and relationships in sequential data, and LSTM, in particular, is good at remembering important details over the long term while ignoring short-lived distractions. It's like having a smart reader who doesn't forget the main plot of a story even when there are small, unimportant details along the way.



You:

explain below in the simple language "Recently, a hierarchical convolutional and recurrent neural network, HAST-IDS [14] has been used to learn spatial and temporal features from the network traffic data"



ChatGPT:

Certainly! Let's simplify this:

New System: HAST-IDS

1. Hierarchical Convolutional and Recurrent Neural Network (HAST-IDS):

- There's a new smart system called HAST-IDS.
- It's like a combination of two tools: one that's good at understanding spatial features (like patterns in space), and another that's great at understanding temporal features (like things changing over time).
- This system is used to learn important details from the data that flows through computer networks.

2. What it Does:

- HAST-IDS is designed to look at the network traffic data, which is like all the information flowing through the internet or a computer network.
- It's not just looking at the data as a whole; it's breaking it down into spatial features (like where things are) and temporal features (how things change over time).

In simpler terms, there's a new smart system called HAST-IDS that's really good at understanding both where things are and how they change over time in the data that moves through computer networks. It's like having a detective that can figure out patterns and relationships in the information flowing through the internet.



You:

explain below in the simple language "The LuNet proposed in this paper is also a hierarchical network. Our design is similar to HAST-IDS in that both utilize CNN and RNN for spatial-temporal feature extraction. However, there are some major differences: 1) HAST-IDS stacks all RNN layers after CNN layers, while LuNet has a hierarchy of combined CNN and RNN layers. 2) In HAST-IDS, because the CNN hierarchy is placed before RNN, the deep CNN may drop out the temporal information embedded in the raw input data, which makes RNN learning ineffective. LuNet, on the other hand, synchronizes both the CNN learning and the RNN learning into multiple steps with the learning granularity being gradually increased from coarse to fine; Therefore, both spatial and temporal features can be adequately captured. 3) We apply batch normalization between CNN and RNN so that the learning efficiency and accuracy can be further improved. The design of LuNet is presented in the next section."



ChatGPT:

Absolutely, let's break this down:

LuNet: A Smart Network Proposed in the Paper

1. Hierarchical Network:

- The LuNet introduced in this paper is like a smart system organized in levels, similar to HAST-IDS.

- Both LuNet and HAST-IDS use two tools, CNN and RNN, to understand both spatial (patterns in space) and temporal (changes over time) features in the data that moves through computer networks.

2. Differences with HAST-IDS:

- 1st Difference:** HAST-IDS puts all the RNN layers after the CNN layers. LuNet, however, has a hierarchy where CNN and RNN layers are combined at each level.
- 2nd Difference:** In HAST-IDS, the deep CNN layers might lose some temporal information from the raw input data before RNN gets to learn. LuNet avoids this issue by synchronizing both CNN and RNN learning in multiple steps, gradually increasing the level of detail. This way, it captures both spatial and temporal features effectively.
- 3rd Difference:** LuNet uses something called batch normalization between CNN and RNN to improve learning efficiency and accuracy.

3. Improvements in LuNet:

- LuNet is designed to make sure it learns both spatial and temporal features effectively by synchronizing the learning process of CNN and RNN in a step-by-step manner.
- The addition of batch normalization between CNN and RNN is like adding a tool to make the learning process more efficient and accurate.

In simpler terms, the LuNet proposed in this paper is a smart system similar to HAST-IDS but with some improvements. It's designed to effectively learn both where things are and how they change over time in the data flowing through computer networks. The way it organizes the learning process and adds batch normalization helps it do this more efficiently and accurately. The detailed design of LuNet is explained in the next section of the paper.



You:

explain the following in the simple language "III. LUNET A. Overview Structure As stated above, CNN targets on spatial features while RNN aims for temporal features. The existing design HAST-IDS simply structures CNN and RNN in tandem, as illustrated in Fig.1(a). When the learning progresses along the multiple levels in the CNN hierarchy, the information extracted becomes more spatial oriented. The temporal features may be lost by the CNN hierarchy, which significantly limits the learning effectiveness of the following RNN (LSTM)."



ChatGPT:

Absolutely, let's simplify this:

Section III: LuNet

A. Overview Structure

1. What is LuNet Doing:

- LuNet is like a smart system designed to understand both where things are (spatial features) and how they change over time (temporal features) in data that flows through computer networks.

2. Spatial and Temporal Features:

- CNN is like a tool that's really good at understanding where things are, like recognizing patterns in space.
- RNN, on the other hand, is great at understanding how things change over time, like recognizing sequences of events.

3. Issues with Existing Design (HAST-IDS):

- HAST-IDS, which is another system, puts CNN and RNN side by side. As you go through the levels in the CNN part, the information it picks up becomes more focused on where things are in space.
- However, this might cause a problem. The part of the information that talks about how things change over time (temporal features) might get lost as you move through the levels of the CNN.
- This loss of temporal features can make it harder for the RNN (which is good at understanding temporal features) to learn effectively.

In simpler terms, LuNet is a smart system that wants to understand both where things are and how they change over time in computer network data. The issue it's addressing is that in another existing design (HAST-IDS), the tool used for recognizing patterns in space (CNN) might accidentally lose information about how things change over time, making it difficult for the system to learn effectively.



You:

explain the following in the simple language "Rather than allow CNN to learn to its full extent first, in LuNet, we mingle the CNN and RNN subnets and synchronize both the CNN learning and the RNN learning into multiple steps and each step is performed by a combined CNN and RNN block, or LuNet block, as illustrated in Fig.1(b). Since CNN is able to extract high-level features from a large amount of data, we place CNN before RNN at each level."



ChatGPT:

Certainly! Let's simplify this:

How LuNet Works:

1. Combining CNN and RNN:

- Instead of letting CNN (the tool for recognizing patterns in space) learn everything on its own first, LuNet does something different.
- It mixes up the learning of CNN and RNN together. They work as a team in multiple steps.
- At each step, both CNN and RNN work together to understand the data. This combined process is called a "LuNet block."

2. Why CNN Goes First:

- CNN is really good at figuring out patterns from a lot of data. So, in LuNet, they make CNN go first in each step.
- This is like having a detective look at a big picture and pick out the important details before another detective (RNN) looks at how things change over time.

In simpler terms, LuNet works by making CNN and RNN learn together in multiple steps. Each step involves a combined effort, and they let CNN go first because it's great at recognizing patterns in a lot of data. It's like having two detectives – one looks at the big picture, and the other looks at how things change over time.



You:

explain the following in the simple language "The learning starts from the first step on coarse-grain learning, hence the CNN output will still retain temporal information that will then be captured by RNN. The learning granularity becomes detailed as the data processing flows to the next step; but at each level, both CNN and RNN learn input on the same granularity. In such a way, CNN and RNN can learn to their full capacity without much interference with each other. Given the overall structure of LuNet, the effectiveness of its learning is closely related to the hidden layers design, which is discussed in the next sub section."



ChatGPT:

Certainly! Let's simplify this:

How LuNet Learns:

1. Starts with Coarse Learning:

- LuNet begins its learning process with a big-picture view, called "coarse-grain learning." This means it looks at the overall patterns in the data first.
- In this initial step, CNN is in charge, and it still keeps information about how things change over time (temporal information).

2. Detailed Learning as it Progresses:

- As LuNet moves to the next steps, it becomes more detailed in understanding the data. However, at each level, both CNN and RNN learn at the same level of detail (granularity).
- Imagine zooming in on a picture – you start with the general view and gradually focus on the smaller details.

3. CNN and RNN Learning Together:

- LuNet is smart about letting both CNN and RNN learn at their full capacity without getting in each other's way.
- By starting with coarse learning and gradually becoming more detailed, both tools can do their jobs effectively without causing problems for each other.

4. Hidden Layers Design Matters:

- The success of LuNet's learning depends on how it organizes its hidden layers. These are like the internal workings that help it understand the data.
- The paper will discuss this hidden layers design in the next section.

In simpler terms, LuNet starts by looking at the big picture, gradually zooms in on details, and lets both tools (CNN and RNN) learn together without causing confusion. The way it organizes its hidden layers is crucial for its learning process, and this will be explained in more detail in the next part of the paper.



You:

explain the following in the simple language "B. Hidden Layers Design We measure the learning granularity in terms of the number of filters/cells used in the CNN/RNN network. The smaller the filter/cell number, the larger the granularity. Fig.2 shows a LuNet with three CNN+RNN levels. The filter/cell number increases from 64 for the first level to 256 for the last level."



ChatGPT:

Absolutely, let's break it down:

Hidden Layers Design in LuNet:

1. What Are Hidden Layers:

- Hidden layers are like the internal parts of LuNet that help it understand data. They're not the input data, and they're not the final results, but they play a crucial role in the learning process.

2. Measuring Learning Granularity:

- Learning granularity means how much detail LuNet looks at when understanding data. It's measured in terms of the number of filters/cells used in the CNN/RNN network.

3. Filter/Cell Number and Granularity:

- When they talk about the "filter/cell number," they mean how many of these internal filters or cells LuNet uses.
- The smaller the number of filters/cells, the larger the granularity. It's like saying, "If there are fewer of these internal parts, LuNet is looking at the data more broadly."

4. LuNet Structure:

- Fig. 2 is a diagram that shows LuNet with three levels of learning. Each level is a combination of CNN and RNN.
- In the first level, LuNet uses 64 filters/cells. As it progresses to the next levels, it increases the number to 256 filters/cells for the last level.

In simpler terms, the hidden layers in LuNet are like its internal machinery for understanding data. The learning granularity, or how detailed LuNet looks at the data, is measured by the number of filters/cells

it uses. If there are fewer filters/cells, LuNet is looking at the data more broadly. The diagram (Fig. 2) shows that in LuNet's three levels of learning, it starts with 64 filters/cells and increases it to 256 filters/cells in the final level.



You:

explain the following in the simple language "The design considerations for each layer in a LuNet block and the final four processing layers for the learning outputs, as highlighted in shaded boxes in Fig.2, are given below. 1) Convolutional Neural Network (CNN): CNN mainly consists of two operations: convolution and pooling. Convolution transforms input data, through a set of filters or kernels, to an output that highlights the features of the input data, hence the output is usually called feature map. The convolution output is further processed by an activation function and then down-sampled by pooling to trim off irrelevant data. Pooling also helps to remove glitches in the data to improve the learning for the following layers [15] - [16]."



ChatGPT:

Certainly! Let's break this down:

Design Considerations in LuNet for Each Layer:

1. LuNet Block Layers:

- LuNet is like a detective, and it has these building blocks called "LuNet blocks." Each block has its own layers for understanding data.

2. Processing Layers in a LuNet Block (Highlighted in Fig.2):

- Fig. 2 shows the different layers in a LuNet block. The final four layers are especially important for getting the learning results.

3. Layer 1: Convolutional Neural Network (CNN):

- CNN does two main things: convolution and pooling.
- **Convolution:** This is like a special filter that looks at the input data and highlights important features, creating what's called a "feature map."
- **Activation Function:** After convolution, an activation function further processes the output. It's like giving a signal to the system that says, "Here are the important features."
- **Pooling:** The output is then down-sampled (made smaller) through pooling. It's like trimming off unimportant details. This helps improve learning for the next layers.

4. Purpose of CNN Operations:

- The whole point of these CNN operations is to transform the input data, highlight important features, and remove unnecessary details. It's like LuNet's way of saying, "Let's focus on what matters and ignore the rest."

In simpler terms, LuNet uses these building blocks called "LuNet blocks" to process data. The important layers in each block, especially the final four, work to transform the data, highlight important features, and get rid of unnecessary details. It's like LuNet's way of preparing the data for better learning.



You:

explain the following in the simple language "CNN learns the input data by adjusting the filters automatically through rounds and rounds of learning processes so that its output feature map can effectively represent the raw input data. Since the network packet is presented in a 1D format, we use 1D convolution1 in LuNet, as illustrated in Formula (1) that specifies the operation to the input vector g with a filter f of size m . $(f \square g)(i) = \sum_{j=1}^m g(j) \cdot f(i - j + m/2)$, (1) where i is the position of different values in the sequence data. Because the rectified linear unit (ReLU): $f(z) = \max(0, z)$ is good for fast learning convergence, we therefore, choose it as the activation function. We also use the max pooling operation, as commonly applied in other existing designs [17]."



ChatGPT:

Certainly! Let's simplify this:

Learning with Convolutional Neural Network (CNN):

1. What CNN Does:

- CNN is like a learning tool that adjusts itself automatically to understand input data better. It does this through many rounds of learning.
- Its job is to transform the input data in a way that its output, called a "feature map," effectively represents the raw input data. It's like highlighting the important parts.

2. 1D Convolution in LuNet:

- LuNet uses 1D convolution because the data it's dealing with (like network packets) is presented in a 1D format, like a sequence.
- The formula (1) in the paper describes how this 1D convolution works. It's a way of processing the input data with a filter to create the feature map.
- The formula uses positions (i) to figure out how the values in the sequence data interact with the filter.

3. Activation Function (ReLU):

- After convolution, LuNet uses something called ReLU (Rectified Linear Unit) as an activation function. It's a fancy term for a function that helps the system learn faster.
- ReLU works like this: if the input value is positive, it stays the same; if it's negative, it becomes zero. This kind of function is good for quick learning.

4. Max Pooling Operation:

- LuNet also uses a process called max pooling. It's like looking at the most important information and keeping that, while throwing away less important details.
- This operation is commonly used in other designs too because it helps improve learning.

In simpler terms, CNN is LuNet's learning tool that adjusts itself to understand input data better. LuNet uses a 1D convolution for its 1D format data, ReLU to learn faster, and max pooling to focus on the most important information. It's like LuNet's way of processing and simplifying the data for effective learning.



You:

explain the following in the simple language "2) Batch Normalization: One problem with using the deep neural network is that the input value range dynamical changes from layer to layer during training, which is also known as covariance shift. The covariance shift causes the learning efficiency of one layer dependent on other layers, making the learning outcome unstable. Furthermore, because of the covariance shift, the learning rate is likely restricted to a low value to ensure data in different input ranges to be effectively learned, which slows down the learning speed. Batch normalization can be used to address this issue."



ChatGPT:

Certainly! Let's break this down:

Batch Normalization: Making Learning Smoother

1. Problem with Deep Neural Networks:

- When we use deep neural networks (like LuNet), there's a problem. The input values change a lot from one layer to the next during training. This changing range is called "covariance shift."

2. Covariance Shift:

- Covariance shift causes a bit of trouble. It makes the learning efficiency of one layer depend on the other layers, making the learning outcome a bit shaky or unstable.

3. Learning Rate Challenge:

- Because of this covariance shift, we often have to use a low learning rate. It's like going slowly to make sure the system can effectively learn from data in different input ranges. This slow learning speed can be a problem.

4. Batch Normalization Solution:

- Batch normalization is like a tool to fix this issue.
- It helps in keeping the input value range consistent across layers during training. This way, each layer doesn't have to rely too much on the others, making the learning process more stable.

- With batch normalization, we can use a higher learning rate, making the learning faster. It's like speeding up the learning process without compromising on stability.

In simpler terms, when we use deep neural networks, the input values change a lot from layer to layer, causing some instability in learning. Batch normalization is like a fix for this. It ensures that the input values stay consistent across layers, making the learning smoother and faster. It's like having a tool that helps the system learn more efficiently.



You:

explain the following in the simple language "Normalization scales data to the unit norm in the input layer and has been used to accelerate training deep neural network for image recognition [18]. We use batch normalization to adjust the CNN output for RNN in a LuNet block. The normalization subtracts the batch mean from each data and divides the result by the batch standard deviation, as given in Formula (2). $\hat{X} = (x - \mu_B) / \sqrt{\text{square}(\delta_B) + \epsilon}$ "



ChatGPT:

Certainly! Let's simplify this:

Batch Normalization in LuNet: Making Data Uniform

1. What is Normalization:

- Normalization is like a tool that makes sure data is in a standard or uniform form.
- It's been used before to speed up training in deep neural networks used for recognizing images.

2. Batch Normalization in LuNet:

- LuNet uses something called "batch normalization" to adjust the output from CNN for RNN in each LuNet block.
- This means it makes sure the data coming out of CNN is in a consistent and standard form before RNN processes it.

3. How Batch Normalization Works:

- The normalization process subtracts the average (mean) value of the batch from each piece of data.
- Then, it divides the result by the standard deviation of the batch. This standard deviation is like a measure of how spread out the data is.
- The formula (2) in the paper describes this process. It looks a bit complex, but it's essentially making sure that all the data is in a similar format, so RNN can work with it effectively.

In simpler terms, batch normalization in LuNet is like a tool to make sure the data coming out of CNN is in a consistent and standard form before RNN deals with it. It's like preparing the data to be more

uniform, so the system can understand and learn from it better. The formula is just a way of mathematically expressing this process.



You:

explain the following in the simple language "1One issue involved in convolution multiplication is to maintain the kernel size as same as the input, for which two typical schemes can be used: zero or no-zero padding. Since both schemes do not generate much difference to the learning outcome, here we choose no-zero padding for simplicity where x is a value in the input batch, and μ_B and δ_B are, respectively, the batch mean and variance. The \square is an ignorable value, just to ensure the denominator in the formula non-zero. Based on the normalized \hat{x} , the normalization produces the output y as given in Formula (3), where the γ and β will be trained in the learning process for a better learning outcome. $y^{\hat{}} = \gamma \hat{x} + \beta$."



ChatGPT:

Certainly! Let's break this down:

Convolution Multiplication and Padding in LuNet: Keeping Things Consistent

1. Convolution Multiplication Issue:

- When we multiply things through convolution (a specific mathematical operation), we need to make sure the size of what we're multiplying stays the same as the input size.
- To do this, we can use two ways: zero-padding or no-zero padding. For simplicity, LuNet chooses no-zero padding in this case.

2. What is No-Zero Padding:

- No-zero padding means we add some extra values around the input data to keep the size consistent. The formula uses a variable (ϵ) to make sure the denominator (the bottom part of a fraction) doesn't become zero.

3. Normalized Value (\hat{x}):

- The formula involves a value called \hat{x} . This is the input data normalized using the batch mean (μ_B) and batch standard deviation (δ_B).

4. Normalized Output (\hat{y}):

- The normalization process produces an output \hat{y} using another formula. This formula involves γ and β , which are values that will be adjusted during the learning process to improve outcomes.
- γ and β are like adjustable knobs that LuNet turns during learning to get better results.

In simpler terms, when LuNet multiplies things through convolution, it wants to make sure the size stays the same as the input. It uses a method called no-zero padding. Then, it normalizes the data using batch mean and batch standard deviation to keep things consistent. The formula involves

variables like ϵ to make sure everything works smoothly. The output is adjusted further during learning using values γ and β to get better results. It's like LuNet's way of ensuring consistency and improvement in the learning process.



You:

explain the following in the simple language "3) Long Short Term Memory (LSTM): Different from CNN that learns information on an individual-data-record basis, RNN can establish the relationship between data records by feeding back what has been learned from the previous learning to the current learning, and hence can capture the temporal features in the input data"



ChatGPT:

Absolutely! Let's simplify this:

Long Short Term Memory (LSTM) in LuNet: Capturing Relationships Over Time

1. Difference Between CNN and RNN:

- CNN and RNN are both tools in LuNet, but they work a bit differently.
- CNN focuses on learning from each piece of data individually.
- RNN, on the other hand, does something special – it learns by remembering what it learned before and uses that knowledge in the current learning.

2. Role of RNN - Capturing Temporal Features:

- RNN is like a memory system. It doesn't just look at one piece of data; it looks at the relationship between pieces of data over time.
- This ability to remember and connect information from the past to what's happening now helps RNN capture "temporal features." Temporal features are like patterns or changes over time in the input data.

In simpler terms, while CNN looks at each piece of data by itself, RNN (specifically, using LSTM in LuNet) is like a memory system. It doesn't forget what it learned before and uses that knowledge to understand how things change over time. This is crucial for capturing temporal features or patterns in the data that evolve over a sequence.



You:

explain the following in the simple language "However, the simple feedback used in the traditional RNN may have a learning error accumulated in the long dependency. The accumulated errors may become large enough to invalidate the final learning outcome. LSTM (Long Short-Term Memory), a gated recurrent neural network, mitigates such a problem. It controls the feedback with a set of gate

functions such that the short-lived errors are eventually dropped out and only persistent features are retained. Therefore, we use LSTM for RNN"



ChatGPT:

Certainly! Let's break this down:

Long Short-Term Memory (LSTM) in LuNet: Avoiding Accumulated Errors

1. Problem with Traditional RNN Feedback:

- In traditional RNN, when it learns over a long sequence of data, there can be a problem.
- The feedback mechanism (remembering what it learned before) might accumulate errors over time, especially when dealing with long sequences.

2. Issue of Accumulated Errors:

- As RNN learns more and more, errors can add up. These accumulated errors might become so large that they mess up the final learning outcome. It's like small mistakes building up and causing a big problem.

3. LSTM as a Solution:

- LSTM (Long Short-Term Memory) is like a smarter version of RNN. It's a "gated recurrent neural network."
- Instead of blindly using feedback, LSTM has a set of "gate functions" that control the information being remembered.
- These gate functions are like filters that decide what to keep and what to drop. They make sure only the important and persistent features are retained, while short-lived errors are dropped out.

4. Why LuNet Uses LSTM:

- Because LSTM manages feedback with these gate functions, it's better at handling long dependencies in data. LuNet chooses LSTM for RNN to avoid the problem of accumulated errors over time.

In simpler terms, when dealing with long sequences of data, traditional RNN might make mistakes that add up over time. LSTM, being a smart version of RNN, uses gate functions to control and filter information. This way, it keeps the important stuff and drops out errors that might accumulate. LuNet uses LSTM to avoid the problem of mistakes piling up during learning.



You:

explain the following in the simple language "For a brief description of LSTM operation, we create a high-level data processing diagram of LSTM 2. LSTM can be abstracted as a connection of four sub-networks (denoted as p-net, g-net, f-net and q-net in the diagram), a set of control gates, and a

memory component. The input and output values in the diagram are vectors of the same size determined by the input $x(t)$. The state, $s(t)$, saved in the memory, serves as the feedback to the current learning. All the sub nets in LSTM have a similar structure, as specified in Formula (4). $b + U \times x(t) + W \times h(t - 1)$, (4) where $x(t)$, $h(t - 1)$, b , U and W are, respectively, current input, previous output, bias, weight matrix for the current input, and recurrent weight matrix for the previous output. Each of the four nets have a different b , U , and W ."



Certainly! Let's simplify this:

How LSTM (Long Short-Term Memory) Works: A Simple Overview

1. LSTM Structure:

- Think of LSTM like a small team of workers handling information. It's organized into four mini-teams (sub-networks) and a few special components.

2. The Mini-Teams in LSTM:

- These mini-teams are labeled as p-net, g-net, f-net, and q-net in the diagram.
- Each mini-team has a specific job, and they work together to make sense of the input data.

3. Control Gates:

- There's something called "control gates" – these are like traffic lights or filters. They decide what information should be let through and what should be kept in check.

4. Memory Component:

- There's a memory component, which is like a notepad where important information is written down. This memory helps in feedback for the current learning.

5. Input and Output:

- The input and output values in this system are like packets of information, and they have a specific size determined by the input data at a particular moment ($x(t)$).

6. Structure of Each Mini-Team:

- Each mini-team has a specific way of handling information, described by Formula (4).
- It involves the current input ($x(t)$), the previous output ($h(t-1)$), a bias (b), a weight matrix for the current input (U), and a recurrent weight matrix for the previous output (W).
- All these elements together help each mini-team process information in its unique way.

7. Why Four Mini-Teams:

- Each of the four mini-teams (p-net, g-net, f-net, q-net) has a slightly different setup with its own set of b , U , and W . This diversity helps in handling different aspects of the input data.

In simpler terms, think of LSTM like a team with four different roles and a few filters (control gates). Each role has a specific way of processing information, and they work together to understand and

remember things over time. The diversity in their setups helps them handle various aspects of the input data effectively. It's like a small group of workers each doing their job to make sure everything is well-organized and remembered properly.



You:

explain the following in the simple language "The outputs from the sub nets ($p(t)$, $g(t)$, $f(t)$ and $q(t)$) are then used, through two types of controlling gates (σ and \tanh) to determine the feedback $s(t)$ from the previous learning and the current output $h(t)$, as given in Formulas (5) and (6), respectively. $s(t) = \sigma(f(t)) * s(t - 1) + \sigma(p(t)) * \tanh g(t)$, (5) $h(t) = \tanh s(t) * \sigma(q(t))$. (6) LSTM learns the inputs by adjusting the weights in those nets and the σ value such that the temporal features between the input data can be effectively generated in the output."



ChatGPT:

Certainly! Let's simplify this:

How LSTM (Long Short-Term Memory) Decides What to Remember:

1. Outputs from Mini-Teams:

- Remember those mini-teams (p-net, g-net, f-net, q-net)? They each produce some outputs labeled as $p(t)$, $g(t)$, $f(t)$, and $q(t)$.

2. Controlling Gates - The Decision Makers:

- Now, there are two decision-making gates, let's call them "traffic lights." One gate uses σ (sigma) and the other uses \tanh . These gates decide what information should be remembered and what should be passed on.

3. Feedback Mechanism ($s(t)$):

- The feedback mechanism, represented by $s(t)$, is like remembering important things from the past learning. This feedback is determined by two formulas:
 - One part is controlled by the σ gate using $f(t)$ and $s(t-1)$ from the previous learning.
 - The other part is controlled by the σ gate using $p(t)$ and $g(t)$.

4. Current Output ($h(t)$):

- The current output, represented by $h(t)$, is determined by combining the remembered information ($s(t)$) with another gate controlled by \tanh and $q(t)$.

5. Learning Process - Adjusting Weights:

- LSTM learns and improves by adjusting the weights in those mini-teams and the values in the σ gate.
- The goal is to adjust things in such a way that the output effectively captures the temporal features (patterns over time) in the input data.

In simpler terms, think of LSTM as a decision-maker. It uses the outputs from its mini-teams, guided by two types of gates (σ and tanh), to decide what to remember from the past learning ($s(t)$) and what to output as the current result ($h(t)$). The adjustment of weights and values is like fine-tuning, making sure it learns the important patterns in the input data effectively.



You:

explain the following in the simple language "4) Dimension Reshape: Since in LuNet, the learning granularity changes from one CNN+RNN level to another, the output size of one level is different from the input size expected at the next level. We, therefore, add a layer to reshape the data for the next LuNet block"



ChatGPT:

Certainly! Let's simplify this:

Dimension Reshape in LuNet: Adjusting Data for Learning

1. Changing Learning Granularity:

- LuNet is designed to learn in different levels, and as it moves from one level to another, the granularity of what it's learning changes.

2. Output Size Mismatch:

- The data size (think of it like the amount of information LuNet is handling) at the end of one level might be different from what the next level expects as input.

3. Solution - Dimension Reshape:

- To fix this difference in data size, LuNet adds a special layer.
- This layer is like a magic trick – it reshapes or adjusts the data so that it fits perfectly for the next level of learning.

In simpler terms, as LuNet moves through different learning levels, the amount of information it deals with changes. To make sure everything fits smoothly, LuNet adds a layer that adjusts or reshapes the data to be just right for the next level of learning. It's like preparing the data so that LuNet can continue to understand and learn effectively.



You:

explain the following in the simple language "5) Overfitting Prevention: One typical problem when learning big data using a deep neural network is overfitting – namely, the network has learned the

training data too well, which restricts its ability to identify variants in new samples. This problem can be handled by Dropout [19]. Dropout randomly removes some connections from the deep neural network to reduce overfitting. We add the dropout layer with a default rate value of 0.5 after the CNN+RNN hierarchy in LuNet."



Certainly! Let's simplify this:

Preventing Overfitting in LuNet: A Safety Measure

1. The Problem - Overfitting:

- When LuNet is learning from a large amount of data, there's a common issue called overfitting.
- Overfitting happens when LuNet gets a bit too obsessed with the training data. It learns it so well that it struggles to recognize variations in new, unseen data.

2. The Solution - Dropout:

- To prevent this overfitting problem, LuNet uses something called Dropout.
- Dropout is like a safety measure. It randomly disconnects or "drops out" some connections in the deep neural network during the learning process.

3. Why Dropout:

- By dropping out some connections, it's like forcing LuNet to be more flexible. It prevents LuNet from becoming too fixated on the specifics of the training data.
- This flexibility helps LuNet become better at recognizing new patterns in data it hasn't seen before.

4. Where Dropout is Added:

- LuNet adds the Dropout layer after the CNN+RNN hierarchy. It's like a checkpoint, making sure LuNet doesn't get too attached to the training data.

5. Default Rate Value:

- The rate value of 0.5 means that, during training, each connection has a 50% chance of being dropped out. This randomness helps in keeping the learning process more adaptable.

In simpler terms, when LuNet is learning a lot of data, there's a risk it becomes too focused on the training details. To avoid this, LuNet uses a safety measure called Dropout. This is like introducing randomness by disconnecting some connections in its learning process. It ensures LuNet doesn't get too fixated and can identify new patterns in unseen data effectively. The Dropout layer is like a helpful guard to keep LuNet's learning more balanced.



You:
Replying to:
fixated
give me simple meaning



ChatGPT:
Certainly! In the context of LuNet preventing overfitting:

Fixated:

- Imagine you're learning how to recognize different types of animals. If you become too fixated on the details of just one animal, like a cat, you might struggle to identify variations in other animals like dogs or birds. In the same way, LuNet being fixated on training data means it learns it too well, making it less flexible when faced with new, unseen data.
-



You:
explain the following in the simple language "6) Final Layers: Finally, an extra convolution layer and a global average pooling layer are used to extract further spatial-temporal features learned through the LuNet blocks. The final learning output is generated by the last layer, a fully-connected layer."



ChatGPT:
Certainly! Let's simplify this:

Final Touches in LuNet: Wrapping Up the Learning Process

1. Extra Convolution Layer and Global Average Pooling Layer:

- After LuNet has gone through all its learning blocks, it does a little more to make sure it catches all the important features.
- It adds an extra convolution layer, which is like a last sweep to gather any remaining spatial-temporal features (patterns in space and time).
- Then, there's a global average pooling layer. Think of this as a summarizer; it takes the important information and averages it out to create a condensed representation.

2. Generating the Final Output:

- The very last layer is a fully-connected layer. This layer takes all the processed information and generates the final learning output.
- It's like LuNet's way of saying, "After all that learning, here's the result!"

In simpler terms, after LuNet has learned a bunch of things through its blocks, it does a final check. It adds a little more processing to make sure it didn't miss anything important. Then, it uses a special layer to generate the final result, like saying, "This is what I've learned!" It's like putting the finishing touches on the learning process.



You:

explain the following in the simple language "IV. DATASETS AND THREAT MODEL The evaluation of a neural network design is closely related to the dataset used. Many datasets collected for NID contain significant amount of redundant data [20] - [21], which makes evaluation results unreliable [5] - [6] - [7] - [14] - [22]. To ensure the effectiveness of the evaluation, we select two non-redundant datasets: NSL-KDD and UNSW-NB15 in our investigation."



ChatGPT:

Absolutely! Let's break this down:

Datasets and How LuNet is Tested:

1. Importance of Datasets:

- When testing how well LuNet works, the choice of the dataset (a collection of data used for testing) is crucial.
- Some datasets used for Network Intrusion Detection (NID) have a lot of repeated or unnecessary information, making it hard to trust the results of the evaluation.

2. Problem with Redundant Data:

- Imagine if you were trying to figure out how good your detective skills are, but the clues you had were repeated or not very useful. It would be hard to tell if you're truly skilled at solving mysteries.
- Similarly, if a dataset has a lot of redundant (repeated or unnecessary) information, it makes it difficult to trust the evaluation results of LuNet.

3. LuNet's Approach - Non-Redundant Datasets:

- To ensure that the evaluation of LuNet is reliable and trustworthy, the researchers choose two specific datasets: NSL-KDD and UNSW-NB15.
- These datasets are carefully selected because they are non-redundant. This means they don't have unnecessary or repeated information, making the evaluation results more dependable.

In simpler terms, when testing LuNet's abilities, it's important to use a dataset that doesn't have a lot of repeated or useless information. This ensures that the results are trustworthy. LuNet's evaluation is

done using two specific datasets, NSL-KDD and UNSW-NB15, which are chosen because they are non-redundant and provide reliable results for assessing LuNet's performance in detecting network intrusions.



You:

explain the following in the simple language "NSL-KDD [23] was generated by Canadian Institute for Cyber Security (CICS) from the original KDD'99 dataset. It consists of 39 different types of attacks. The attacks are categorized into four groups: Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L) and Probe. A big issue with this dataset is its imbalanced distribution, which often leads to a high false positive rate due to insufficient data available for training [24] - [25]. Here, we tackle this problem with a cross-validation scheme3."



ChatGPT:

Certainly! Let's simplify this:

About the NSL-KDD Dataset:

1. Origin of NSL-KDD:

- NSL-KDD is a special set of data created by the Canadian Institute for Cyber Security (CICS). They made it from the original KDD'99 dataset.

2. Types of Attacks in NSL-KDD:

- This dataset represents different ways that a computer network can be attacked. There are 39 types of attacks, and they fall into four main groups: Denial of Service (DoS), User to Root (U2R), Remote to Local (R2L), and Probe.

3. Problem with Imbalanced Distribution:

- However, there's a challenge with this dataset. The distribution of attacks is not equal; some types of attacks happen more frequently than others.
- This imbalance can cause an issue. Imagine if you were trying to learn about different animals, but you had way more information about cats than any other animal. You might end up thinking everything is a cat!
- Similarly, in NSL-KDD, this imbalance might lead to LuNet wrongly thinking certain attacks are more common than they actually are.

4. Addressing the Imbalance with Cross-Validation:

- To fix this problem, the researchers use something called a cross-validation scheme. Think of it as a smart way of learning from the dataset.
- It's like making sure LuNet doesn't get tricked into thinking one type of attack is more important just because there's more data about it. The cross-validation scheme helps LuNet learn in a balanced way.

In simpler terms, the NSL-KDD dataset is a collection of information about different ways a computer network can be attacked. However, there's a challenge because some attacks are more common than others. To help LuNet learn without being biased, researchers use a cross-validation scheme, making sure it gets a fair understanding of all types of attacks. It's like teaching LuNet to recognize all animals, not just the ones that show up more often in the data.



You:

explain the following in the simple language "UNSW-NB15 [27] - [28], generated by Australian Center for Cyber Security (ACCS) in 2015, is a more contemporary dataset. For the dataset, the attack samples were first collected from the three real-world websites: CVE (Common Vulnerabilities and Exposures)⁴, BID (Symantec Corporation)⁵, and MSD (Microsoft Security Bulletin)⁶. The sample attacks were then simulated in a laboratory environment for the dataset generation. There are nine attack categories in UNSW-NB15: DoS, Exploits, Generic, Shellcode, Reconnaissance, Backdoor, Worms, Analysis, and Fuzzers."



ChatGPT:

Absolutely, let's break this down:

About the UNSW-NB15 Dataset:

1. Creation of UNSW-NB15:

- The UNSW-NB15 dataset was made by the Australian Center for Cyber Security (ACCS) in 2015.

2. Contemporary Nature of the Dataset:

- It's a more modern or contemporary dataset, meaning it's based on recent information about cyber threats.

3. Source of Attack Samples:

- To create this dataset, they didn't just make things up. Instead, they collected samples of real attacks from three actual websites: CVE (Common Vulnerabilities and Exposures), BID (Symantec Corporation), and MSD (Microsoft Security Bulletin).

4. Simulating Attacks in a Lab Environment:

- After collecting these attack samples, they didn't unleash them on the internet! That would be dangerous. Instead, they simulated these attacks in a controlled lab environment to generate the dataset.

5. Categories of Attacks in UNSW-NB15:

- The dataset includes nine categories of attacks: Denial of Service (DoS), Exploits, Generic attacks, Shellcode attacks, Reconnaissance, Backdoor attacks, Worms, Analysis, and

Fuzzers.

In simpler terms, the UNSW-NB15 dataset is a collection of information about various cyber threats. It's not just made up; the attacks are based on real-world samples from important websites. To make sure things are safe, these attacks were simulated in a controlled environment. This dataset is more up-to-date and covers different types of cyber threats. It's like a comprehensive guide for LuNet to learn about the latest tricks cyber attackers might use.



You:

explain the following in the simple language "V. EVALUATION AND DISCUSSION To evaluate our design, we have implemented LuNet (as given in Fig. 2) with TensorFlow backend, Keras and scikit-learn packages and we run the training on a HP EliteDesk 800 G2 SFF Desktop with Intel (R) Core (TM) i5-6500 CPU @ 3.20 GHz processor and 16.0 GB RAM. For comparison, we also implemented a set of state-of-the-art machine learning algorithms."



ChatGPT:

Absolutely, let's simplify this:

Evaluation and Testing LuNet:

1. Creating LuNet:

- The researchers built LuNet, the design they've been talking about, using tools like TensorFlow, Keras, and scikit-learn.
- It's like they built LuNet using a toolbox that helps make complex things (like LuNet) easier to create.

2. Where They Ran the Tests:

- To see how well LuNet works, they ran the training (teaching LuNet) on a specific computer. This computer is an HP EliteDesk 800 G2 SFF Desktop with an Intel Core i5-6500 CPU running at 3.20 GHz and 16.0 GB of RAM.
- Think of it like LuNet going to school on a particular type of computer.

3. Comparing LuNet to Other Methods:

- To check if LuNet is really good, they didn't stop there. They also implemented several other advanced machine learning methods for comparison.
- It's like testing LuNet against other smart students to see if it's the best in the class.

In simpler terms, the researchers put LuNet to the test! They created it using special tools, taught it on a specific computer, and then compared its performance with other smart methods. It's like seeing how well LuNet performs in a competition against other clever students.



You:

explain the following in the simple language "The description of experiments and experiment results and discussion are presented below. A. Data Preprocessing 1) Convert Categorical Features: For the experiment to be effective, we need data to conform to the input format required by the neural network. Raw network traffic data include some categorical features as shown in Fig.4. The text information cannot be processed by a learning algorithm and should be converted into numerical values. Here, we use the 'get dummies' function in Pandas [29] to operate the conversion. 4CVE: <https://cve.mitre.org/> 5BID: <https://www.securityfocus.com> 6MSD: <https://docs.microsoft.com/en-us/security-updates/securitybulletins>"



ChatGPT:

Certainly! Let's break down the process of preparing the data for experiments:

Getting the Data Ready:

1. Overview of Experiments:

- Before they started testing LuNet, the researchers needed to make sure the data was in a suitable form. This section talks about what they did to prepare for the experiments.

2. Data Preprocessing - Step 1: Convert Categorical Features:

- The raw network traffic data comes with some features that aren't numbers, like categories or types. Machines prefer numbers, so they needed to convert these categories into numerical values.
- Imagine you have a list of animals, and some are labeled as "cats" or "dogs." Machines don't understand "cats" and "dogs," so you'd need to convert them into numbers like 1 for cats and 2 for dogs.
- In this case, they used a tool called 'get dummies' in Pandas to do this conversion. Pandas is like a translator that turns categories into numbers, making it easier for LuNet to understand.

3. Example of Categorical Features:

- The data they were working with had information like Common Vulnerabilities and Exposures (CVE), Symantec Corporation (BID), and Microsoft Security Bulletin (MSD). These are examples of categorical features.
- Converting them into numbers allows LuNet to process and learn from this information effectively.

In simpler terms, before testing LuNet, the researchers needed to make sure the data was in a language it could understand. They had data that used words, like types of vulnerabilities or security updates, and they needed to change these words into numbers. They used a tool called 'get dummies' to do this, making LuNet's job easier by giving it numerical information to work with. It's like turning a list of animals into a list of numbers, so LuNet can learn from it more easily.



You:

explain the following in the simple language "2) Standardization: Input data may have varied distributions with different means and standard deviations, which may affect learning efficiency. We apply standardization to scale the input data to have a mean of 0 and a standard deviation of 1, as is often applied in many machine learning classifiers"



ChatGPT:

Certainly! Let's simplify this:

Making the Data Fair:

1. Different Distributions in the Data:

- Imagine you have a bunch of numbers, but some are big and some are small. It's like having a mix of dollars and cents without a clear pattern.
- The input data (information given to LuNet) can be like this, with numbers having different averages and spreads.

2. Why It Matters:

- This variety in numbers can make it a bit confusing for LuNet. It's like trying to learn about animals when the information comes in different languages.

3. Making Things Fair - Standardization:

- To make things fair, they apply something called standardization.
- It's like converting all the numbers into a common language. They want the numbers to follow a standard pattern so that LuNet can understand and learn from them more easily.

4. How They Do It:

- They scale the input data so that, on average, the numbers have a mean (average) of 0 and a standard deviation (a measure of how spread out the numbers are) of 1.
- Think of it like making all the numbers speak the same language and use the same currency. This helps LuNet learn efficiently without getting confused by the different scales of the numbers.

In simpler terms, the researchers wanted to make sure LuNet could learn from the data without being confused by different scales of numbers. It's like making all the numbers play by the same rules, so LuNet can understand and learn from them more effectively.



explain the following in the simple language "3) Stratified K-Fold Cross Validation: NSL-KDD and UNSW-NB15 contain 148,516 and 257,673 samples, respectively. To realize the large non-redundant data for training and verification, we employ a Stratified K-Fold Cross Validation strategy, also commonly used in machine learning. The scheme splits all samples in a dataset into k groups; Among them, k-1 groups, as a whole, are used for training and the rest one group for validation; hence the strategy is also called Leave One Out Strategy."



Certainly! Let's break down the concept of Stratified K-Fold Cross Validation:

Fair Testing with Cross Validation:

1. Different Datasets Sizes:

- Imagine you have two sets of data - let's call them Set A and Set B. Set A has 148,516 pieces of information, and Set B has 257,673 pieces.

2. Making Sure the Test is Fair:

- To test LuNet fairly, they want to use all the available data. However, using all of it at once might be tricky. So, they came up with a smart way to do it.

3. Stratified K-Fold Cross Validation - How It Works:

- They use something called Stratified K-Fold Cross Validation, which is like a systematic way of testing.
- Here's how it works:
 - They divide each set into k groups. For example, if k is 5, they'd have 5 groups.
 - They then use 4 groups for training (teaching LuNet) and the remaining 1 group for testing how well LuNet has learned.
 - They repeat this process, making sure every group gets a chance to be the test set.

4. Why It's Special - Stratified:

- The word "Stratified" is there to ensure that each group has a fair mix of different types of information. It's like making sure that in each round of testing, LuNet gets a taste of all the different things it might encounter.

5. Leave One Out Strategy:

- Another name for this strategy is "Leave One Out." This means they leave one group out for testing, and the others are used for training.
- It's a bit like having a friend taste different parts of a big cake to make sure it's delicious all around.

In simpler terms, to test LuNet fairly, they split the data into groups and take turns using each group to see how well LuNet learns. It's like making sure LuNet gets a good taste of all the information without

being overwhelmed. This way, they can be confident that LuNet can handle various situations it might face.



You:

explain the following in the simple language "B. Evaluation Metrics We evaluate LuNet in terms of the validation accuracy (ACC), detection rate (DR) and false positive rate (FPR). ACC measures LuNet's ability to correctly predict both attacked and non-attacked normal traffic, while DR indicates its ability of prediction for attacks only. A high DR can be shadowed by a high rate of false positive alarm (FPR), which therefore needs to be jointly considered with DR. The detail definitions of the three metrics are given in Formulas (7), (8) and (9). ACC = $(TP + TN) / (TP + TN + FP + FN)$, (7) DR = $TP / (TP + FN)$, (8) FPR = $FP / (FP + TN)$, (9) where TP and TN are, respectively, the number of attacks and the number of normal traffic correctly classified; FP is the number of actual normal records mis-classified as attacks, and FN is the number of attacks falsely classified as normal traffic."



ChatGPT:

Certainly! Let's simplify the evaluation metrics:

Measuring How Well LuNet Performs:

1. Three Important Metrics:

- When checking how well LuNet is doing, the researchers look at three main things: validation accuracy (ACC), detection rate (DR), and false positive rate (FPR).

2. Validation Accuracy (ACC):

- Validation accuracy is like checking how many times LuNet gets it right overall.
- It's calculated by adding up the number of times LuNet correctly predicts attacks and non-attacks and dividing it by the total predictions made.

3. Detection Rate (DR):

- Detection rate focuses specifically on how well LuNet catches attacks.
- It's calculated by dividing the number of attacks LuNet correctly identifies by the total number of actual attacks.

4. False Positive Rate (FPR):

- False positive rate looks at how often LuNet says there's an attack when there isn't.
- It's calculated by dividing the number of normal records (non-attacks) that LuNet incorrectly classifies as attacks by the total number of actual normal records.

5. Why All Three Matter:

- Validation accuracy gives an overall picture of LuNet's performance.
- Detection rate shows how well LuNet catches attacks.

- False positive rate warns if LuNet is too trigger-happy, saying there's an attack when there isn't.

6. In Simple Terms:

- If you imagine LuNet as a security guard, validation accuracy is how often they get everything right, detection rate is how well they catch the bad guys, and false positive rate is how often they mistakenly think someone innocent is a bad guy.

In simpler terms, when checking how well LuNet is doing, the researchers use three main measures: how often it gets everything right, how well it catches bad things, and how often it mistakenly thinks something is bad when it's not. It's like evaluating a security guard's performance in making sure they do a good job overall, catch the bad guys, and don't mistakenly accuse innocent people.



You:

explain the following in the simple language "C. LuNet Results In our experiments, we use RMSprop [30] (a popular gradient descent algorithm) to optimize weights and biases for LuNet training, and for the learning rate, we set it to a median value 0.001 in a range given by the tensorflow library. We also choose the default rate value, 0.5, for the dropout layer."



ChatGPT:

Certainly! Let's simplify the LuNet results section:

How LuNet was Trained and Tuned:

1. Training LuNet:

- Think of training LuNet like teaching it to be good at its job. It needs to adjust its "weights and biases," which are like the settings that help it make decisions.
- They used a popular method called RMSprop, which is a way of fine-tuning LuNet's decision-making process.

2. Optimizing Weights and Biases:

- Optimizing is like finding the best settings for LuNet so it can be really good at catching attacks and not making too many mistakes.
- They used the learning rate (how quickly LuNet adjusts) and set it to a value that's considered a good middle ground.

3. Dropout Layer:

- LuNet has a "dropout layer" which is like LuNet taking breaks during training to avoid getting too good at just the specific data it's learning from.
- They set the dropout rate to a default value of 0.5, meaning it randomly removes half of the connections during training to keep things diverse.

In simpler terms, when training LuNet, they used a well-known method to fine-tune its decision-making settings, adjusted how quickly it learns, and made sure it takes breaks during training to stay flexible. It's like teaching LuNet to be really good at its job without becoming too specialized or making too many mistakes.



You:

explain the following in the simple language "We first measure the performance of LuNet based on two scenarios: (1) binary classification, namely LuNet predicts a packet either as an attack or as a normal traffic; (2) multi-class classification, where LuNet identifies a packet either as normal or as one type of attacks given in the dataset attack model (namely, 5 classes for NSL-KDD and 10 classes for UNSW-NB15). The experiment results are given below."



ChatGPT:

Absolutely! Let's break down the evaluation scenarios for LuNet:

Testing LuNet in Two Ways:

1. Binary Classification:

- In the first scenario, it's like LuNet is playing a game of "attack or not attack" with each network packet.
- LuNet has to decide if a packet is either an attack or just regular normal traffic.
- It's a bit like LuNet being a detective, figuring out if each packet is a bad guy or not.

2. Multi-Class Classification:

- Now, in the second scenario, LuNet is playing a more complex game. It's like LuNet has a list of different types of attacks (like different flavors of bad guys) that it has to recognize.
- For NSL-KDD, there are 5 classes of attacks, and for UNSW-NB15, there are 10 classes.
- LuNet has to not only say if something is an attack or not but also figure out which specific type of attack it is.
- It's like LuNet upgrading from a detective to a detective who can identify different types of crimes.

Experiment Results:

- The researchers then ran experiments to see how well LuNet did in both of these scenarios.
- The results of these experiments are what they're about to share – it's like telling the story of how LuNet performed in these games of "attack or not attack" and "which type of attack is it."

In simpler terms, they tested LuNet in two ways: one where it had to decide if each packet is an attack or not, and another where it had to identify the specific type of attack. The results they're about to share will tell us how well LuNet did in these scenarios, like a report card for its detective skills.



You:

explain the following in the simple language "1) Binary Classification: Table I shows the detection rate, accuracy and false positive rate for the binary classification of LuNet under different Stratified K-Fold Cross Validations, with k ranging from 2 to 10. The average values are given in the last row of the table. As can be seen from the table, LuNet can achieve around 99.24% and 97.40% validation accuracy on NSL-KDD and UNSW-NB15, respectively. The best validation accuracy of 99.36% on the NSL-KDD dataset and 97.67% on the UNSW-NB15 can be observed. It can also be seen that LuNet offers a high detection capability while incurs a low false positive alarm rate, as plotted Fig.5. On average, DR=99.42% and FPR=0.53% for NSL-KDD, and DR=98.18% and FPR=3.96% on UNSW-NB15 can be obtained. It is worth to note that because of the cross-validation used, the false positive alarm rate is not affected by the imbalanced NSL-KDD dataset, as is expected."



ChatGPT:

Absolutely, let's break down the results in simpler terms:

Binary Classification Results for LuNet:

1. Detecting Attacks or Normal Traffic:

- Imagine LuNet as a superhero detective trying to decide if each network packet is a bad guy (attack) or just a regular citizen (normal traffic).

2. Table I Summary:

- Table I is like LuNet's report card, showing how well it did in different situations (cross-validations).
- The table has numbers for Detection Rate (DR), Accuracy, and False Positive Rate (FPR). These numbers tell us how good LuNet is at catching attacks without making mistakes.

3. Validation Accuracy:

- For NSL-KDD (the first dataset), LuNet is like a superhero with an accuracy of around 99.24%, meaning it rarely makes mistakes.
- For UNSW-NB15 (the second dataset), it's still impressive with an accuracy of about 97.40%.

4. Detection Capability and False Alarms:

- LuNet is really good at catching attacks (high DR) while keeping false alarms low (low FPR).
- On average, it catches about 99.42% of attacks with only a 0.53% false positive rate for NSL-KDD.
- For UNSW-NB15, it catches about 98.18% of attacks with a 3.96% false positive rate.

5. Cross-Validation Note:

- They used something called cross-validation to make sure LuNet's accuracy isn't influenced by the imbalanced nature of the NSL-KDD dataset (where there are fewer examples of attacks). Cross-validation is like a fair testing ground.

In simpler terms, LuNet is a superhero detective with an excellent record of catching bad guys (attacks) while rarely mistaking normal citizens (keeping false alarms low). The results show how trustworthy LuNet is in different situations.



You:

explain the following in the simple language "2) Multi-Class Classification: Table II shows the multi-class classification results. It can be seen from the table that LuNet can achieve an average of 99.05% accuracy, 98.84% detection rate on NSL-KDD, and 84.98% accuracy and 95.96% detection rate on UNSW-NB15 with, respectively, the 0.65% and 1.89% false positive rate on the two datasets. Fig.6 shows the detection rate (DR%) and false positive rate (FPR%) for 5 category (Normal, DoS, Probe, R2L, U2R) classification on NSL-KDD. "



ChatGPT:

Absolutely, let's simplify the results for the multi-class classification:

Multi-Class Classification Results for LuNet:

1. Recognizing Different Types of Attacks:

- Now, imagine LuNet as a superhero detective who not only catches bad guys but also figures out which specific type of bad guy they are.

2. Table II Summary:

- Table II is like LuNet's report card for recognizing different types of attacks.
- The table has numbers for Accuracy, Detection Rate (DR), and False Positive Rate (FPR).

3. Accuracy and Detection Rates:

- For NSL-KDD (the first dataset), LuNet is a superhero with an average accuracy of about 99.05%, which means it's really good at identifying various types of attacks.
- It also catches about 98.84% of attacks, showing its ability to recognize specific bad guys.
- For UNSW-NB15 (the second dataset), LuNet maintains a good accuracy of around 84.98% and a detection rate of about 95.96%.

4. False Positive Rates:

- LuNet is careful not to make too many mistakes. It only has a 0.65% false positive rate on NSL-KDD and a 1.89% false positive rate on UNSW-NB15. This means it doesn't wrongly identify normal traffic as attacks very often.

5. Fig.6:

- Fig.6 is like a visual representation of how well LuNet does in recognizing different attack types on NSL-KDD. It shows the Detection Rate (DR%) and False Positive Rate (FPR%) for different categories of attacks.

In simpler terms, LuNet is an impressive superhero detective that not only catches bad guys but can also tell you exactly which type of bad guy they are. The results show how accurate LuNet is in recognizing various types of attacks without making too many mistakes.



You:

explain the following in the simple language "As we can see from the figure, LuNet shows an excellent capability to handle attacks in all categories (with a high detection rate and low false positive rate), except for U2R and R2L. For U2R and R2L, LuNet presents a relatively low detection rate, which indicates that the main features extracted by LuNet for U2R and R2L are not effectively distinct. There may be significant feature overlap between U2R and other attacks. The same reason is also applied to the Backdoor and Worms attacks, as manifested in the similar plot shown in Fig.7 for classification of 10 categories (Normal, Analysis, Backdoor, DoS, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, Worms) on UNSW-NB15."



ChatGPT:

Let's break down what the figure and explanation are saying:

Understanding LuNet's Performance on Different Attacks:

1. Figure Analysis:

- Imagine the figure is like a superhero report card for LuNet. It shows how well LuNet deals with different types of attacks (like villains).
- The goal is to have a high "Detection Rate" (finding the bad guys) and a low "False Positive Rate" (not mistakenly thinking good guys are bad).

2. Excellent Handling of Most Attacks:

- In the figure, you can see LuNet is like a superhero ace at handling most types of attacks. It's really good at finding them (high detection rate) and rarely gets it wrong (low false positive rate).
- Categories like "Normal," "DoS," "Probe," and others are a piece of cake for LuNet.

3. Challenges with U2R and R2L:

- Now, there's a bit of a challenge with "U2R" and "R2L." LuNet isn't as confident in detecting these types. The detection rate is not as high.
- This suggests that LuNet finds it a bit tricky to distinguish these attacks from others. It's like saying the features that make U2R and R2L attacks unique aren't standing out clearly.

4. Feature Overlap:

- The reason for this might be that the characteristics of U2R attacks and some other attacks overlap. Imagine if some villains wore similar disguises – LuNet might find it hard to tell them apart.

5. Similar Situation for Backdoor and Worms:

- The same situation happens with "Backdoor" and "Worms" attacks in another plot (Fig.7). LuNet faces a challenge in separating these from other attacks, suggesting some shared features.

In simpler terms, LuNet is like a superhero that handles most types of attacks exceptionally well, but there are a few tricky ones like U2R and R2L. It's like these villains have some similarities, making it a bit tough for LuNet to always identify them correctly.



You:

explain the following in the simple language "Fig.7 illustrates that LuNet can successfully detect most of the Normal traffic, Exploits, Generic, DoS, Shellcode and Reconnaissance attacks (see the overlapped lines at the very top of the plots) and has a moderate capability to detect Analysis attacks with the low false positive rate (FPR). However, it is unable to discover Backdoor and Worms attacks, as can be observed in the confusion matrix (as shown in Fig. 8) generated from the experiment. We discover that most of Backdoor and Worms were, in fact, classified as Exploits attacks by LuNet. The possible reason is that the Backdoor and Worms attacks use the common exploit flaws in the programs that listen for connections from remote hosts and the signature of those attacks is similar to the Exploits attacks. Therefore, the Backdoor and Worms attacks are treated as the Exploits attacks by LuNet. Another possible reason may come from the insufficient data available for the Backdoor and Worms attacks; There are only around 1.4% Backdoor attacks and 1.1% Worms attacks in the dataset."



ChatGPT:

Let's break down what is being explained in simple terms:

Understanding LuNet's Detection and Challenges with Specific Attacks:

1. Figure 7 Overview:

- Think of Figure 7 like a comic strip showing LuNet's superhero abilities in detecting various types of network traffic attacks.
- It's good at finding "Normal" traffic and several types of attacks like "Exploits," "Generic," "DoS," "Shellcode," and "Reconnaissance" (shown at the top of the plots).

2. Detection Limitations:

- However, there are challenges. LuNet struggles to spot "Backdoor" and "Worms" attacks. It's like these villains are slipping through LuNet's radar.

3. Confusion Matrix (Fig. 8):

- Imagine a detective's board with pictures and notes – that's what the confusion matrix (Fig. 8) is like. It shows how LuNet categorized different attacks.
- Surprisingly, LuNet often mistakes "Backdoor" and "Worms" attacks as "Exploits." It's like LuNet sees these villains wearing similar disguises, and it gets a bit confused.

4. Possible Reasons for Confusion:

- **Similar Exploit Tactics:** LuNet might think "Backdoor" and "Worms" attacks are actually "Exploits" because they use similar strategies to exploit weaknesses in computer programs.
- **Not Enough Villain Data:** There's also a lack of data about "Backdoor" and "Worms" attacks (only about 1.4% and 1.1% in the dataset). It's like LuNet hasn't seen enough of these villains to recognize them well.

In simpler terms, LuNet is like a superhero detective with impressive skills but faces a bit of trouble distinguishing between certain villains, especially "Backdoor" and "Worms," due to their similar tactics and limited information about them.



You:

explain the following in the simple language "D. Comparative Study We compare the performance of LuNet with other five state-of-the-art supervised learning algorithms that have been investigated in the literature: Support Vector Machine (SVM) with Gaussian Kernel (RBF) [5], Multi-Layer Perceptron (MLP) [31], Random Forest (RF) [7], Adaptive Boosting (AdaBoost) [6], and HAST-IDS [14]."



ChatGPT:

Alright, let's simplify this:

Comparing LuNet with Other Superheroes (Algorithms):

1. LuNet – Our Superhero:

- Imagine LuNet as a superhero with unique powers designed to catch bad guys in network data.

2. Other Superheroes (Algorithms):

- LuNet doesn't want to work alone; it wants to team up with other superheroes (algorithms) to see who catches more villains.

3. List of Superheroes (Algorithms):

- LuNet wants to compare its performance with five other superheroes: Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), Random Forest (RF), Adaptive Boosting (AdaBoost), and HAST-IDS.

4. Their Superpowers:

- Each superhero (algorithm) has its own special powers. SVM uses a powerful "Gaussian Kernel," MLP is like a multi-layered superhero, RF is a forest of algorithms, AdaBoost adapts and boosts its abilities, and HAST-IDS has its unique skills.

5. The Showdown:

- LuNet is ready for a showdown to see who can catch more villains in the network data.

In simple terms, the "Comparative Study" is like a superhero team-up. LuNet wants to see if it catches more bad guys than other superhero algorithms. Each algorithm has its own special powers, and they're all in a friendly competition to protect the network world.



You:

explain the following in the simple language "We also run the experiment on individual CNN and LSTM networks, presented in LuNet. The detection rate (DR%), accuracy (ACC%), and false positive rate (FPR%) of the two design groups (ML, the classical machine learning algorithms and DL, the deep learning algorithms) are given in Table III. The average values of each group are also provided in the table"



ChatGPT:

Sure, let's simplify this:

Testing Superpowers - LuNet, CNN, and LSTM:

1. Our Superheroes:

- Imagine we have three superheroes - LuNet, CNN, and LSTM. Each superhero has its own unique powers to catch villains in the network.

2. The Experiment:

- We decided to run an experiment to see how good each superhero is at catching bad guys.

3. Two Superhero Groups:

- We grouped our superheroes into two teams - "ML" (Classic Machine Learning) and "DL" (Deep Learning). It's like dividing superheroes based on their superpowers.

4. Testing Individual Superheroes:

- First, we tested LuNet, then CNN, and then LSTM separately. It's like checking each superhero's performance on their own.

5. Results - Who's the Best?

- We measured three things for each superhero - how well they detect bad guys (Detection Rate - DR%), how accurate they are (Accuracy - ACC%), and how often they make mistakes (False Positive Rate - FPR%).

6. Team Averages:

- Finally, we averaged out the performance of all superheroes in each group. It's like finding out how good the whole team of "Classic Machine Learning" and "Deep Learning" superheroes is on average.

In simple terms, we tested each superhero's powers individually and grouped them based on their superpowers to see which group is more effective in catching villains. It's like a superhero competition to find out who's the best in different categories.



You:

explain the following content in the simple language "As can be seen from Table III, the deep neural networks generally have better performance than the three classical machine learning algorithms; On average, the deep neural networks have a higher detection rate (94.57%) and accuracy (82.78%) and a lower false positive rate (4.72%), as compared to the corresponding average values of 89.03%, 77.53% and 10.95% from the classical machine learning. Among the five deep learning networks, LuNet is better than other four DL algorithms due to the special combined CNN+RNN hierarchy used. In summary, LuNet demonstrates its superiority – achieving a high detection rate and accuracy, while keeping a low false positive rate"



ChatGPT:

Absolutely, let's simplify that:

Superhero Showdown Results: LuNet vs. Others

1. The Battle of Superpowers:

- We had two teams - one with classic superheroes (like Superman and Batman) called "Classical Machine Learning," and the other with new-age superheroes (like Iron Man and Wonder Woman) called "Deep Learning."

2. Showdown Stats - Who's Better?

- We compared their performances in a superhero showdown (Table III). The deep learning superheroes turned out to be more effective overall.

3. Deep Learning Heroes Shine:

- On average, the deep learning superheroes (including LuNet) showed better results. They had a higher success rate in catching bad guys (Detection Rate - 94.57%) and were more accurate (Accuracy - 82.78%). They also made fewer mistakes (False Positive Rate - 4.72%).

4. LuNet Steals the Show:

- Among all the deep learning superheroes, LuNet emerged as the ultimate superhero. It outperformed the other four deep learning heroes because of its unique combined powers (CNN+RNN hierarchy).

5. In a Nutshell:

- LuNet proved it's the real superhero champion - it caught more bad guys, was accurate in its actions, and made fewer errors compared to other superheroes.

In simple terms, it's like comparing traditional superheroes with modern superheroes. The new ones, especially LuNet, showed they have better abilities and are more effective in keeping the city safe.



You:

explain the following in the simple language "VI. CONCLUSION In this paper, we present a deep neural network architecture, LuNet, to detect intrusions on a large scale network. LuNet uses CNN to learn spatial features in the traffic data and LSTM for temporal features. To avoid the information loss due to different learning focuses of CNN and RNN, we synchronize both CNN and RNN to learn the input data at the same granularity. To enhance the learning, we also incorporate batch normalization in the design. Our experiments on the two non-redundant datasets, NSL-KDD and UNSW-NB15, show that LuNet is able to effectively take advantages of CNN and LSTM. Compared with other state-of-the-art techniques, LuNet can significantly improve the validation accuracy and reduce the false positive rate for network intrusion detection. It must be pointed out that LuNet does not work well to classify attacks of insufficient samples in the training dataset, such as Backdoors and Worms, as observed in our experiment, which will be investigated in the future."



ChatGPT:

Certainly! Let's break it down:

Conclusion of the Superhero Journey - LuNet's Triumph

1. The Birth of LuNet:

- We introduced our superhero, LuNet - a powerful network defender designed to catch the bad guys in a large network.

2. Superpowers Unleashed:

- LuNet comes equipped with two special powers: CNN (for spatial features) and LSTM (for temporal features). Think of it like having superhero senses to see and predict what's happening.

3. Teamwork for Success:

- LuNet's special power duo (CNN and LSTM) work together seamlessly to understand both the space and time aspects of the network data. This teamwork ensures no important details are left out.

4. Boosting Learning with Special Techniques:

- To make LuNet even stronger, we added a special technique called batch normalization. This helps LuNet learn more efficiently and accurately.

5. Superhero Training Grounds - Real-world Challenges:

- We put LuNet to the test in the real world, using two unique datasets (NSL-KDD and UNSW-NB15). The results were impressive!

6. LuNet's Victory Dance:

- LuNet outshone other superhero techniques. It not only increased the accuracy of identifying threats but also reduced the chances of mistaking normal activities for attacks.

7. Facing Kryptonite - Areas to Improve:

- LuNet, like all superheroes, has a weakness. It struggled a bit with attacks that didn't have enough training examples, like Backdoors and Worms. We acknowledge this and plan to work on it in the future.

In a nutshell, LuNet has proven to be a superhero in the world of network defense, and although it has its challenges, it's a force to be reckoned with!
