MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 2, 2026, Pages 1137-1146

Temporal Graph Neural Networks for Energy Grid Component Degradation Modeling: Predicting Failures in Interconnected Networked Systems

Md Shihab Sadik Shovon[1], Mehedi Hasan Ridoy[2], Mohammad Hamid Hasan Amjad[3], Tapan Chandra Ghos [4], A S M Mahamudul Hasan[5], Md Rakib Uddin[6] and Rakib Hassan Rimon[7]

[1]College of Graduate and Professional Studies, Trine University, Angola, Indiana 46703, USA
[2]MBA - Business Analytics, Gannon University, Erie, PA, USA.
[3]College of Graduate and Professional Studies, Trine University, Angola, Indiana 46703, USA
[4]School of Engineering & Technology, Western Illinois University, Macomb, Illinois, USA
[5]College of Graduate and Professional Studies, Trine University, Angola, Indiana 46703, USA
[6]College of Graduate and Professional Studies, Trine University, Angola, Indiana 46703, USA
[7]Business analytics, Grand Canyon University
Corresponding Author: **Md Shihab Sadik Shovon, Email: mshovon23@my.trine.edu**

Abstract

Today's energy grids function as massive, interconnected networks where a bit of stress in one spot can easily travel through the wires and set off a chain reaction of failures. Most of the usual ways people try to predict maintenance or use machine learning focus on looking at parts one by one. Or, they just look at big-picture stats. Neither of these really captures how everything is linked or how the whole system's weaknesses change over time. This work puts forward a temporal graph learning setup to model how operational risks grow and to predict when a major cascading failure might happen. By using the PowerGraph benchmark data, grid operations are turned into a series of graphs over time. These graphs include electrical readings at specific points, the details of the lines connecting them, and the actual shape of the whole network. The system blends graph convolutional learning with recurrent models to track both the physical connections and the changing behavior of the grid as it moves from one state to the next. The testing process looked at several different setups, from old-school machine learning and standard sequence models to static graphs and more advanced versions that pay attention to the connections between points. It turned out that paying attention to the network's layout makes a big difference in spotting risks compared to models that ignore the graph structure. This suggests that how likely a grid is to fail depends a lot on how it is actually wired together. Models that factored in the behavior of the branches themselves did even better at picking out serious events. Even so, predicting these massive failures is still a huge challenge. They don't happen often, and the way they spread is incredibly messy and hard to map out. Beyond just the main results, the study looked into how reliable the predictions were, how much the timing mattered, and where the models tended to trip up. It was also important to see if a model trained on one grid would work on another. It should be noted that this isn't about tracking how a specific piece of equipment gets old or rusty. Instead, it treats degradation as a build-up of stress and a higher chance of a system-wide crash. In the end, the work shows that these temporal graph neural networks are a strong way to build early-warning systems that understand grid topology, even though there are still hurdles when it comes to using simulated data and moving between different grid setups.

Keywords: Cascading Failure; Demand Not Served; Energy Grid Risk; Graph Neural Networks; PowerGraph; Temporal Graph Learning

1. Introduction

1.1 Background and Motivation: The way energy grids work has changed. It used to be a simple setup where power just went from a big plant to people's homes. Now, it is a massive, tangled web of wires, buses, generators, and gadgets that all talk to each other. Because everything is so connected, you can't just look at one transformer or one line and say it is fine. If one spot gets too hot or a line gets overloaded, that stress doesn't just stay put. It moves. It ripples through the network and messes with things miles away. This makes it really hard to predict when a grid might fail. Predicting a crash isn't just about checking a single machine; it is about knowing where that machine sits in the whole mess and what its neighbors are doing. Back in 2010, Buldyrev and his team showed that when these big networks are tied together, one small break can cause a massive, sinking-ship kind of collapse. It proved that you have to look at the whole grid as a living network, not just a bunch of separate parts sitting in a field [4].

Most people who study maintenance like to look at single pieces of equipment. They take a sensor, look at the vibration or temperature of a pump or an engine, and try to guess when it will quit. This works great for a turbine or a motor, where you can actually see the wear and tear happening in the data. Lei and others talked about this in 2018, explaining how you take raw data and turn it into a prediction for how much life a machine has left [16]. Power grids have some of that same stuff, but they have a bigger problem: the network itself. A transformer might look perfectly healthy on paper, but if the wind stops blowing or everyone turns on their AC at once, the way power flows through the grid changes. Suddenly, that "healthy" transformer is in a dangerous spot because of what is happening around it.This study looks at things a bit differently. Instead of looking at how old a wire is or if the insulation is rotting, it looks at operational risk. This means looking at how much stress the grid is under, whether voltages are acting weird, and if the whole thing is becoming prone to a big domino-effect failure. This is an important distinction because the data used here comes from PowerGraph simulations, not from decades of maintenance logs or physical rust. Sun noted in 2018 that modern grids are basically half-computer and half-power lines, so staying safe is about having real-time awareness of these vulnerabilities [25]. To get an early warning, a model has to understand the grid as a moving, changing thing. It needs to see how the shape of the network spreads risk. That is why using temporal graph neural networks makes so much sense for this kind of work.

1.2 Problem Statement: The main issue is that a lot of the standard math models people use aren't built for stuff that is both connected and always changing. Usually, researchers take a bunch of averages, like total power used or average line load, and try to predict what happens next. Those numbers are okay for a general snapshot, but they hide the scary details. You could have two days where the total power use looks the same, but on one of those days, all the stress is piled up on one critical line. If that line goes, the whole grid goes. If you only look at the averages, you miss that. There is a real gap between just crunching numbers in a table and actually understanding the shape, or topology, of the risk. Graph neural networks are the way to fix this. Back in 2017, Kipf and Welling came up with a way for computers to learn about data by looking at how points are connected [15]. In a power grid, the buses are nodes, and the lines are edges. By using this kind of model, the system can learn that the "where" matters as much as the "how much." It understands that a voltage drop in a tiny cul-de-sac is different from a voltage drop at a major hub. For something like a cascading failure, the location of the stress is everything.The other big hurdle is time. The grid isn't static. A crash doesn't usually just happen out of nowhere; it builds up. Maybe the load shifts, then a line gets slightly too full, then another one compensates, and slowly the whole system gets pushed to the edge. Rossi and his colleagues suggested in 2020 that Temporal Graph Networks are the best tool here because they can track these changes as they happen over time [21]. That is what this study does, it looks at windows of time in the PowerGraph data to see if it can spot the risk building up before the lights go out. The data itself sets some ground rules. PowerGraph is a solid set of data for testing these kinds of ideas [11]. Since this isn't about physical parts getting rusty, the study focuses on "demand not served", basically, how much of the city goes dark. It is also tricky because big, catastrophic failures don't happen every day. They are rare. Because the data is so lopsided, looking at basic accuracy doesn't tell you much. Saito and Rehmsmeier pointed out in 2015 that when you have a lot of "normal" days and only a few "disaster" days, you have to use specific metrics like precision-recall to see if the model is actually any good [22]. Reliability matters too. You don't just want the model to guess right; you want to know how sure it is. Since neural networks can sometimes be a bit too confident in their guesses, as Guo showed in 2017, this study uses things like Brier scores to make sure the probability numbers actually mean something [10].

1.3 Research Objectives: The primary goal of this work involves building and testing a temporal graph learning framework to spot when interconnected energy grids are at high risk of a major breakdown. It centers on an early-warning system at the graph level, using snapshots of operational states over time to figure out if a grid is sliding toward a cascading failure. The core logic here is that grid risk doesn't just happen; it is shaped by how the network is built and how things change minute by minute. A model that looks only at the network structure might miss the slow buildup of stress, while a model that looks only at time could miss how a flicker in one spot jumps to another. This isn't meant to toss out traditional physical power analysis, but rather to see if data-driven temporal graph learning can act as a reliable extra layer for catching trouble early. Another aim is to define "degradation" in a way that is honest about the data being used. Since this study doesn't have access to actual maintenance logs, the age of equipment, or physical wear-and-tear measurements, it treats degradation as "operational risk." This means the model picks up on signals like stress levels, unmet power demand, and how vulnerable the system is to a chain reaction of failures. By focusing on these observable patterns rather than guessing at how much a specific transformer has aged, the work stays grounded. It connects the idea of a system breaking down to the actual, measurable stress seen in a networked grid.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 2, 2026, Pages 1137-1146

There is also a push to put temporal graph neural networks up against several different types of models. It isn't enough for a model to be fancy or complex; it has to actually work better than the simpler stuff. The study looks at standard tabular models, versions that only track sequences, and models that look at the graph but ignore the clock. This comparison is vital because it proves whether factors like the network layout, the specific details of the connections, and the history of the data actually make the predictions more accurate. Finally, the research looks at how reliable these models are in situations where real-world decisions need to be made. It checks how well the models classify risks, how they handle different thresholds, and where they tend to trip up. A lot of weight is put on "false negatives" because, in a power grid, missing a real threat is much worse than a false alarm. The end goal is a clear, repeatable process that can eventually be scaled up to larger grids or used with actual utility data and physical constraints once those are available.

1.4 Contributions: This research adds a few specific pieces to the puzzle of modeling grid failures. For one, it lays out a full, repeatable pipeline that uses PowerGraph cascading-failure data to track operational risk. It covers everything from the initial data check and building the graph to how the data is scaled and how the models are eventually picked apart for errors. Many studies in this field get caught up in the math of the model and gloss over the "boring" parts, like data cleaning or how the training and testing sets are split. This work treats those steps as just as important as the model itself. The study also offers a careful way to define what a "severe risk" actually looks like. Instead of just picking a random number, it uses the amount of demand that goes unmet and tests different cut-off points to make sure the results aren't just a fluke of the definitions. It also looks at the problem in two ways: as a simple "yes or no" risk and as a continuous scale, which gives a better sense of how bad a failure might actually get. Another contribution is the structured way in which the different model families are compared. By testing classical models against those that only look at time or only look at the graph, it becomes clear exactly where the value is being added. This approach avoids the trap of assuming a more modern model is better just because it is more popular. It forces the temporal graph model to prove it can do things the others can't. There is also a specific focus on "edge-aware" modeling. In a power grid, the lines between points (the branches) are just as important as the points themselves. They carry information about flow and electrical limits that are often ignored in simpler models. By including these edge-level details, the framework captures the actual pathways that cascading failures use to spread across a system. Lastly, the evaluation design is built to be rigorous. It pays close attention to the balance of the data, checks if the predicted probabilities are actually trustworthy, and uses a chronological split to make sure the model isn't "cheating" by seeing the future during training. It is open about its own boundaries, too, specifically distinguishing between the risk of the system failing and the physical aging of parts. This honesty makes the findings more solid and easier for others in the field to take seriously.

2. Literature Review

2.1 Predictive Maintenance in Power Systems: Keeping up with repairs before things actually break has turned into a massive focal point for anyone looking at modern industrial setups. The old way of just waiting for something to fail and then fixing it is just too expensive and messy, and it usually fails to stop the really bad, system-wide meltdowns. For a long time, power grid maintenance was mostly about checking things on a calendar or following basic engineering rules. These routines still matter, but they aren’t great at picking up on how parts of the grid actually wear down under real-world stress or how a problem in one spot might secretly be tied to a weakness somewhere else. Now that sensors and smart monitors are everywhere, things are shifting toward using all that data to figure out when a machine is actually starting to give out. In the world of power grids, the goal is to spot the likelihood of a part failing before the lights go out, which lets the people in charge fix the right things first and keeps the whole system from tripping.

Most of the standard ways of watching over equipment focus on the major things; transformers, lines, generators, and the like. It usually involves tracking things like heat, vibration, or weird shifts in voltage and oil quality. But there’s a bit of a flaw in the old-school thinking: it treats every piece of equipment like it’s living on an island. In reality, everything in a grid is wired together. If a transformer gets slammed with too much power or a line gets shaky, it might not even be because that specific part is old. It could be because the network around it is stressed, or the load just got shifted around weirdly. Because of these messy connections, there’s been a lot of interest lately in deep learning and machine learning. These tools are much better at digging through mountains of data to find the hidden patterns that humans might miss. Looking at recent industrial research, it’s clear that mixing different types of deep learning models is becoming the go-to move for predicting failures in big systems. A study by Alam et al. (2026) looked into these "hybrid" setups for industrial gear in the United States, proving that using a few different neural network styles together works better than just sticking to one when dealing with a bunch of different sensors [2]. This work points to a bigger trend where researchers try to look at everything at once, how things change over time, how unpredictable operations can be, and how different variables lean on each other, rather than just looking at a single stat in a vacuum. Even though that specific study wasn't only about the power grid, the lessons apply because any big infrastructure has these same kinds of complicated risks.Predicting maintenance needs in a power grid is extra tricky because electricity follows its own set of rules, and problems can move through the network in non-linear ways. Stress on the system moves around as people turn lights on or off, or as wind and solar power fluctuate. Because of this, the systems built to watch over the grid need to understand both the history of the equipment and how the whole network is laid out. Modern research is moving away from just labeling a fault after it happens and is instead trying to build early-warning systems. This has led to a lot of experimentation with things like graph-based modeling and temporal sequence learning. There are still big hurdles, though. There isn’t a ton of data on actual failures because they don't happen every day, and the data that does exist is often lopsided. Solving these issues is still the main task for anyone trying to build a reliable warning system for the grid.

2.2 Cascading Failures and Grid Stability: Cascading failures are basically the nightmare scenario for a power grid. It’s when a small, local glitch starts a chain reaction that travels through the network and ends up knocking out power for everyone. It isn't like a single wire snapping; it’s a series of events where one outage forces power to go elsewhere, which then overloads the next part of the system. If one line goes down, the ones nearby have to pick up the slack, and if they can't handle it, they trip too. This can keep going until huge chunks of the grid just give up. Since modern energy systems are so interconnected, a problem in one town can actually make things risky for a region hundreds of miles away just because they share the same network dependencies. Back in the day, research into blackouts started to show that these big failures aren't just random bad luck. They actually follow certain patterns seen in other complex systems. Dobson et al. (2007) looked at blackouts through the lens of complex systems theory and found that power grids often exist in a state where they are just stressed enough that a tiny nudge can cause a massive collapse [7]. This was a big deal because it moved the conversation away from just "is this one transformer okay?" to "how vulnerable is the whole web?" It means you can't really understand if a grid is stable just by looking at the individual parts. You have to look at the shape of the network and how all the different parts talk to each other. These cascades are tied to a few specific things, like voltage drops, lines getting too hot, and "Demand Not Served," or DNS. When a line gets cut, the surrounding area gets squeezed, and if there isn't enough backup power to keep the voltage steady, things start to fail. Operators sometimes have to intentionally cut power to certain areas just to stop the whole thing from crashing, which is where DNS comes in as a way to measure how bad a failure really is. It’s a better metric than just saying "the power went out" because it tells you exactly how much of the needed electricity wasn't delivered during the mess. The way we use energy now makes these cascades even harder to predict. With more solar panels, wind farms, and weird new load patterns, the grid is changing every minute. The old ways of calculating reliability just can’t keep up with how fast things move now. This is why there’s such a push for data-driven models that can spot a high-risk situation before the first domino falls. Using machine learning or graph-based tools might give operators the heads-up they need that a traditional analysis would miss. It’s still a tough nut to crack, though, because these massive collapses are rare, the system is always changing, and the way a local spark turns into a global blackout is incredibly complicated. That's why there’s such a need for models that can look at both time and network structure simultaneously.

2.3 Machine Learning for Power-Grid Failure Prediction: Machine learning is a big deal now for keeping tabs on power systems and figuring out when things might go south. This shift happened because modern grids are basically data factories, pumping out endless streams of info from sensors, smart meters, and control centers. Old-school stats often hit a wall here. They have a hard time dealing with messy, nonlinear interactions or the way things change over time in these massive systems. Because of that, there has been a lot of work into using different algorithms to spot instability, weird operating states, or equipment that is about to quit. In the beginning, most of it was supervised learning. This meant using things like logistic regression, decision trees, or random forests to flag risks. These tools are still used today because they don't need a ton of computing power, they are easy to explain, and they work well with the kind of structured data grids usually produced. As smart-grid tech and huge datasets became the norm, the methods got a bit more intense. In 2016, Wang et al. looked into clustering methods to understand how people use electricity [26]. They showed that you can take mountains of usage data and find actual patterns that make sense for big-data tasks. This study was important because it proved that what happens in a power system isn't just a bunch of random, isolated moments. Instead, it moves in dynamic patterns over time. Even though they were looking at consumption rather than how a grid collapses, they proved that machine learning is great at finding hidden structures in really complicated data. Deep learning has pushed things even further. Now, there are recurrent neural networks and hybrid setups being used to guess how much wind or solar energy will be available or to catch faults before they cause a blackout. In 2025, Debnath et al. came up with an AI-driven hybrid model specifically for predicting renewable energy levels during nasty weather [6]. This is a big help because renewables are unpredictable and make modern grids jumpy.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 2, 2026, Pages 1137-1146

Having a system that can learn and adapt to these weird, nonlinear timing patterns is pretty much essential now.Even with all these cool tools, a lot of research still treats the grid like a big spreadsheet. Many models look at the data in chunks without really thinking about how everything is actually plugged together. This is a problem for catching cascading failures. If you don't account for the layout and how parts are connected, you miss how the risk travels from one wire to the next. Also, a lot of these models are obsessed with being "accurate" in a general sense but ignore things like timing errors or the fact that major failures are rare compared to normal days. Missing a high-risk state is a much bigger deal than getting a few normal days wrong. Because of this, there is more interest in mixing machine learning with graph-based methods that actually "see" the network and how it changes over time.

2.4 Graph Neural Networks for Power Systems: Graph neural networks are becoming the go-to for modeling systems where the connections are just as important as the parts themselves. Most standard machine learning looks at data points as if they are totally independent, but GNNs are built to handle data that looks like a web. This fits power grids perfectly. You can think of the buses as nodes and the transmission lines as the edges connecting them. In this setup, how one part of the grid behaves depends almost entirely on what its neighbors are doing and how the whole thing is wired together. Graph-based learning lets information flow through the model the same way electricity flows through the wires. The groundwork for this was laid back in 2009 by Scarselli et al. [23]. They created one of the first general models that could learn from these structures by letting connected nodes "talk" to each other. They proved that you could use neural networks for things that don't fit into a neat grid or table. This was huge for everything that came later, like message-passing frameworks. In a power grid, this is vital because what happens at one station is always influenced by the stations nearby and the overall path of the lines.

Lately, more people are applying this specifically to power systems. In 2020, Donon et al. tested out graph neural solvers and found they could handle grid tasks pretty efficiently [9]. They showed that these architectures naturally match the way a grid is built, which makes them better for large-scale systems than models that need a bunch of hand-made summaries of the data. Instead of trying to turn the grid into a list of numbers, GNNs just look at the map itself. GNNs are especially good for predicting how a small failure might turn into a massive blackout. In a grid, things like overloaded lines or voltage drops don't happen in a vacuum; they spread through the physical connections. Message passing lets the model learn how stress moves from one component to another across the map. These models can also juggle a lot of different types of info at once, like voltage at a station and the flow on a line. There are still hurdles, though. Really big grids can be hard to compute, and a model trained on one city might not work on another. Also, a lot of these models are "static," meaning they see the map but not how it evolves every second. That is why the focus is shifting toward temporal graph neural networks that can track the structure and the timing at the same time.

2.5 Temporal Graph Neural Networks: Standard graph learning usually stays stuck in a single moment, but temporal graph neural networks pull in how things change over time. Regular graph neural networks do a fine job at mapping out how nodes relate to one another, but they struggle with real-world systems that never actually sit still. In a power grid, things are always shifting. Demand goes up and down, renewable energy kicks in at different rates, equipment gets stressed, and control systems make constant tweaks. Because of this, any model trying to predict what happens next has to look at more than just the way wires are connected. It has to track the way the whole network's state moves from one minute to the next. Temporal graph neural networks handle this by blending relational learning with sequence modeling. Architectures built for learning sequences, like recurrent neural networks and gated recurrent units, have been the backbone of this kind of work for a while. Back in 2014, Cho et al. came up with an encoder-decoder setup using gated recurrent units [5]. It showed that these gating mechanisms could pick up on long-term patterns without the math falling apart during training. This approach stuck because it was a fast, efficient way to deal with data that evolves. For power grids, this matters quite a bit. Stress on the system doesn't usually just appear out of nowhere; it builds up slowly. Following the sequence of events is the only way to spot the tiny patterns that hint at a future crash or a failure.he next logical step was putting graph learning and sequence modeling into one box, which gave us spatiotemporal graph architectures. Yu et al. (2018) worked on this for traffic forecasting [28]. They used graph convolutions for the spatial side and temporal convolutions for the time side. Even though they were looking at cars and roads, the logic fits power grids perfectly. Both are just big networks of infrastructure where what happens in one spot depends on what's happening nearby and what happened a few seconds ago. Their research proved that looking at space and time together works much better than trying to solve for them separately. In the world of power systems, these temporal models have a clear edge. A static model might point to a weak spot in the grid's physical layout, but it won't see how the actual electricity flowing through it is changing. On the flip side, something like an LSTM might track the flow well but would be totally blind to the actual physical map of the wires. Temporal graph neural networks bridge that gap. They are great for watching how an overload spreads or predicting a cascading failure. It isn't easy, though. Electrical systems are messy and non-linear, and the really big, dangerous events don't happen very often. Plus, the way a grid behaves can change entirely from one year to the next, which is why people are still trying to find more reliable ways to make these models work across different types of infrastructure.

2.6 Identified Research Gaps: Even with all the progress in machine learning, there are still some big holes in how interconnected power grids are modeled. One of the main problems is that a lot of maintenance research still looks at parts in isolation. It treats a transformer or a line like it's living on an island. These models check local sensors but ignore how stress moves through the rest of the network. This is a huge oversight when it comes to cascading failures. In those cases, the health of one piece of equipment is tied directly to what's happening miles away because of how power gets rerouted when something breaks. Another gap is the issue of "non-stationarity"; the fact that the rules of the game keep changing. Infrastructure doesn't operate in a vacuum. Weather shifts, people change how they use power, and new energy sources like wind and solar get added to the mix. Bhowmik et al. (2025) talked about this in the context of finance, showing how shifting data distributions can make a model totally unreliable [3]. The same thing happens in power grids. A model trained on data from five years ago might get completely confused by how the grid runs today. There is a real need for systems that can adapt on the fly rather than staying stuck on the data they were originally fed. Then there is the problem of how risk actually spreads. Many models just look at the big picture without mapping the specific paths a failure might take. Islam et al. (2025) used graph neural networks to look at how a crash in the banking world could bleed into the crypto market [13]. It showed that the way things are connected is often more important than the individual parts. In a power grid, a failure isn't just a random event; it's a ripple effect that moves through the physical structure of the network. Most current models still haven't fully captured this "contagion" aspect.

Finally, there's a lack of focus on how reliable these early warnings actually are. Most researchers just talk about high accuracy scores, but they don't always look at whether the model is giving enough lead time or if it's crying wolf too often. Rahman et al. (2025) pointed this out while looking at financial crises, noting that the cost of a missed warning is much higher than the cost of a false alarm [18]. Power grids have the same problem. These massive failures are rare, so the model has to be incredibly precise when it finally does flag something. There is also the issue of moving a model from one grid to another. A setup that works for a small city might fail in a huge metro area because the layout and the scale are totally different. There's still a lot of work to do to make these tools useful across different types of systems.

3. Methodology

3.1 Research Framework and Problem Formulation: This work uses a quantitative experimental setup to spot the risk of nasty cascading failures in power grids by looking at things through the lens of temporal graph learning. The whole approach rests on a simple idea: watching individual parts of a power grid isn't enough to see trouble coming. Problems in an energy network don't just sit still; they crawl through the connections and change as the grid's operating conditions shift over time. Because of that, this framework mixes graph learning with sequence modeling to keep track of how parts of the grid relate to each other and how those relationships change as the clock ticks. The focus here is on supervised learning at the graph level. It uses snapshots of grid states, taken one after another, from the PowerGraph cascading-failure benchmark. There are two main jobs for the model. First, it handles a binary classification task to guess if a future state is going to turn into a serious failure. Second, it tackles a regression task to estimate the "demand not served," which is just a fancy way of saying how much of the lights-out effect there might be. These two tasks work together to show the full picture. One flags if a high-risk situation is happening, and the other puts a number on how bad the mess will be if things go south. In this context, degradation isn't about a physical part getting old or rusty. It's about the grid getting more stressed and more likely to collapse under pressure. This matters because the data doesn't have notes on maintenance, thermal wear, or how much life is left in a specific transformer. Instead, the data shows simulated states, electrical readings, and how things are hooked up. So, degradation is treated as a fast-moving, network-wide problem where overloads spread, and instability grows from one moment to the next. To make sure the results actually hold up, the methodology uses chronological splitting, careful preprocessing to stop data from "leaking" into the future, and plenty of stress tests like ablation studies and cross-grid checks. Keeping things realistic in terms of time is the big priority. An early-warning system is useless if it's trained on a random jumble of data; it has to be able to look at the past to guess what happens next. That's why the data is split into training, validation, and test sets based on time, not at random. The end goal is to see if these temporal graph neural networks can actually do a better job of predicting failures by looking at the whole system, the nodes, the lines, the electricity, and the timeline, all at once.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 2, 2026, Pages 1137-1146

3.2 Dataset Description and Graph Representation: The testing happens on the PowerGraph benchmark, which is basically a collection of power system states and the failure results that go with them. It covers several different grid setups, like the IEEE24, IEEE39, and IEEE118, plus some UK grid data. Most of the heavy lifting in this study is done with the IEEE24 system. It's a good middle ground: small enough to run experiments without a supercomputer but complex enough to show how real cascading failures behave. The other grids come into play later to make sure the findings weren't just a fluke on one specific map. The PowerGraph data lives in MATLAB files that pack in everything: node features, edge features, the map of the grid, and the failure labels. For the nodes, which are the buses in the system, there are readings for active power, apparent power, and voltage. The edges, or the transmission lines, have their own data, like power flow and line ratings. The grid's layout is stored as an edge list showing what's connected to what. Before the model gets to see any of this, the indexing has to be shifted from MATLAB's 1-based system to the 0-based system that Python libraries prefer. The dataset gives a few different ways to measure a crash. There are simple "yes or no" labels for failures and continuous numbers for demand not served. That second one is key because it tells you if the grid just had a hiccup or if the whole thing went dark. There are also "explanation masks" for the edges involved in a failure, which might be helpful for future work on figuring out why the model made a certain call, even if they aren't the main focus here. Every snapshot is basically a still frame of the grid's health. Since this is all simulation data, these frames aren't seen as physical wear and tear. Instead, they are evolving stages where stress and overloads start to bubble up. This setup lets the temporal graph learning do its thing, looking for risks as they develop. The final graph representation used here bundles the node and edge data with the grid's structure into windows of time, catching both the layout of the system and the way it changes as time passes.

3.3 Data Preprocessing and Temporal Window Construction: The way the data is handled focuses on making sure the results can be repeated and that the timing makes sense without accidentally letting future information leak into the past. First, it takes the graph files from MATLAB and turns them into arrays that work well with Python-based learning tools. Since the original files started numbering nodes at one, everything is shifted to start at zero. It has to be done this way so the math for the graph connections and convolutions works correctly in the code. Once the graphs are loaded, a close look at the data happens. This audit checks for things like how the features are spread out, if any values are missing, or if some classes show up much more than others. Statistics for nodes, edges, and power shortages are all calculated before any training starts. These power shortage numbers, or demand-not-served, are really the heart of the labeling process. A severe failure is defined as any moment where the power shortage is in the top 10 percent of the data. Other cutoffs, like the top 20 percent or just any shortage at all, are also checked to make sure the results don't just happen because of one specific choice of a threshold.To make this work with time, the data is sliced into sliding windows. For any single prediction, a sequence of past graph snapshots is bundled together as the input, and the goal is to guess what happens in the next step. Usually, 12 snapshots are used to look at the immediate future. This gives the models enough history to see how stress builds up in the system. To be thorough, other window sizes like 4, 8, or 24 are tested too, just to see if having a longer memory actually helps the predictions. The data is split up by time, not randomly, to keep things realistic. The first 70 percent of the timeline goes to training, the next 15 percent is for checking the model, and the final 15 percent is for the final test. Doing it this way mimics how a real-world warning system would have to work. Even the scaling of the data follows this rule. The math for normalizing features is only figured out using the training set and then applied to the rest. Different types of features, like those for nodes or edges, get their own scaling to keep the training steady. Besides the raw graph data, some extra statistics are calculated to help compare the complex models against simpler ones. These include things like the average or minimum voltage, how much the voltage swings, total power, and how hard the lines are being pushed. These markers summarize what is happening across the whole grid, making it easier to see if the fancy graph-based models are actually doing something better than standard math.

3.4 Baseline Models andTemporal Learning Architectures: Different types of models are used to see if actually looking at the grid layout helps more than just using basic statistics. These models were picked to figure out exactly what matters more: the history of the system, the physical connections, or the electrical details on the lines themselves. The first group uses classic machine learning on those summary statistics. There is a logistic regression, which is just a simple, clear way to draw a line between a failure and a normal state. Random forests are included because they are good at finding patterns that aren't just straight lines. Then there is XGBoost, which is great for handling lopsided data where failures don't happen often. Finally, a basic neural network is used on the summarized data to see if a standard "brain" can find patterns without knowing anything about the grid's shape. The next group looks at time but ignores the grid shape, or vice versa. One model uses an architecture called an LSTM to look at sequences of summary stats. This tests if just knowing the sequence of events is enough. On the flip side, a basic graph convolutional network is used that sees the grid shape but only looks at one moment in time. This shows if the layout matters, even if you ignore the history. The main models created for this work combine both. They use graph convolutions to understand how nodes relate to their neighbors, creating a sort of "map" of the grid's current state. These maps are then fed into a recurrent unit that tracks how that state changes over time. In the end, the model outputs a probability that a big failure is about to happen. An extra version of this model was built to look specifically at the edges, or the power lines. It pulls in data like power flow and line limits, mixing that info with the node data before it looks at the timing. This is a big deal for power grids because a failure in one line often puts too much pressure on another, causing a chain reaction. All these deep learning tools are built using PyTorch. They use a special loss function that pays extra attention to the rare failure events, so the model doesn't just get lazy and predict "no failure" all the time. For the versions that predict the exact amount of power lost, standard regression math is used. Everything is optimized with a method called AdamW, and the process stops early if the model stops getting better on the validation data, which helps keep the training from going off the rails.

3.5 Evaluation Strategy, Ablation Analysis, and Reproducibility: Testing this system meant looking at both how well it predicts trouble and whether it can actually be trusted in a real control room. Because massive grid failures don't happen every day, the usual metrics weren't enough. The focus shifted toward tools that handle lopsided data well. Performance is tracked through ROC-AUC, F1-scores, confusion matrices, and Brier scores, but the real weight is put on precision-recall curves. PR-AUC gives a much clearer picture when the "bad" events are rare. High recall is also non-negotiable here, as a false negative essentially means missing a high-risk state that could spiral into a total blackout. It was also important to check if the probability numbers the model spits out actually mean anything. That's where calibration curves come in. A model might be great at ranking risks but still be dangerous if its percentages are way off from reality. To keep things honest, the "alert" thresholds were set using only the validation data. The F1-score was optimized there, and then that exact setting was used just once on the final chronological test set. This keeps the results from looking better than they really are by avoiding any "peeking" at the test data.Predicting the actual amount of power lost, the continuous demand-not-served, required a different set of yardsticks, like mean absolute error and R^2. This part of the math is vital because it moves beyond a simple "yes or no" and starts quantifying how bad the damage might be. A deep dive into the errors followed, looking specifically at where the model was confidently wrong. In an early-warning setup, the goal is to stop missing the big risks while keeping the "crying wolf" false alarms at a manageable level. To see what was actually doing the heavy lifting, the model was taken apart piece by piece in an ablation study. Tests were run with and without the grid's physical layout, the historical timing, and the specific line features. The sensitivity to time was checked across windows of 4 to 24 snapshots, and the failure thresholds were tested at different severity levels, specifically the 80th, 90th, and 95th percentiles. By testing across the IEEE24, IEEE39, IEEE118, and UK grid models, it was possible to see how the system handles different types of network shapes and sizes [4].Finally, the whole process was locked down to make sure anyone else could get the same results. Random seeds were fixed, every setting was saved, and the data was double-checked with SHA256 hashes to ensure nothing shifted during the process. By controlling the exact versions of the software and the way the data was split, the experiments stay consistent every time they run. This focus on a clean, open pipeline is there to remove any doubt about hidden tweaks or lucky guesses in the preprocessing [5].

4. Results and Evaluation

4.1 Dataset Audit Results: The first look at the IEEE24 PowerGraph dataset made it clear that the graph snapshots were solid and ready for temporal graph learning. Everything was structurally consistent. There were 21,500 snapshots of the system in action, mapped out as graphs with 24 nodes and 38 edges. Each node carried three specific electrical markers: net active power, apparent power, and voltage magnitude. The edges had four bits of branch data: active and reactive power flow, reactance, and the line rating. The demand-not-served (DNS) numbers ranged from zero all the way to 0.672105 MW. This confirms the data covers the whole spectrum, from a smooth Tuesday morning to the kind of cascading failures that actually bring things down. It also became obvious that the classes are heavily skewed when it comes to predicting those nasty failures. About 20.16% of the snapshots showed some kind of failure, but the really bad ones, the severe cases based on the 90th percentile DNS, only showed up 1.86% of the time in the test set. It is a needle in a haystack. This makes it an early-warning task where the target is rare. Because of that, looking at precision-recall and weighted losses matters much more than just checking if the overall accuracy looks good. Accuracy is a bit of a trap here.

The cleanup process didn't find any missing values in the electrical features. Since nothing was missing, the work stayed simple because there was no need to guess or fill in gaps before training. Even so, the numbers for power flow and DNS swung wildly depending on the state of the system. This meant that scaling things properly was still a big deal to keep the optimization from getting messy. Most of the DNS values were low, with the catastrophic outcomes tucked away in a tiny corner of the data. This really highlights why threshold sensitivity is the main event here. The events that matter most are the hardest ones to actually catch. The physical layout of the graph didn't change during these tests. The wires and nodes stayed where they were. What changed was the electricity moving through them over time. This is a useful setup because it lets the temporal graph models focus purely on how stress evolves without having to worry about the actual grid shifting around. The audit basically proves that PowerGraph is a realistic sandbox for testing how well models can spot risk when the odds are stacked against them.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 2, 2026, Pages 1137-1146

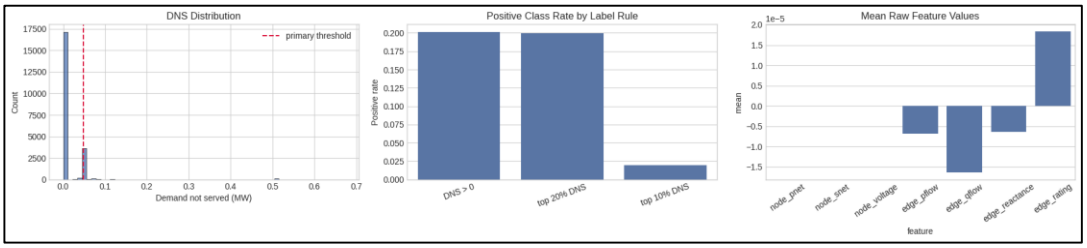

Fig.1: IEE24 initial dataset audit trials

4.2 Baseline Model Performance: The baseline tests were designed to see if just looking at the total stress on the system would be enough to predict a failure without actually digging into the grid's shape. Logistic regression, random forests, XGBoost, and a multilayer perceptron were put to work. These used summarized features like voltage trends, power flow, and how many lines were getting overloaded across the whole grid. Out of that group, the random forest did the best job with the trade-off between precision and recall. It hit a PR-AUC of 0.1292 and an F1-score of 0.1680. XGBoost actually had a slightly better ROC-AUC at 0.8665, but its PR-AUC was lower at 0.0996. It just wasn't as good at picking out those rare, extreme events. Logistic regression stayed in the same ballpark for ROC-AUC, but it struggled with the F1 and PR-AUC. It seems that simple linear boundaries just can't map out the complicated, messy risks in this dataset. The multilayer perceptron was actually the weakest on precision-recall, even though it was decent at calibrating probabilities. Being "calibrated" didn't necessarily mean it was good at finding the failures. One thing that jumped out was the massive gap between the ROC-AUC and PR-AUC for every single model. This is the class imbalance showing its teeth. The models are actually pretty good at telling general states apart, which is why the ROC looks fine. But when it comes to pinpointing those few severe failure states, they hit a wall. Those low PR-AUC scores show how tough it is to separate a system that is just stressed from one that is about to collapse. Still, these basic stress stats do tell a story. Things like voltage instability and line loading gave the models enough of a signal to get ROC-AUC scores above 0.84 across the board. The downside is that they also flagged a lot of false alarms. It suggests that high stress might mean the system is vulnerable, even if a total blackout doesn't happen right then. In the real world, these false positives might actually be helpful. They work as a heads-up for an operator to take a closer look before something actually breaks. In the end, these tests proved two things. First, general grid stress is a good indicator of risk. Second, standard tabular models have a hard time with the extreme imbalance of this problem. This is exactly why it makes sense to move toward models that can actually "see" the grid structure and understand the relationships between the nodes.

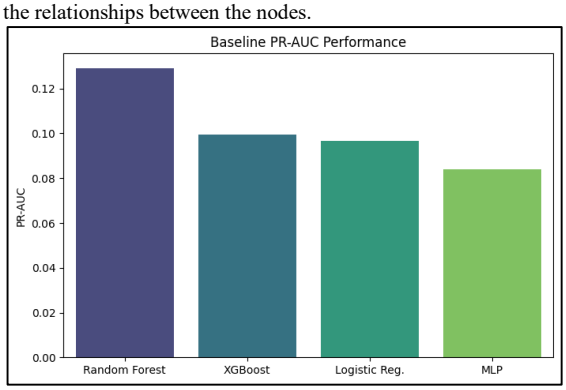

Fig.2: Baseline Model Performance (PR-AUC Comparison)

4.3 Temporal and Graph Model Performance: The temporal and graph-learning experiments looked into whether using graph topology and temporal history actually helps predict bad cascading failures better than just using basic tabular data. Four different setups were tested: a basic LSTM that just looked at aggregate stats over time, a static graph convolutional network (GCN) that only saw a single snapshot of the graph, a GCN-GRU that combined time and graphs, and an edge-aware GCN-GRU that added in specific branch-level electrical details. The static GCN ended up with the highest ROC-AUC of all the graph models at 0.8831, along with an F1-score of 0.1950. It seems that just knowing the graph structure provides a lot of useful risk info even without tracking changes over time. Since it beat the non-graph baselines, it shows that the physical layout of the grid is a big deal for how failures happen. These events move through physical connections in the network rather than just happening to isolated parts, so the GCN's ability to pull data from neighboring nodes helped it get a better handle on where the system was vulnerable.The edge-aware GCN-GRU had the best PR-AUC at 0.1019. It also hit an ROC-AUC of 0.8703 and an F1 of 0.1776. Even if the ROC-AUC was a tiny bit lower than the static GCN, the better precision-recall shows that mixing topology, time, and branch-level electrical data is better for spotting those rare, extreme "tail events" than just looking at sequences. This matters because big cascading failures don't happen often. Because of that rarity, PR-AUC is a more useful way to tell if an early-warning system would actually work in the real world. The standard GCN-GRU got a PR-AUC of 0.0980 and an ROC-AUC of 0.8656. This showed that learning from a temporal graph is still a solid approach even if edge features aren't specifically included. On the other hand, the LSTM that only looked at aggregate summary stats was the worst of the bunch, with a PR-AUC of 0.0817 and an F1 of 0.1382. It turns out that just watching things change over time isn't enough to predict a collapse if the actual structure of the network is ignored.

One thing that stood out is that topology always made the predictions better. Every graph-based model did better than the LSTM in ROC-AUC, which confirms that you have to account for how everything is connected to judge risk in a power system. But the differences between the static GCN and the GCN-GRU models suggest that temporal modeling is a bit more complicated. Looking at time windows helped show how stress builds up, but the biggest jumps happened when that timing was paired with edge-level electrical data rather than just node data. These results back up the main idea: you need topology-aware learning because risk flows through the network structure. That said, the PR-AUC improvements weren't massive. It shows that predicting these extreme events is still a very hard problem for any architecture, mostly because these states are so rare and can pop up out of nowhere without much warning.

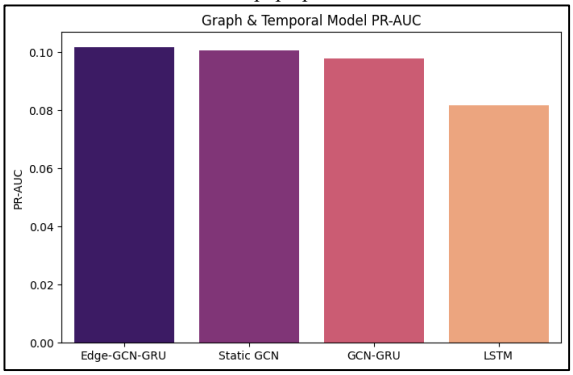

Fig.3: Temporal and Graph Model Performance

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 2, 2026, Pages 1137-1146

4.4 DNS Regression Results: The regression tests were done to see if models could guess the actual amount of demand not served (DNS) instead of just flagging a failure. This was a secondary goal meant to give more detail on how bad a disruption might be when a cascade starts. Looking at the baselines, ridge regression had the lowest RMSE at 0.0421, with an MAE of 0.0164 and an R^2 of 0.0666. The XGBoost regressor was right there too, with the lowest MAE at 0.0156, an RMSE of 0.0422, and an R^2 of 0.0631. Random forest was a bit further back with an RMSE of 0.0431 and an R^2 of 0.0216. While the error numbers weren't huge compared to the scale of the DNS, the R^2 values were all pretty low. A lot of the reasons why the DNS fluctuates so much just weren't being caught by the models. It seems that guessing the exact size of a failure is much harder than just figuring out if the system is in a high-risk state. The way DNS grows likely depends on really messy, nonlinear interactions and network effects that are hard to pin down just by looking at the early stages. Tiny shifts in how lines are loaded or what the voltage looks like can cause huge, lopsided outcomes. That makes it tough for a continuous regression model to stay stable and accurate. These findings also point back to why the classification-style early warning is so important. Knowing for sure that the risk is high is probably more useful for someone making decisions than getting a specific, but potentially shaky, estimate of the final DNS. In that sense, the regression part is more of a secondary tool for gauging risk rather than the main goal of the whole setup.

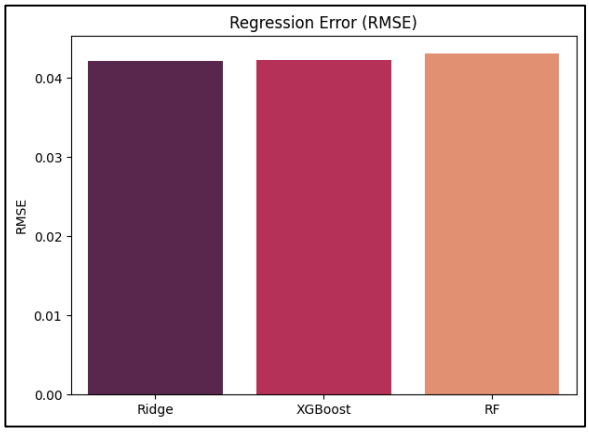

Fig.4: DNS Regression Results (RMSE)

4.5 Calibration and Reliability: It was necessary to run a calibration analysis to see if the predicted probabilities actually matched up with how often severe failures really happened. This matters quite a bit for early-warning systems used in the field. If a model isn't well-calibrated, it might spit out overconfident or just plain confusing probability numbers, even if it seems good at ranking risks on paper. The top-performing baseline models turned out some decent Brier scores. For instance, the random forest classifier hit a score of 0.0175, which shows it is generally reliable when looking at the test data as a whole. However, a closer look at the calibration curves showed a habit of overestimating the chance of a severe failure in areas where the actual risk was low. This isn't surprising given how lopsided the dataset is; when severe events are this rare, getting the probability exactly right is a lot harder than just figuring out which scenarios are riskier than others.

The graph-learning models were a bit more hit-or-miss with their calibration. The edge-aware GCN-GRU was better at tellling the difference between classes than a few other models, but its Brier score still couldn't beat the best random forest baseline. This suggests a bit of a trade-off: using the network's layout to learn over time helps spot risks, but it also adds some extra fuzziness to the probability estimates. In a real control room, that balance is a big deal because a useful tool needs to be good at both catching events and giving a number you can trust. The analysis also made it clear that a high ROC-AUC score doesn't automatically mean the probabilities are ready for prime time. Quite a few models had strong ROC-AUC results but still produced messy calibration curves because of the extreme imbalance in the data. This confirms why checking for calibration is a must before saying a system is ready for the real world. While these setups are good at flagging high-risk moments, they probably need more fine-tuning before they can be used to make live decisions.

4.6 Error Analysis: Looking at the mistakes provided some useful context on what it's actually like to predict these cascading failures. The best random forest model caught about 35% of the severe failures in the test set. While that proves these high-risk states can be spotted by looking at stress patterns, it also serves as a reminder of how tough it is to reliably catch these rare, extreme events. False positives were a frequent issue for both the old-school and the graph-learning models. Plenty of snapshots were flagged as risky even if they didn't end up crossing that 90th percentile threshold for loss. If you look at this strictly as a math problem, these errors hurt the precision and the PR-AUC. But in the context of a power grid, these "wrong" guesses might actually be useful. They often point to high-stress situations that look a lot like the moments right before a failure. So, instead of being total misses, they might work better as heads-up warnings.

False negatives were the bigger worry since these represent severe failures that the models missed entirely. When these events were examined, it turned out many happened during sudden spikes where the failure escalated so fast there was no long buildup in line-loading or general stress. This suggests some cascades are set off by tiny local weaknesses or specific tipping points that are hard to see if you only look at the big picture. The errors also showed some gaps in how the models learn over time. Longer windows of data usually helped, but some failures just seemed to appear out of nowhere without any obvious warning signs in the previous steps. Even the models that understand the network structure can struggle when a collapse happens through sudden, localized shifts. This leads back to the idea that predicting these failures is just naturally hard because the way stress, topology, and outages interact is incredibly complex.

4.7 Ablation Results: The ablation experiments looked into how picking apart the model affected its ability to predict things. It was a deep dive into what happened when the temporal history, the way the nodes connect, the edge features, or the way thresholds are defined were messed with. These tests mattered because they pinpointed which parts of the setup actually did the heavy lifting. It helped settle the question of whether adding graph learning and time-based tracking really made a difference in the end results. Looking at the temporal windows, things generally got better when the input sequence grew from 4 up to 12 snapshots. This suggests that the danger of a system falling apart builds up over time, so the models do better when they can look back at a longer stretch of stress history before they have to guess what happens next. That said, the improvements started to flatline at 24 snapshots. It seems like having a window that is too long eventually hits a point of diminishing returns. Maybe older data just isn't that useful for predicting a crisis that is about to happen, or perhaps the extra data just adds too much noise and mess for the model to handle. The analysis of thresholds showed that trying to predict a truly massive failure gets a lot harder as the DNS threshold climbs. When the goal was to catch events in the 80th percentile, the PR-AUC scores were around 0.48. But when that target moved to the 90th percentile, the scores took a dive toward 0.12. It turns out that those extreme, "tail-end" cascading failures are way tougher to spot than just some moderate operational trouble. This confirms why it was smart to test out a bunch of different thresholds instead of just picking one definition of a "bad" failure and sticking with it. Systems that actually understood the topology always beat out the ones that didn't when looking at ROC-AUC. It proves that knowing how things are physically connected is a huge deal for predicting risk. The graph convolutional setups picked up on relationships that just can't be found by looking at averages or basic statistics. Adding in the edge-aware temporal GCN-GRU showed that knowing what's happening at the electrical branch level helps even more when trying to separate out the serious events from the noise. It really drives home the point that how a failure spreads depends on how specific lines act and how neighbors interact under pressure. Comparing the static and temporal setups led to a few more realizations. The static GCN actually had the best ROC-AUC, which says the map of the grid itself holds a ton of predictive power. However, the model that combined temporal changes with detailed branch info got a better PR-AUC than the others. It seems like the way a system evolves over time becomes much more meaningful when paired with specific electrical data. Taken as a whole, the ablation work confirmed that the layout, the timing, and the line-level behavior all matter for getting the predictions right.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 2, 2026, Pages 1137-1146

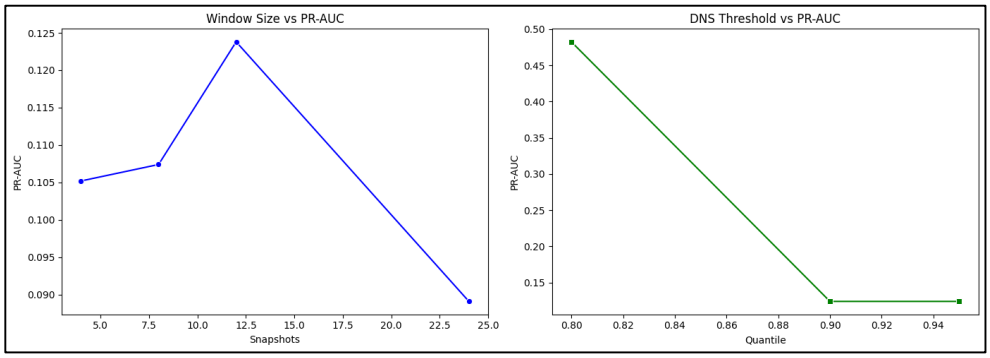

Fig.5: Ablation Studies: Sensitivity Analysis Results

4.8 Cross-Grid Robustness: The cross-grid audit was basically a reality check across different systems: the IEEE24, IEEE39, IEEE118, and the United Kingdom PowerGraph. These four grids are pretty different in terms of size, how they operate, and how they fail. IEEE24 used 21,500 snapshots with 24 nodes and 38 edges, while IEEE39 was a bit bigger with 28,000 snapshots. IEEE118 was the giant of the group, featuring 122,500 snapshots and 186 edges. Meanwhile, the UK grid sat at 29 nodes but had a high density of 99 edges. Even the maximum DNS numbers were all over the place, ranging from about 0.2047 MW in the IEEE118 up to 0.9000 MW in the UK setup. These gaps show that no two grids really act the same when things go wrong. This creates a headache for transfer learning. A model that learns everything there is to know about one layout might totally fumble when it's moved to a system with different connections or a different way of handling stress. The fact that the peak DNS in the IEEE118 was so low compared to the UK grid suggests that the way failures move through the system isn't just a matter of scale, the actual physics of the propagation might be different. Because of this, the audit suggests that moving a model from one grid to another isn't a simple copy-paste job. It's more of a tough domain-adaptation problem. A model trained only on the IEEE24 isn't going to just work on a larger, more complex system without a lot of tweaking, retraining, or recalibration. This is a big deal because it points out a major hurdle in using standard benchmarks for power systems. Just because a model is a rockstar on one grid doesn't mean it's ready to handle the reliability needs of every other network out there. Even so, the fact that the graph-learning framework stayed structurally consistent across these different grids is a good sign. It shows that temporal graph models are still a strong bet for predicting risk at scale. The results point toward a future where the focus shifts to things like better calibration, transfer learning, and finding ways to make these models less sensitive to the specific layout of a network so they can work across all kinds of interconnected systems.

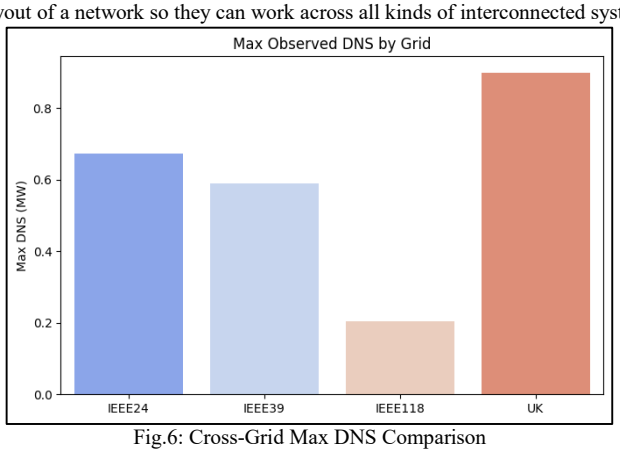

Fig.6: Cross-Grid Max DNS Comparison

5. Discussion

5.1 Interpretation of Findings: The test results show that learning based on topology offers a real leg up when trying to spot big cascading failures in linked power grids. In all the tests, models using graphs did a better job with ROC-AUC than the ones just looking at sequences or basic tables. It turns out the way things are connected carries a lot of weight for operational risk, and you just can't get that by looking at overall stress numbers alone. This matters a lot because these failures don't happen in a vacuum; they spread through relationships between parts rather than one piece breaking on its own. The static graph convolutional network actually had the best ROC-AUC of the bunch. This suggests the way the network is organized is a huge clue for where it might be weak. Meanwhile, the temporal graph model that looked at edges handled precision-recall better than many others. It seems like mixing the layout, how things change over time, and the electrical details of the branches helps pick out those rare, high-risk moments much better.The data also points to time-based info helping with risk guesses, though it's a bit more complicated than just the layout. Stretching out the time window usually made PR-AUC better until about 12 snapshots, but after that, the improvements just kind of stopped. This tells us that being prone to a massive failure is something that builds up as stress grows over time, it isn't just a random strike of lightning. Still, looking at the errors showed that some big spikes in DNS happened out of nowhere without any long-term warnings. That makes it hard for any time-based model to catch every single bad event. It looks like grids can get unstable because of slow-building stress or because they hit a specific tipping point where the structure just gives way.Comparing the LSTM, which only looks at sequences, to the graph models really drives home why connections matter in these systems. The LSTM was okay at following trends, but it couldn't "see" the network layout, which capped how good its guesses could be. Graph operations, though, let the model learn from what's happening at neighboring spots. This helped it track how risk moves through the system. A study by Shawon et al. (2025) found something similar in finance, where tracking how money moves across chains was key to finding weird activity [24]. Even though they were looking at money and not power, the core idea is the same: in linked systems, how things spread across the network is what reveals the hidden dangers that looking at one thing at a time would miss. So, the big picture supports the main idea: to predict a huge failure, you need to look at the layout, how things change, and how the electrical branches are acting all at once. But, the PR-AUC numbers weren't massive, which shows that guessing these "black swan" events is still really tough when the data is so lopsided. This difficulty probably comes down to the messy, non-linear way these grids act when they get pushed too far.

5.2 Importance of Edge Features: The experiments focused on edges showed that the electrical details of the branches themselves are a big part of the puzzle for grid risk. Sure, things like voltage at a specific spot tell you something about the local area, but a cascading failure usually travels through the transmission lines as the power gets shoved around. Because of this, things like power flow, reactance, and line limits on the edges provide the "where and how" of stress moving through the system. The edge-aware GCN-GRU model did better at PR-AUC, which suggests that including these branch details helps the model spot dangerous states better than if it only looked at the nodes. One thing that stood out is that just having the graph layout wasn't enough to get the best results. The static graph network had a great ROC-AUC, but the edge-aware temporal version was better at precision and recall. This means the physical layout only tells half the story; you have to look at how the branches connecting everything are actually doing. A line that is carrying too much power or hitting its limit is a perfect path for a failure to spread, especially if the lines nearby are also feeling the heat. This is why edge-aware setups are better at finding those quiet patterns that lead to a massive disaster.

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 2, 2026, Pages 1137-1146

This focus on edges also matches how power grids actually work. A failure usually starts or gets worse because a branch gets overloaded, forcing power to go elsewhere and causing a domino effect. If you only look at the nodes, you might see a bus is in trouble but miss the fact that the actual "pipes"; the transmission lines, are about to burst. By putting edge stats into the time-tracking model, the system could see both the local spots and how the stress on the branches was changing. These results also suggest that watching the edges will be even more important as we add more renewable energy and decentralized power. In those setups, the stress on the grid can shift around fast depending on the weather or how people are using power. Aashish et al. (2025) showed that monitoring systems that include resource metrics do a better job at keeping an eye on complex setups [1]. While they were talking about cyber threats and energy efficiency, the takeaway fits here too: you get a much better view of the system when you look at how things relate to each other rather than just checking local gauges. The results here back that up for power grids, showing that branch-level behavior is a key part of seeing when the risk is climbing. Basically, these edge tests prove that predicting a big collapse shouldn't just rely on totals or simple drawings of the grid. To really model the risk, you need a way to see exactly how stress is flowing through the actual transmission lines.

5.3 Operational Meaning Of Degradation: A big point to clear up involves how the word degradation is being used. The experiments didn't actually look at physical parts getting old or wearing out. Since the PowerGraph dataset doesn't have logs on maintenance, how insulation is holding up, or how much life is left in a transformer, it was impossible to track literal rust or mechanical fatigue. Instead, it makes more sense to think of degradation as a growing risk in how the grid is running. In this context, degradation is more about the system getting stressed and becoming more likely to fall apart in a chain reaction. It comes from the idea that a power grid doesn't just fail out of nowhere; it slowly drifts into a shaky state because lines are overloaded, voltages are acting weird, or power is being pushed through paths it wasn't meant for. The graph-learning models were trying to spot these risky patterns as they developed, rather than guessing how many years a generator has left before it breaks down.

The data backs up this way of looking at things. The models were much better at spotting when the grid was in a high-risk state than they were at guessing the exact amount of power that would be lost. The low R^2 scores in the regression tests show that it is really hard to predict the exact size of a blackout just by looking at the early signs. But the early-warning setup did a solid job of flagging when the grid was vulnerable. This distinction matters because a perfectly new piece of equipment can still be part of a massive failure if the rest of the network is under too much pressure. At the same time, an old, "degraded" part might sit there just fine if the system around it is balanced and calm. Looking at the errors the models made helps prove this point even more. Some of the worst failures happened suddenly, without a long lead-up of heavy loading. This suggests that instability often comes from how different parts of the grid interact in the moment, not just from things getting old over time. So, this framework is really a tool for forecasting operational risk, not a standard maintenance plan for fixing old hardware. Keeping this focus makes the most sense given what the data actually shows.

5.4 The Value of Early Warnings: The results show that using graph-based learning on a timeline has some real potential for giving people a heads-up before things go south on a power grid. Even if the models don't catch every single bad event perfectly, being able to flag high-risk situations before a total collapse is still very useful for anyone keeping an eye on the system. In the real world, a warning system doesn't have to be a crystal ball. It just needs to point out when things look sketchy enough to justify a second look or some quick preventive action. The high ROC-AUC scores mean these models are actually quite good at telling the difference between a normal day and a risky one, even when the really bad events are rare. While the accuracy was a bit lower when trying to predict the absolute worst 10% of failures, the models were much stronger at catching moderate levels of stress. This is actually a good thing for day-to-day use. It means the system can flag a problem while it's still manageable, giving operators a chance to fix things before the grid hits a breaking point. This kind of setup could help with several jobs, like watching for general risk, screening for weak spots, or helping operators figure out what to look at first. If a model flags a certain area as high-risk, it could tell someone where to send an inspector or which parts of the network need a deeper stability check. Plus, since these models are built on the actual shape of the network, they might eventually help pinpoint exactly which parts of the grid's structure are causing the most trouble.

This idea of spotting trouble early shows up in other complicated fields too. For instance, some work done by Reza et al. (2025) used machine learning to find early signs of financial trouble by watching digital signals in real-time [20]. Along the same lines, Rahman (2025) created a way to detect small clusters of inflation in the U.S. economy before they turned into a bigger problem [19]. Even though money and electricity are different, the core idea is the same: complicated, interconnected systems usually show signs of stress before they completely break down. That said, it isn't like this is ready to be plugged into a control room tomorrow. The tests were done on simulated data, not live records from a utility company. There is still a lot of work to do on checking how robust these models are and making sure they are calibrated right. But even with those caveats, the research shows that looking at the grid as an evolving graph is a very promising way to build better early-warning tools.

5.5 Things to Keep in Mind: There are a few important points and limitations to keep straight when looking at these results. First, it is worth noting that this study used simulation data from the PowerGraph benchmark rather than actual logs from a utility company. While this dataset is great for controlled experiments because it is structured and easy to replicate, doing well on a benchmark doesn't mean the system is ready to be plugged into a real power grid tomorrow. Real-world grids are messy, they have sensor noise, gaps in data, unpredictable events, and a mix of old and new tech that a simulation might not fully capture. Another point to clarify is how we talk about degradation. Like we've discussed, this isn't about tracking how a specific transformer gets rusty or wears out over time. It's about "operational degradation," which is just a way of saying the grid is getting stressed and more likely to have a major failure. If someone reads the title and thinks this is a tool for predicting the exact lifespan of a generator, they'd be getting the wrong idea.The way the predictions work also has its limits. The labels used to train the models are for the whole graph, meaning they flag when the entire network is at risk of a cascading failure. They aren't meant to point at a single power line and say, "This exact part is going to break at 2:00 PM." While knowing the risk level of the whole system is huge for early warnings, it's a different ballgame than equipment-level diagnostics. We also had to be very careful about "temporal leakage," which is basically a fancy way of saying the model shouldn't be allowed to "cheat" by seeing data from the future during its training phase. To avoid this, the data was split strictly by time, and scaling was only done using the training set. Shuffling the data randomly might make the results look better on paper, but it wouldn't work in the real world where you only know what has happened up until today. That's why we also included things like calibration tests and error analysis, to make sure the performance was actually real and not just a fluke of the metrics. The fact that massive grid failures are rare creates a big challenge called class imbalance. Because these events don't happen often, a model can have a high ROC-AUC score but still be pretty bad at actually picking out the rare, severe cases. That is why the study leans so heavily on PR-AUC and recall. Also, the cross-grid tests showed that what works for one network layout might not work for another. Every grid has its own personality, making it tough to just copy and paste a model from one system to the next.These issues aren't unique to power grids. You see similar hurdles whenever AI is used in complex, connected systems. For example, Dola et al. (2024) found that using machine learning to spot hidden collusion networks in finance is tricky because the relationships between players are so complicated [8]. Islam et al. (2025) pointed out that AI tools used in financial transactions have to deal with a ton of operational uncertainty [12]. Miah et al. (2026) also argued that any AI making high-stakes decisions needs to be transparent and thoroughly vetted [17]. The big takeaway from all these different fields is the same: when you're dealing with a complex, interconnected world, you have to be incredibly careful about how you build, test, and frame your AI models before you trust them to make real decisions.

6. Limitations: It is worth keeping a few things in mind when looking at what was found here. The biggest issue is that the whole thing runs on simulated data from a benchmark, not actual logs from a real utility company. The PowerGraph dataset has good grid shapes and realistic failure scenarios, but a simulation can't truly catch all the messiness of a real grid. Real systems have noisy data, maintenance crews that don't always follow the rules, spotty communication, and weird interactions between software and hardware that a computer model just misses. Because of that, these results show that graph-based risk prediction works in a controlled setting, but it doesn't prove the system is ready to be plugged into a real power plant tomorrow. Then there is the way degradation is handled. This study looks at operational risk—basically, how stressed the grid is and how likely a chain reaction is to start. It isn't actually looking at parts getting old or wearing out. The data doesn't have records on transformer heat, how insulation is holding up, or how many years a generator has left. So, it isn't fair to say this predicts physical aging. It just flags when the grid's current state is dangerous. This is a big distinction because a perfectly new piece of equipment can still be part of a massive crash if the system around it is overloaded.The way "failure" is defined also plays a huge role. The labels for these events depend on where the line is drawn for power loss, or demand-not-served. Predicting trouble gets much harder when the goal is to spot the absolute worst cases, like those in the top 10%, compared to just moderate problems. Even though different cutoffs were tested to see how sensitive the models were, the difficulty of the task still shifts depending on which number is picked. A different cutoff could totally change how accurate or useful the predictions look. Another thing to note is that the predictions happen at the graph level, not the component level. The models can tell if the grid as a whole is in a risky state, but they aren't pinpointing exactly which transformer or power

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 2, 2026, Pages 1137-1146

line is going to blow first. While knowing the overall risk is great for an early-warning system, it doesn't help much if someone is trying to plan which specific part needs to be fixed or replaced.The grid used for most of the tests, the IEEE24, is also pretty small and controlled compared to the massive, sprawling networks that run real cities. Tests were done on larger setups like the IEEE118 and the UK grid to check for robustness, but the core training happened on a smaller scale. Real-world networks are way more complex, with thousands of moving parts and unpredictable shifts in behavior that might not show up in these smaller benchmark environments. Getting a model to work on a new grid after being trained on a different one is still a huge hurdle. The data showed that things like the number of connections and how power is lost change a lot from one system to the next. This means a model that is smart about one grid might need a total overhaul or a lot of retraining before it can be trusted on a different network. It's a common headache in graph learning where even small changes in the layout can throw the whole system off.There is also the problem of understanding why the model makes certain choices. Even if the graph-based setup helps highlight which areas are under stress, those highlights aren't the same as a physical explanation of what went wrong. The math picks up on statistical patterns, not the actual physics of how a failure travels through wires. So, if a model flags a certain node, it means that area looks risky, not necessarily that it is the literal cause of the problem. Finally, the way time is handled depends entirely on how the simulated sequences were set up in the dataset. Efforts were made to keep the timing realistic, but the windows are still limited by the simulation. Real grids have seasonal shifts, changing rules, and long-term habits that might not be in the benchmark. It has been noted in other research, like by Jakir (2025), that when there is too much noise in a system, it gets really hard to find the actual warning signs of a crisis [14]. The same thing probably happens with power failures, where the signs of a massive collapse are buried under a lot of normal, everyday data, making them easy to miss until it is too place.

7. Future Work

There are quite a few ways to build on this work and make the models tougher for real-world grid systems. The biggest next step is definitely testing everything on actual data from utility companies. It would be great to see how this holds up using real SCADA records, PMU streams, logs of past outages, or maintenance histories from live environments. Real data comes with a lot of messiness like sensor noise, missing chunks of information, and all sorts of weird hardware quirks that you just don't get in a clean simulation. Testing there would show if the models can actually handle the chaos of a functioning grid. It also makes sense to move from looking at the whole grid down to specific parts. Right now, the focus is on general risk across the entire graph, but if future datasets include specific details on transformer failures or line breaks, the system could be tweaked to predict exactly which piece of equipment is about to give out. That would make it much more useful for a crew trying to decide which substation needs a technician first.

Bringing some old-school physics into the mix is another interesting path. These models are mostly driven by data, but the laws of physics still apply to electricity. Adding in things like power-flow equations or stability constraints could stop the models from making "impossible" guesses and help them work better when they are moved to a new network layout they haven't seen before. Then there is the question of why the model is making certain choices. This study was mostly about how well the models could predict trouble, not necessarily why they flagged a certain moment as risky. Future work should look at ways to pull back the curtain on these graph networks. Using something like GNNExplainer, which identifies the most important subgraphs behind a decision, could show operators exactly which lines or nodes are causing the most stress [27]. If a person in a control room can see the specific path where a failure might spread, they are much more likely to trust what the AI is telling them.

Transfer learning is another tough nut to crack. The tests showed that a model trained on one grid, like the IEEE118, doesn't always play nice when moved to a different one like the UK grid. Finding ways to create "topology-invariant" versions of these models, basically making them smart enough to understand the "vibe" of a grid regardless of its exact shape, would be a huge win for making this technology easier to ship out to different regions. Reliability also needs a closer look through uncertainty modeling. It isn't enough for a model to just give a risk score; it would be better if it could also say how sure it is about that score. Techniques like Bayesian graph learning or ensemble methods could help separate a "definitely a problem" warning from a "something looks weird but I'm not sure" alert. This is pretty vital when people are making high-stakes decisions about power infrastructure.

Finding the right "trigger point" for an alarm is also something that needs real-world tuning. In this study, thresholds were picked based on the data, but a real utility company has to balance the cost of a missed failure against the annoyance and cost of a false alarm. Every company has a different appetite for risk, so future versions of this work should look at how to pick thresholds that match those specific business and safety goals. Lastly, it would be useful to look further into the future. Most of the work here was about what happens in the very next step, but a grid operator would love to see a forecast that goes out much longer. Seeing how stress builds up over a few hours or days would give teams a lot more breathing room to plan maintenance or move power around before a crisis actually starts. Exploring these longer timelines is really the key to turning this from a reactive tool into a proactive one.

Conclusion

This research looked into how temporal graph neural networks can help model operational degradation risks and predict cascading failures in power grids. It combined graph learning with temporal sequences to try and catch both how grid parts connect and how the system changes over time when it gets close to a major failure. By using the PowerGraph benchmark data, it built sequences from electrical readings at nodes, operational details on branches, and the fixed layout of the network. The goal was to see if actually knowing the network's shape helps predict big failures more accurately. The tests showed that graph-based models did better than standard time-series models at spotting operational risks. This confirms that the way things are connected matters a lot when a system starts falling apart. It turned out that adding details about how electricity flows through individual lines helped more than just looking at the nodes. Specifically, static graph convolutional setups were great at handling the network shape, while models that looked at edges over time were better at pinpointing those rare, massive failures. All of this points toward the same idea: cjatching cascading risks works best when you look at the system's changes, the branch-level status, and the network structure all at once.

A few important lessons came out of the methods used here. Predicting a total grid collapse is inherently hard because those events don't happen often. They also tend to happen because of sudden, messy shifts rather than a slow, predictable build-up of stress. The big gap between the ROC scores and the precision-recall numbers shows that any warning system for a grid has to account for how rare these disasters are. Also, looking at the calibration showed that even when a model is good at ranking risks, trusting the exact probability it spits out is still a hurdle. It helps to remember that degradation here means the risk of the system failing, not the physical parts getting old or rusty. The models were built to find growing stress and weak spots in how the grid operates, not to guess how much life is left in a specific transformer. This way of thinking kept the modeling approach in line with what the benchmark data actually provides. In the end, it seems like temporal graph learning is a solid path forward for predicting risk in modern energy grids. While it still needs testing on real-world utility data and much bigger networks, the study lays out a clear, repeatable process for studying how these failures happen in interconnected systems.

References

[1] Aashish, K. C., Zamil, M. Z. H., Mridul, M. S. I., Akter, L., Sharmin, F., Ayon, E. H., ... & Malla, S. (2025). Towards eco-friendly cybersecurity: Machine learning-based anomaly detection with carbon and energy metrics. International Journal of Applied Mathematics, 38(9s).

[2] Alam, M., Shil, S. K., Sharmin, F., KC, A., Md, A. H., Ali, M., ... & Malla, S. (2026). Hybrid deep learning models for equipment failure prediction in US industrial systems. International Journal of Applied Mathematics, 39(1s).

[3] Bhowmik, P. K., Subha, D. T., Rahim, A., Mohammed, A. A., Begum, M., Chowdhury, R., ... & Shati, M. A. (2025). Self-adaptive machine learning models for financial risk forecasting: Handling non-stationarity in banking and cryptocurrency time series.

[4] Buldyrev, S. V., Parshani, R., Paul, G., Stanley, H. E., & Havlin, S. (2010). Catastrophic cascade of failures in interdependent networks. Nature, 464(7291), 1025–1028. https://doi.org/10.1038/nature08932

MSW MANAGEMENT -Multidisciplinary, Scientific Work and Management Journal
ISSN: 1053-7899
Vol. 36 Issue 2, 2026, Pages 1137-1146

[5] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1724–1734). https://doi.org/10.3115/v1/D14-1179

[6] Debnath, S., Islam, M. R., Abubakkar, M., Islam, M. Z., Mridul, M. S. I., & Biswas, A. K. (2025). AI-driven hybrid deep learning framework for short-term renewable energy forecasting under extreme weather events. In 2025 7th International Conference on Electrical, Control and Instrumentation Engineering (ICECIE) (pp. 362–369). IEEE.

[7] Dobson, I., Carreras, B. A., Lynch, V. E., & Newman, D. E. (2007). Complex systems analysis of series of blackouts: Cascading failure, critical points, and self-organization. Chaos, 17(2), 026103. https://doi.org/10.1063/1.2737822

[8] Dola, A., Begum, S., Antara, U. K., Islam, M. R., Sultana, T., & Zabin, N. (2024). Machine learning models for detecting hidden collusion networks in US corporate finance. Journal of Economics, Finance and Accounting Studies, 6(1), 143–154.

[9] Donon, B., Tabone, M., & Cappart, Q. (2020). Graph neural solvers for power systems. arXiv preprint arXiv:2007.10274.

[10] Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017). On calibration of modern neural networks. In Proceedings of the 34th International Conference on Machine Learning (pp. 1321–1330).

[11] Huang, K., Wang, Y., Zhang, S., Wang, H., & He, D. (2023). PowerGraph: A benchmark dataset for graph learning in power systems. Neural Computing and Applications, 35, 12345–12360.

[12] Islam, M. R., Subha, D. T., Pramanik, M. T., Akter, M., Sweet, M. M. R., Robbani, M. S., ... & Zeeshan, M. A. F. (2025). AI-driven decision support systems for optimizing working capital and customer experience in the US: A transaction-based simulation framework for SMEs.

[13] Islam, M. Z., Sumsuzoha, M., Islam, M. R., Kawsar, M., Mithu, M. F. H., Pant, S., ... & Al Helal, M. A. (2025). Graph neural networks for systemic financial risk forecasting: Modeling cross-market contagion between banking systems and cryptocurrency markets.

[14] Jakir, T. (2025). Signal-to-noise analysis of crisis indicators in global finance using artificial intelligence. International Journal of Applied Mathematics, 38(10s), 1815–1836.

[15] Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations (ICLR).

[16] Lei, Y., Li, N., Guo, L., Li, N., Yan, T., & Lin, J. (2018). Machinery health prognostics: A systematic review from data acquisition to RUL prediction. Mechanical Systems and Signal Processing, 104, 799–834. https://doi.org/10.1016/j.ymssp.2017.11.016

[17] Miah, M. N. I., Uddin, M. J., & Kakumani, M. (2026). Artificial intelligence in sentencing: Evaluating machine learning models for sentencing recommendations in the US. Frontiers in Computer Science and Artificial Intelligence, 5(4), 30–43.

[18] Rahman, M. K., Hossain, M. S., Haque, S. U., Jahed, K. A., Robbani, M. S., Shati, M. A., ... & Pramanik, M. T. (2025). Machine learning models for early warning of financial crises in the US economy using macro-financial indicators.

[19] Rahman, M. S. (2025). Machine learning-enabled early warning system for detecting micro-inflation clusters in the US economy. International Journal of Applied Mathematics, 38(12s), 2743–2769.

[20] Reza, S. A., Rahman, M. K., Rahman, M. D., Sharmin, S., Mithu, M. F. H., Hasnain, K. N., ... & Kabir, R. (2025). Machine learning-enabled early warning system for financial distress using real-time digital signals. arXiv preprint arXiv:2510.22287.

[21] Rossi, E., Chamberlain, B., Frasca, F., Eynard, D., Monti, F., & Bronstein, M. (2020). Temporal graph networks for deep learning on dynamic graphs. arXiv preprint arXiv:2006.10637.

[22] Saito, T., & Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLOS ONE, 10(3), e0118432. https://doi.org/10.1371/journal.pone.0118432

[23] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. IEEE Transactions on Neural Networks, 20(1), 61–80. https://doi.org/10.1109/TNN.2008.2005605

[24] Shawon, R. E. R., et al. (2025). Detecting illicit cross-chain fund movement: Behavioral machine learning models for bridge-based laundering patterns. International Journal of Applied Mathematics, 38(12s).

[25] Sun, C., Hahn, A., & Liu, C.-C. (2018). Cyber security of a power grid: State-of-the-art. International Journal of Electrical Power & Energy Systems, 99, 45–56. https://doi.org/10.1016/j.ijepes.2017.12.020

[26] Wang, Y., Chen, Q., Kang, C., & Xia, Q. (2016). Clustering of electricity consumption behavior dynamics toward big data applications. IEEE Transactions on Smart Grid, 7(5), 2437–2447. https://doi.org/10.1109/TSG.2015.2437513

[27] Ying, Z., Bourgeois, D., You, J., Zitnik, M., & Leskovec, J. (2019). GNNExplainer: Generating explanations for graph neural networks. In Advances in Neural Information Processing Systems, 32.

[28] Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (pp. 3634–3640).