Anomaly Detection and Signal Classification in Financial Data Using Autoencoder-Classifier Neural Networks

Nicholas James Archambault

Practicum Capstone for the Master of Science in Analytics
Georgia Institute of Technology
21 November 2024

In conjunction with Sandia National Laboratories, Albuquerque, New Mexico.

Introduction

Detecting anomalies in signals is an important undertaking when working with time series data across nearly any domain or application. From monitoring the performance of industrial machinery, to evaluating the implications of medical tests like electrocardiograms and electroencephalograms, to projecting the future behavior of financial, biological, or energetic trends, the detection of abnormal patterns is crucial to fully understanding and leveraging signal data. Such detection can allow for preventative action to be a taken or noteworthy phenomena to be investigated: a machine exhibiting signs of wear can be repaired before breaking down; the cause of a persistent pattern of heart arrhythmia can be identified and treated; an unexpected spike in power consumption can be managed.

For the purposes of this project, a 'signal' is broadly defined as a quantity conveying information that changes through time. Beyond the flagging of specific anomalous data points, another important goal when working with signal data involves distinguishing healthy signals from 'degraded' ones – signals whose profiles over an extended period of time feature consistent anomalies, constituting a broader pattern of deviation from typical output. Whereas sporadic anomalies and their causes may be identifiable, the gradual dissociation of a signal's output from its true or expected behavior can be more difficult to detect and preempt. Such insidious degradation may prove highly detrimental to the signal's system, and addressing the root cause of the issue can be complicated by the challenge of characterizing exactly when and how the signal's overall pattern crossed the threshold into degraded status.

This project evaluates the ability of four neural network architectures to detect anomalies in financial data and distinguish between healthy and degraded signals of stock index price and trading activity. Each architecture is comprised of an autoencoder, suitable for detecting patterns in time series data, along with a dense binary classifier for discriminating between healthy and degraded signals. The autoencoder types examined include a simple recurrent neural network (RNN), long short-term memory (LSTM), gated recurrent unit (GRU), and transformer. All code is implemented using Python and its associated packages for numerical processing, data cleaning, and machine learning.

Each architecture is trained and evaluated on daily closing price data of the Dow Jones Industrial Average, as well as a number of engineered statistical and technical financial features incorporating the Dow's historical price and trading volume. Starting with the Dow's daily closing price, tracked from its inception on 26 May 1896, trading activity is partitioned into 413 60-day periods. Each period constitutes a signal, some of which exhibit anomalous, degraded price behavior due to market volatility or other factors. Degraded signals are labeled by determining whether each 60-day period contains any anomalous data points or trends in price activity. Anomalies are identified using two disparate measures of detection: median absolute deviation and rolling Z-score analysis.

For each period, a number of features are engineered to capture long- and short-term trends in price and volume, as well as technical financial measures of volatility, momentum, and the interplay between price and volume. For each of the four architectures, the autoencoder component is trained on only healthy signals to allow it to learn expected patterns. The autoencoder's performance is assessed using a validation set of healthy signals, and the reconstruction errors of this validation step are utilized to determine a threshold for flagging anomalous data points.

Each architecture's trained autoencoder component predicts on a holdout data set consisting of labeled healthy and degraded signal samples. The latent representation of the autoencoder output from this step serves as training data for the binary classifier component, which learns to distinguish between healthy and degraded signals based on the established anomaly threshold. After training on the autoencoder output's latent representation, the classifier outputs a binary prediction of healthy or degraded for each signal. Its performance is evaluated with a final test data set consisting of labeled healthy and degraded signals.

The four architectures perform moderately well, given their simple designs, and are evaluated using accuracy, precision, recall, ROC-AUC score, and F_1 score. The GRU and LSTM achieve the strongest performance among autoencoder types, thanks to their ability to focus on both localized and long term patterns within relatively short sequence lengths. The transformer under-performs expectations, likely due to data volume and sequence lengths that are insufficient to fully leverage its capabilities. The GRU is selected to form the autoencoder component of a final hyperparameter-tuned, high-performing optimized model, which consists of three GRU layers with neuron sizes 512, 256, and 128, along with three dense binary classifier layers. Training and validating on the same data sets as the baseline architectures, the optimized model achieves outstanding performance across all metrics, including accuracy of 0.964, precision of 1, and F_β score of 0.994, using tuned parameter $\beta = 0.1$ and tuned binary decision threshold $\theta = 0.879$. The results suggest this framework – a multi-layer autoencoder to detect sequential patterns, coupled with robust anomaly detection logic and a multi-layer dense binary classifier to distinguish between healthy and degraded signal behavior – can be used to process signals and determine their anomalous behavior across a variety of contexts.

Background

Each of the four architectures evaluated in this project contains two components: an autoencoder and a dense binary classifier. An autoencoder is a type of neural network that learns latent, underlying patterns in data in an unsupervised manner, meaning no ground truth labels are used to assess performance. The general form of an autoencoder consists of an encoder and a decoder. The encoder compresses and maps input data into a lower-dimensional representation, and the decoder attempts to reconstruct the original input data from that lower-dimensional representation. The intuition behind this form is that, in compressing the input and attempting to faithfully reconstruct it, the model learns the most essential patterns and features in the data. The model's effectiveness is evaluated in its reconstruction error, the discrepancy between its reconstructed output and the original input.

Autoencoders are common in anomaly detection applications such as that of this project. When trained on normal, healthy samples, an autoencoder performs well and yields low reconstruction error, having learned during the compression process to filter out extraneous noise and focus only on the samples' general, core patterns. For anomalous samples, however, the autoencoder fails to generalize and struggles to accurately reconstruct the unusual data points and patterns, leading to conspicuously high reconstruction error. Anomalies are typically identified by setting a percentile threshold within the distribution of all reconstruction error values and flagging samples above that threshold as anomalous.

Autoencoders and their unsupervised nature are particularly well-suited to anomaly detection and prediction due to their lack of assumption about anomaly counts or frequencies. Unlike supervised methods, which would require explicit knowledge of the true distribution of anomalies among the normal data samples, autoencoders are capable of handling scenarios where anomalies are rare or unknown.

The neural network models tested in this project – RNN, LSTM, GRU, and transformer – are not inherently autoencoders, but have been adapted to an autoencoder form for more powerful anomaly detection. A goal of this work is to explore these models and the mechanics of modifying them from their original forms to autoencoder versions, and to compare their performances parsing time series sequential data.

RNN

A recurrent neural network (RNN) is the simplest of the four autoencoder architectures evaluated in this project. A standard RNN processes sequential data by maintaining a memory of previous inputs. Unlike traditional feed-forward neural networks, RNNs pass information across successive timesteps through a hidden state that captures temporal dependencies in the data. This hidden state is computed at each timestep based on the current input and the previous hidden state. Outputs can be generated using an activation function at each individual timestep or only at the final timestep. A traditional RNN is useful in sequence-processing tasks such as text generation and sentiment analysis.

As an autoencoder, this structure changes in a few notable ways. The autoencoder version of the RNN still processes input sequences in successive timesteps, but its encoder component generates outputs that are compressed into a single vector of latent representation of the outputs. The decoder component reconstructs the sequence, timestep-by-timestep, from the compressed latent vector. The model is trained, like a typical autoencoder, to minimize reconstruction error rather than a task-specific loss metric typical of a traditional RNN.

RNNs form the foundation of sequential prediction architectures, but they are limited in crucial ways. They are computationally intensive, due to the nature of the single-step sequential prediction mechanism, and they struggle to maintain long-term dependencies due to the effects of exploding and vanishing gradient instability during optimization.

LSTM and GRU

A specialized type of RNN known as long short-term memory (LSTM) overcomes many of these limitations to more easily handle long-term dependencies across the entire input data sequence. This long-term memory is achieved using ‘gates’ at each timestep to decide which information to keep, update, and discard before proceeding to the next timestep. LSTMs introduce a cell state, which carries information throughout the entire sequence and maintains the long-term memory that RNNs lack. LSTMs feature a hidden state at each timestep akin in purpose and behavior to that found in RNNs, but the calculation of the hidden state is conducted with gates that moderate the flow of information, dynamically manage memory and focus, preserve both long- and short-term dependencies across the sequence, and alleviate gradient instability.

Gated recurrent units (GRUs) function and perform similarly to LSTMs, capturing long-term behavior throughout a sequence while managing information flow using gates. GRUs, however, have fewer parameters and are more computationally efficient to train than LSTMs. They feature just two gates as opposed to LSTMs’ three, simplifying network structure by combining the hidden and cell states into a single vector. GRUs are largely interchangeable with LSTMs in terms of use cases and general performance, but are more ideal for applications when computational efficiency is critical, or on smaller data sets where LSTMs may be more likely to overfit to the data. When adapted to autoencoder form, these models assume the same encoder-decoder structure and construct a similar compressed latent representation as found in RNN autoencoders.

Transformer

The state of the art in sequential data processing has become the transformer, a powerful framework that has overtaken all other sequential models in applications from natural language processing to anomaly detection. Unlike RNNs, LSTMs and GRUs, transformers process inputs in parallel rather than timestep-by-timestep, meaning they are highly efficient even for large, complex data sets. Since they process inputs in parallel, transformers have no inherent notion of order. Inputs must be positionally encoded as ‘tokens’ to provide temporal or sequential information.

Once encoded, the inputs are processed using self-attention, an innovative mechanism that ‘attends’ to all parts of the sequence and allows transformers to capture both local and global dependencies better than any recurrent model. The self-attention mechanism can be engineered to have multiple ‘heads’, rendering a more powerful and versatile model. Whereas even the best LSTMs and GRUs struggle with the longest input sequences due to the limitations of their timestep-by-timestep processing, transformers capture dependencies between all embedded tokens in the sequence, regardless of sequence length. In addition to more efficiently and comprehensively capturing dependency patterns, transformers also improve upon traditional recurrent models by scaling processes more seamlessly to large data sets, allowing simplified gradient flow across the entire sequence rather than in a step-by-step manner, and learning more dynamic, complex feature interactions thanks to the ability of self-attention to simultaneously focus on multiple aspects of the input.

After positional encoding has been conducted, transformers are adapted into autoencoder form in a manner similar to recurrent models, with the encoder-decoder structure processing latent representations ideal for anomaly detection applications.

Literature Review

The application of these models – autoencoders, recurrent neural networks and their variants, and transformers – to anomaly detection and signal classification has been the subject of significant research. A substantial portion of that research has been devoted to these models’ usage with stock price data, where reconstruction errors can identify deviations from normal patterns of trading and provide an edge in understanding certain markets. The existing body of work provides fertile ground upon which to execute this project, and highlights exciting applications of these models in stock activity analysis and beyond.

At a broad level, outside a financial context, these models demonstrate strong performance in general anomaly detection thanks to their unsupervised learning framework predicated on signal reconstruction and the minimization of reconstruction error. LSTMs adapted as stacked autoencoders have been shown to excel in anomaly flagging due to their ability to model complex temporal dependencies without predefined context windows [1]. Such stacked models outperform traditional methods like ARIMA in detecting anomalies in both synthetic and real-world datasets. GRUs have been adapted for anomaly detection with structural optimizations [2] that reduce computational complexity and improve performance by focusing on key temporal dependencies while omitting redundant operations like reset gates.

These types of adapted structures are particularly suitable for real-time anomaly detection in high-frequency trading data or fast-moving stock markets, where signals are characterized by short-lived but significant deviations. One of the first stock data applications of autoencoders found that statistical and unsupervised learning approaches were effective for identifying irregularities in big data sets, emphasizing the need for scalable anomaly detection frameworks in high-dimensional financial data and laying the groundwork for integrating future neural network-based approaches [3].

Transformer models leverage self-attention mechanisms to analyze global temporal dependencies without relying on recurrent operations [4]. While originally implemented in natural language processing, transformers have since been applied to financial data analysis because they excel in handling long sequences and feature interactions, which are common in financial data. It has been posited, however, that transformers' heavy reliance on positional encoding to impose temporal order may make them less intuitive than LSTMs and GRUs in purely sequential anomaly detection tasks.

Additionally, combinations of autoencoder anomaly detectors and binary classifiers – the hybrid structure used in this project – have proven effective for distinguishing healthy stock signals from degraded ones. Recent work has found that ensemble neural network methods could significantly enhance predictive power in financial applications, and suggested that combining anomaly detection outputs with binary classifiers not only improved detection accuracy but also provided more interpretable results for investment strategies [5].

Prior research collectively underscores the importance of tailoring anomaly detection methods to the specific characteristics of stock market data. LSTM and GRU autoencoders remain foundational due to their sequential modeling capabilities [2], while transformers offer complementary advantages in handling high-dimensional and multi-variate datasets [4]. Combining these architectures with binary classifiers enhances the detection and classification of healthy versus degraded signals [5], addressing critical needs in financial market analysis. Several works discuss the ancillary challenge of defining appropriate thresholds for reconstruction errors, which play a pivotal role in characterizing anomalies [1, 3]. The integration of machine learning models into financial datasets requires careful attention to generalizability across different markets and time periods to mitigate bias and overfitting.

Methodology

Data

The core data set used in this project was historical daily data for the Dow Jones Industrial Average. The Dow is a stock index of 30 prominent companies listed on United States exchanges. It is price-weighted, meaning stocks with higher prices have greater influence on the index value while those with lower prices contribute less, regardless of market capitalization or total value. The Dow is the second-oldest U.S. market index, having first been calculated on 26 May 1896.

Historical Dow data was pulled from the Polish financial data website *Stooq.com* and downloaded as a CSV file. The initial data file contained, for each day of trading in the Dow's history, its open, high, low, and close prices, as well as total trading volume. The first day of trading captured in the data set was the day of the Dow's inception, 26 May 1896, and the most recent day was 18 October 2024.

After ingestion, the data required no initial cleaning besides ensuring it was complete and free of mistakenly labeled entries. The day of the week was extracted from the date value into its own column. All days with zero listed trading volume were dropped, in part due to the effect such a value would have on future feature calculations and in part to exclude days exhibiting unusual trading and price behavior in the early phases of the Dow's existence.

The remaining data was partitioned into 60-day periods, each of which constituted a signal, and each row was tagged with a number denoting the period to which it belonged. The interval of 60 days was chosen for a number of reasons. This sequence length aligns well with the capabilities of recurrent network architectures, while providing enough data points to capture meaningful patterns and both days- and weeks-long trends in trading data. Shorter periods, such as 30 days, may have introduced noise due to data volume insufficient for identifying meaningful trends over each period. Additionally, since anomalous data points would be labeled individually, the 60-day period provided enough temporal context for a baseline of typical behavior to be identified. Anomalous data would be compared against this baseline, which would minimize false positives by ensuring detected deviations were significant and sustained. This also aligned with the goal of constructing a labeled data set consisting of fewer, genuinely degraded signals, rather than more numerous but less potent or distinguishable anomalous ones. This effort would engender more meaningful classification of healthy versus degraded signals.

All periods were composed of successive 60-day increments of trading activity; no days were rearranged or omitted, including days of notable market activity in financial history. These days were maintained precisely because they typify 'anomalous' behavior in the stock market. They would serve both as valuable corroborations of the methodology used to label anomalies in the data set and as unassailable examples of signals truly degraded.

The last period from October 2024, containing fewer than 60 days, was dropped from the data to ensure uniformity across all period lengths. In its final pre-processed form prior to feature engineering, the data set contained eight columns across 413 60-day periods, ranging from 27 February 1930 to 11 October 2024.

Anomaly Labeling

A crucial step in this project's workflow was creating labels for the data set based on identified anomalies. Periods of Dow trading activity were not natively labeled as healthy or degraded. While consensus wisdom maintains that certain price patterns can be generally identified – recent momentum effects, post-earnings announcement changes, day-of-week effects, and price-to-earnings trends, for example – there exists no authoritative record tracking granular trading anomalies through financial history, and thus no way to label anomalous events in the data set by simple cross-reference. Determining the methods for identifying anomalies and defining degraded signal periods, and validating those methods against both historical truth and statistical intuition, was fruitful ground for exploration and an essential component of facilitating robust model performance.

The intuition behind this process was that the close prices for certain days within a 60-day period may exhibit behavior distinct from those of the rest of the period, betraying themselves as outliers. Degraded signals would be labeled as such if they featured at least one anomaly with their 60-day span, as confirmed by joint identification by two distinct methods: median absolute deviation (MAD) and rolling Z-score analysis. MAD was calculated as

$$MAD = median(|x_i - median(x)|)$$

where x_i is an individual data point from the sequence of a signal's close prices x. Outliers could be identified by comparing deviations to a scaled version of MAD

$$z_{MAD} = \frac{|x_i - median(x)|}{MAD}$$

where a threshold categorized individual points as outliers or not. A scaling factor of 1.4826 was used to make MAD comparable to standard deviation. A threshold of three of these approximated standard deviations was applied to determine outlying points: any point with $1.4826 \cdot |z_{MAD}| > 3$ was deemed an outlier. MAD quantifies the variability of a sequence of data, but is more robust than a typical Z-score to the influence of extreme outliers since it leverages the median rather than the mean. It is also particularly useful for data that does not follow a normal distribution. With a threshold of three scaled 'standard deviations' from the median, MAD identified 403 anomalous data points across 86 signals (20.8%).

To reduce the effect of noisy anomaly classification as much as possible, a second detection metric, rolling Z-score, was also employed. The rolling Z-score was calculated for each data point x_i as

$$Z_i = \frac{x_i - \mu_{15}}{\sigma_{15}}$$

where μ_{15} is the mean price over a 15-day window within the 60-day signal period, and σ_{15} the standard deviation. With threshold of $|z_i| > 3$, rolling Z-score identified 640 anomalous data points across 220 signals (53.2%).

These metrics jointly identified 116 points across 40 signals (9.7%) as anomalous. The low distribution of degraded signals among the data set aligns with real-world expectations about how often actual anomalies occur, and indicates the choice to corroborate anomalous points' validity using multiple metrics was well-founded as a way of reducing false positives. The joint approach mitigated the limitations of each individual technique by balancing a robust global baseline not susceptible to extreme outliers with a more localized rolling score to highlight shorter-term deviations relative to recent activity. Anomalies flagged by both methods were likely to be persistent and significant, representing deviations that stand out among disparate statistical perspectives. For example, a price surge may be flagged as an anomaly by rolling Z-score due to a sharp deviation from recent prices but ignored by MAD if it is within the dataset's overall variability. Conversely, a rare, extreme event might be flagged by MAD but missed by rolling Z-score if it is part of a larger trend.

Visual inspection of the signal periods flagged as degraded indicates the methodology worked as desired. Precipitous, infamous market crashes are reflected in the data as anomalous events, including Black Monday, 19 October 1987; the collapse of Lehman Brothers bank and the onset of the Great Recession, 15 September 2008; and successive days from 9-20 March 2020 marking the arrival of the COVID-19 pandemic in the United States. Notable single-day historic events, as well as longer-term market slides, are also captured: the Dow tumbled on 22 November 1963, the day of President John F. Kennedy's assassination, and consistently fell during the height of the Great Depression in the early 1930s.

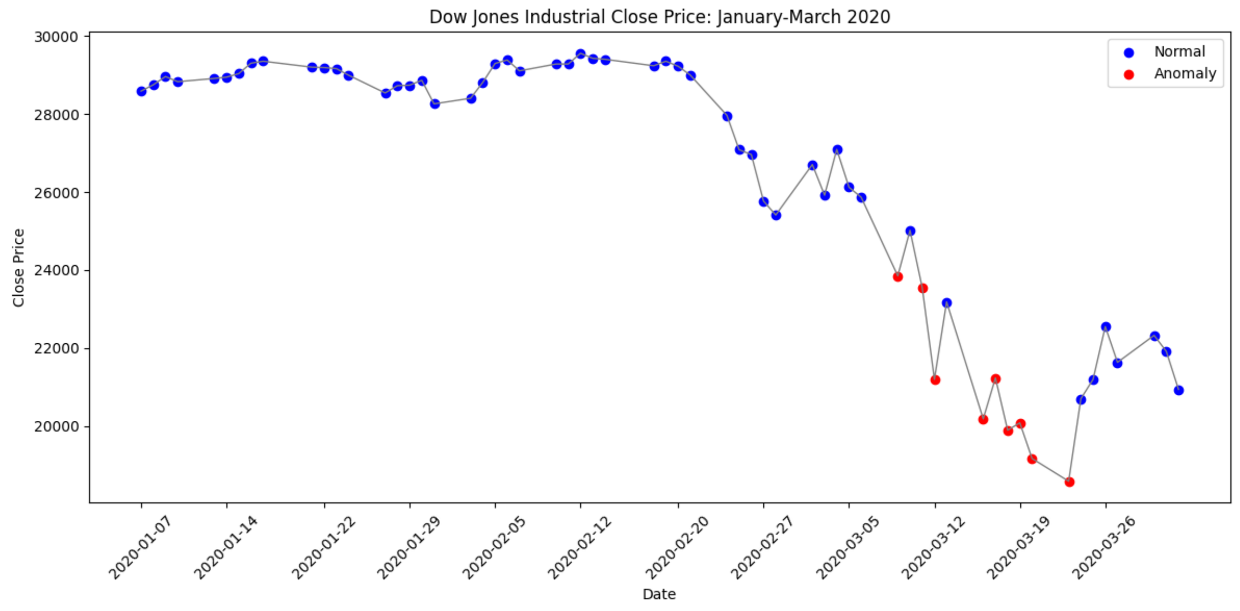


Figure 1: An example of normal and anomalous trading activity as identified by the joint MAD and rolling Z-score anomaly detection framework for the months of January through March, 2020. The COVID-19 pandemic fully arrived in the United States during the first week of March, and precipitous changes in trading immediately followed.

An early idea for conducting anomaly detection and labeling was IsoForest, an algorithm that isolates outliers through recursive partitioning. IsoForest builds a collection of binary trees by randomly selecting a feature in the data and splitting the data at a random point within the feature's range, repeating this process until every point is isolated in its own partition. The number of splits required to isolate each point is recorded. Anomalies, being distinct and sparse in the feature space of the data, intuitively require fewer splits to isolate, while densely concentrated normal points require more. An anomaly score is calculated based on the number of splits, and a threshold is defined based on the distribution of these scores to identify anomalous points.

IsoForest is a robust technique that is computationally efficient and well-suited to high-dimensional data sets. After initial exploration, however, it was dismissed as an effective method of anomaly detection for this project due to its requirement that an estimate of anomaly frequency be known. This is reflected in IsoForest's 'contamination' parameter, which requires an estimated proportion of anomalies to be set prior to running the algorithm. The result is that IsoForest will always identify a proportion of anomalous data points that mirrors the pre-set contamination value. The optimal labeling technique for this analysis would be blind to such an estimation, given that the frequency of anomalies is unknown and it is crucial to organically characterize anomalous behavior within each signal period, rather than in adherence to pre-set expectations across the global breadth of the data. While it initially presented itself as an interesting potential method for anomaly flagging and data labeling, IsoForest proved impractical for the context of the data labeling task at hand. It did, however, inspire careful consideration of the best way to approach labeling, ultimately leading to the joint MAD and rolling Z-score methodology that was utilized.

Feature Engineering

With the data set properly labeled with instances of healthy and degraded signals, a collection of features could be engineered to capture volatility, long- and short-term trends, and specific financial technical indicators for both the Dow's close price and trading volume.

Simple financial features calculated on individual day-rows of the data included the the ratio of close price to volume, the close price-volume interaction, the price range, and the true range. Trend-based statistical measures calculated for each 60-day period included the rolling five-day and 20-day mean and standard deviation of close price and volume, as well as the rolling kurtosis, skew, minimum, and maximum of close price. Additional rolling statistics calculated included the five-day and 20-day exponential weighted means of close price, which place more emphasis on recent timesteps rather than attributing equal importance across timesteps, as with rolling means. The rolling five-day and 20-day means and standard deviations of true price were also calculated, as were returns, volatility, and the correlation between volatility and volume. These features were intended to capture straightforward, interpretable, powerful measures of short- and long-term trends and volatility among both close price and trading volume.

A number of technical financial features were also engineered on each period using custom-built Python functions in order to uncover deeper patterns or capture meaningful behavior outside the scope of familiar statistical and temporal metrics. These features included moving average convergence divergence (MACD), signal line, relative strength index (RSI), on-balance volume (OBV), price-volume trend (PVT), directional movement index (DMI), and momentum. These indicators are often used in concert to anticipate price movements and evaluate the strength of trends, making them valuable tools for comprehensive market analysis.

With all features engineered and added to the data set, extraneous columns that would not be included as predictors were dropped. The numerical features for prediction were standardized in preparation for ingestion into each model, and the categorical day-of-week feature was encoded using one-hot encoding. All null values were backfilled. The final processed data set consisted of 46 predictors, which are listed below alongside their definitions.

- **close**: the daily close price
- **volume**: the daily total of shares traded, reflecting investor interest in the index
- **price_range**: the daily discrepancy between close and open prices:
  $close - open$
- **price_volume_ratio**: the daily ratio of price to shares traded:
  $\frac{close}{volume}$
- **price_volume_interaction**: the daily interaction of price and shares traded:
  $close \cdot volume$
- **true_range**: the daily range of price movement, defined as the maximum of three values. Takes into account gaps and volatility to accurately measure price action:
  $max(high - low, |high - prev.close|, |low - prev.close|)$
- **rolling_mean_20**: the 20-day rolling mean of close price

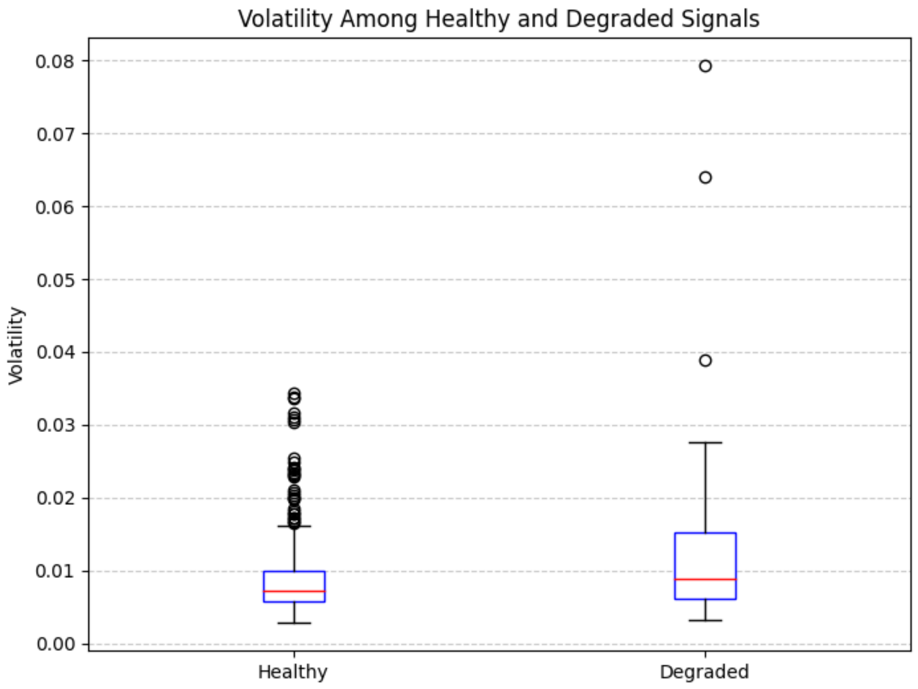


Figure 2: Boxplots of volatility among healthy and degraded signals, demonstrating fundamental differences in the nature of these classes.

- **rolling_mean_5**: the five-day rolling mean of close price
- **rolling_std_20**: the 20-day rolling standard deviation of close price
- **rolling_std_5**: the five-day rolling standard deviation of close price
- **rolling_kurt_20**: the 20-day rolling kurtosis of close price
- **rolling_kurt_5**: the five-day rolling kurtosis of close price
- **rolling_skew_20**: the 20-day rolling skew of close price
- **rolling_skew_5**: the five-day rolling skew of close price
- **rolling_min_20**: the 20-day rolling minimum of close price
- **rolling_min_5**: the five-day rolling minimum of close price
- **rolling_max_20**: the 20-day rolling maximum of close price
- **rolling_max_5**: the five-day rolling maximum of close price
- **volume_mean_20**: the 20-day rolling mean of volume
- **volume_mean_5**: the five-day rolling mean of volume
- **volume_std_20**: the 20-day rolling standard deviation of volume
- **volume_std_5**: the five-day rolling standard deviation of volume
- **ewm_mean_20**: the 20-day exponential weighted mean of close price
- **ewm_mean_5**: the five-day exponential weighted mean of close price
- **ewm_std_20**: the 20-day exponential weighted standard deviation of close price
- **ewm_std_5**: the five-day exponential weighted standard deviation of close price
- **MACD**: moving average convergence divergence, the difference between short (10-day) and long (30-day) term exponential weighted means. Indicates whether the index is in an upward or downward trend:
  $EWM_{10} - EWM_{30}$
- **signal_line**: the nine-day exponential weighted mean of MACD, providing a smoother representation of it
- **average_true_20**: the 20-day rolling mean of true price
- **average_true_5**: the five-day rolling mean of true price
- **RSI**: relative strength index, a momentum oscillator that measures the speed and magnitude of price changes over a defined window (in this case, 15 days), where RS is the ratio of the average of all positive price changes over the window to the absolute value of the average of all negative price changes over the window:
  $100 - \left(\frac{100}{1+RS}\right)$
- **OBV**: on-balance volume, a cumulative indicator that combines price and volume to measure buying and selling pressure. Volume is added to the cumulative total on days when the index closes higher than the previous day, and subtracted on days when it closes lower.
- **PVT**: price-volume trend, a cumulative indicator that accounts for changes to price and volume, similar to OBV but scaled by price percentage change. Indicates bullish or bearish sentiment:
  $\sum \left(\frac{\Delta close}{prev.close} \cdot volume\right)$

- **+DI**: positive direction indicator, a component of Directional Movement Indicators (DMI), used to quantify positive price movements over a defined window (in this case, 15 days) using high and low price:

$$+DM = \begin{cases} \text{High}_d - \text{High}_{d-1}, & \text{if } (\text{High}_d - \text{High}_{d-1}) > (\text{Low}_{d-1} - \text{Low}_d) \text{ and } (\text{High}_d - \text{High}_{d-1}) > 0, \\ 0, & \text{else.} \end{cases}$$

$$+DI = \frac{(+DM)}{TrueRange_{15}} \cdot 100$$

- **-DI**: negative direction indicator, a component of Directional Movement Indicators (DMI), used to quantify negative price movements over a defined window (in this case, 15 days) using high and low price:

$$-DM = \begin{cases} \text{Low}_{d-1} - \text{Low}_d, & \text{if } (\text{Low}_{d-1} - \text{Low}_d) > (\text{High}_d - \text{High}_{d-1}) \text{ and } (\text{Low}_{d-1} - \text{Low}_d) > 0, \\ 0, & \text{else.} \end{cases}$$

$$-DI = \frac{(-DM)}{TrueRange_{15}} \cdot 100$$

- **momentum**: a measure of price change over a given window, in this case 15 days:
$close_d - close_{d-15}$
- **returns**: percent change in close price from the previous day
- **volatility**: 10-day rolling standard deviation of returns
- **volatility_volume_corr**: Pearson correlation between volatility and volume, providing insight into market dynamics
- **Sunday**: binary indicator of whether the trading day is a Sunday
- **Monday**: binary indicator of whether the trading day is a Monday
- **Tuesday**: binary indicator of whether the trading day is a Tuesday
- **Wednesday**: binary indicator of whether the trading day is a Wednesday
- **Thursday**: binary indicator of whether the trading day is a Thursday
- **Friday**: binary indicator of whether the trading day is a Friday
- **Saturday**: binary indicator of whether the trading day is a Saturday

Modeling and Results

With the data adequately prepared, the four baseline architectures could be trained and compared to assess relative performance. The data was first split carefully into the appropriate training and validation sets for training both the autoencoder and classifier stages of each model. The full data set of standardized predictors X was of shape (24780, 46), with 413 60-day signal periods and 46 features. There were 40 signals flagged as degraded. One third of the 373 healthy signals were held out of autoencoder training and validation in order to be included in classifier validation and testing, and to protect against data leakage across the two model components. The remaining 248 healthy signals were split into training and validation sets $X_{healthy\_train}$ and $X_{healthy\_val}$ at a 70/30 split. These data sets were reshaped as tensors of shape (N, T, F) expected by the model frameworks, where N is the number of signal samples, T is the number of timesteps per sample, and F is the number of features. $X_{healthy\_train}$ and $X_{healthy\_val}$ had shapes of (173, 60, 46) and (75, 60, 46), respectively.

The autoencoder portion of each model architecture was trained only on healthy signals in order to learn normal patterns in the data and be able to identify anomalies that deviated from them. The autoencoder's encoder component predicted on $X_{healthy\_val}$, and the low-dimensional latent representation of this prediction was used to train the subsequent classifier stage. An anomaly threshold was identified as the 90th percentile of mean squared reconstruction errors from the encoder's prediction on $X_{healthy\_val}$. The 90th percentile, as opposed to the 95th percentile or another value, was chosen to reflect the true distribution of 9.7% degraded signals in the data.

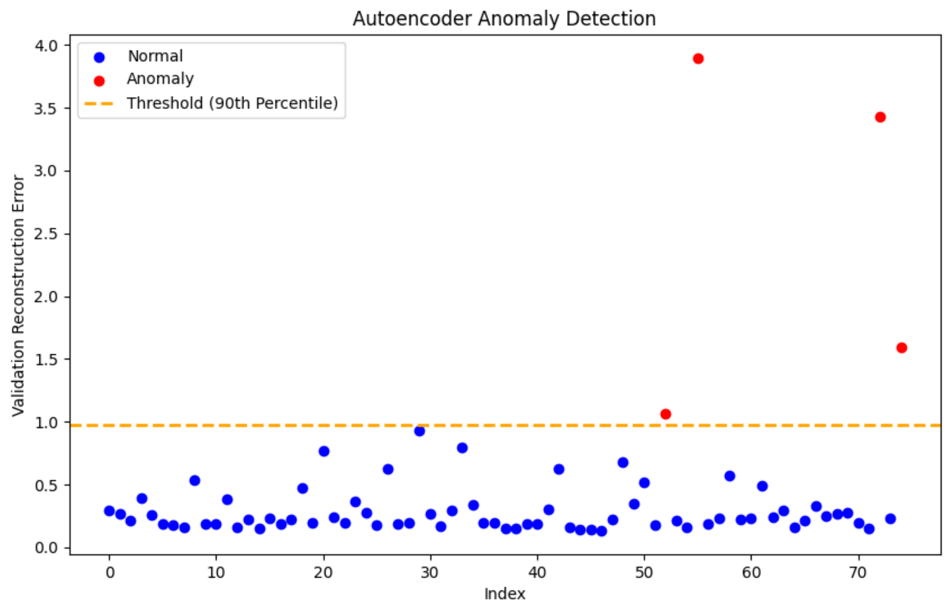


Figure 3: Results from GRU autoencoder training, showing four anomalies detected above the established 90th percentile threshold for reconstruction error on 75 validation signals.

After training on the latent representation output from the encoder of shape (75, 64), the dense binary classifier was validated on X_{val} and its performance was assessed on X_{test}, data sets of shapes (58, 60, 46) and (83, 60, 46), respectively. These data sets were composed of both healthy and degraded signals, none of which were included during autoencoder training or validation. No tuning – of decision threshold θ, F$_{\beta}$ score parameter β, class weighting, or any other model parameter – was conducted for the baseline models during this step.

All models were constructed using similar, basic architectures and identical training parameters. Each model used a single autoencoder layer of 64 neurons with the hyperbolic tangent activation function, trained for 50 epochs using the Adam optimizer with mean squared error loss, a learning rate of 1e-4, and a batch size of 32. Each dense binary classifier featured two layers of 50 and 1 neurons, with hyperbolic tangent activation in the first layer and sigmoid activation in the second. The classifier was trained for 50 epochs using the Adam optimizer with binary cross-entropy loss, a learning rate of 1e-4, a batch size of 32, and five-fold time-ordered cross validation conducted to respect the temporal ordering of the data and prevent information leakage from future data into past predictions. The classification decision threshold was the default value of 0.5.

The transformer autoencoder architecture varied from that used for the recurrent networks, leveraging the same latent dimension of 64 neurons along with eight attention heads, a dropout rate of 0.1, global average 1D pooling, and rectified linear unit (ReLU) activation. The transformer autoencoder was trained using the Adam optimizer for 50 epochs with mean squared error loss, a learning rate of 1e-4, and a batch size of 32. Its dense binary classifier component was identical to that used in the recurrent network architectures.

Model	**Accuracy**	**Precision**	**Recall**	**F_1 Score**	**ROC-AUC Score**
RNN	0.626	0.385	0.125	0.189	0.517
LSTM	0.661	**0.571**	0.1	0.170	**0.744**
GRU	**0.678**	0.556	**0.375**	**0.448**	0.682
Transformer	0.652	0.5	0.075	0.148	0.661

Table 1: Results from validation of baseline architectures. All values averaged across five-fold time-ordered cross validation.

Performance across all models was evaluated using a collection standard metrics for assessing classification results, as shown in Table 1.

Accuracy

Accuracy is the most intuitive metric for evaluating binary classification results: the proportion of correctly predicted instances, true positives plus true negatives, over all instances.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Accuracy values across the four models were relatively close, with GRU leading at 0.678. For an imbalanced data set, however, such as the one used to train the baseline architectures, where healthy signals far outnumber degraded ones and no measures are taken to address this imbalance, accuracy may be misleading, remaining high as the model fattens up on correctly predicted healthy signals while struggling to identify degraded ones. Other metrics must be leveraged alongside accuracy to glean a complete picture of classifier performance.

Precision

Precision measures how many predicted positives are actually correct, and how well the model minimizes false identification of anomalies. In the context of this project, it ensures that the degraded signals identified are truly correct, even if some degraded signals are missed.

$$Precision = \frac{TP}{TP + FP}$$

RNN noticeably lagged behind the other three models in its precision score. The other models were able to identify around half of degraded signals, with LSTM leading at 0.571. This was encouraging performance given the models' simplified architecture, suggesting room for improvement if any hyperparameter tuning or sophisticated design were implemented.

Recall

Recall measures how well the model captures actual positive cases and minimizes false negatives, or true anomalies incorrectly categorized as healthy. In the context of this project, it ensures that as many degraded signals as possible are identified, even if some healthy signals are mistakenly flagged as well.

$$Recall = \frac{TP}{TP + FN}$$

The GRU model exhibited a clear advantage over the others in its recall, with a value of 0.375. The generally low recall scores across all models indicate they performed poorly in even the most aggressive attempts to capture as many degraded signals as possible. This was likely due to the nature of the imbalanced data set, where degraded signals were substantially outnumbered by healthy ones.

F$_\beta$ Score

F$_\beta$ is the weighted harmonic mean of precision and recall, balancing the two metrics using the parameter β.

$$F_\beta = (1+\beta^2) \cdot \frac{Precision \cdot Recall}{(\beta^2 \cdot Precision) + Recall}$$

This score with $\beta = 1$ is known as F_1 score, a traditional classification performance evaluation metric that weights precision and recall equally.

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

For the initial phase of this project when baseline architectures were constructed, F_1 score was used. The GRU achieved the most promising value, more than doubling the F_1 score of the next closest model.

β could be iteratively tuned or balanced differently to reflect different priorities when identifying and acting on degraded signals. A financial firm may use a low β when evaluating degraded signals if it can tolerate a few degraded signals going un-flagged and wants to ensure that flagged anomalies are reliable, minimizing unnecessary legwork to chase down false alarms. If, however, there are severe consequences to missing any degraded signals that arise, the firm may tune β to be more sensitive to recall, avoiding allowing true positives to go unidentified.

ROC-AUC Score

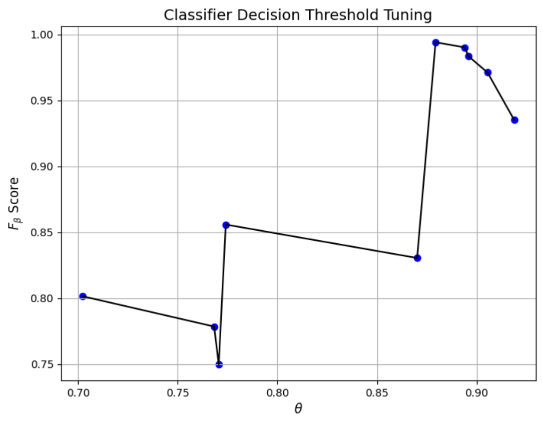
The receiving operating characteristic area under the curve (ROC-AUC) score evaluates a model's ability to distinguish between healthy and degraded signals at various decision thresholds by examining the trade-off between true positive rate (TPR) and false positive rate (FPR). TPR, the proportion of true positives out of all positives predicted by the model, and FPR, the proportion of true negatives out of all negatives predicted by the model, are plotted against each other to show how the model's predictions change under various decision thresholds. The decision threshold θ is the value at which a probability score is converted to a binary prediction. For all baseline architectures, θ was set with the default value of 0.5, meaning a probability score predicted by the classifier greater than 0.5 was converted to a positive, degraded sample, and a score less than 0.5 to a negative, healthy sample. For the final, optimized model, θ was tuned to an optimal value. A ROC-AUC score of 1 indicates perfect discrimination between classes, while a score of 0.5 is little better than random guessing.

ROC-AUC is a useful metric for assessing overall model performance due to its threshold independence: it evaluates model quality at all thresholds, which can be especially important when the optimal threshold is unknown or must be tuned as the result of imbalanced classes, as found in this data set where healthy signals vastly outnumber degraded ones. Though it should still be used in conjunction with other metrics to assess overall model performance amid practical applications, ROC-AUC is valuable for determining the optimal value of θ and gauging general model robustness. The LSTM achieved the highest ROC-AUC of the four baseline architectures, though it seems plausible that the GRU, given its promising performance across all metrics, may have turned out to be the strongest option had its optimal decision threshold been tuned.

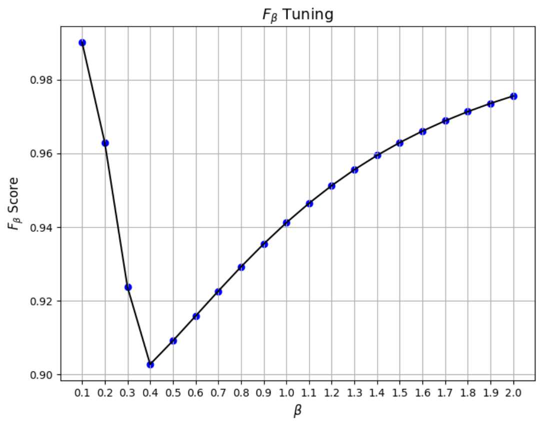
Discussion

Collectively, results across the four baseline architectures were promising, especially given their straightforward structure and training process. The models' performance metrics are not outstanding, likely due to the relatively small size of the training data set: autoencoders thrive on data with large numbers of samples to unearth deep latent patterns. This under-performance, however, was mitigated by the large set of engineered predictors. The GRU and LSTM – featuring similar recurrent, gate-based structures and the ability to detect both long and short patterns in the sequential signal input data – clearly outpaced the other two model types and led in all performance metrics. GRU exhibited marginally higher accuracy and substantially higher recall and F_1 score, while LSTM demonstrated better precision and ROC-AUC score.

As a less capable, more cumbersome version of these two recurrent models, it was unsurprising to find the performance of the simple RNN lagging. It was included in this analysis primarily to serve as a baseline,



(a) F_β score across various values of binary decision threshold θ. Model performance was maximized with $\theta = 0.879$, corresponding to an F_β score of 0.994.



(b) F_β score across various values of β. Model performance was maximized with $\beta = 0.1$, corresponding to an F_β score of 0.994.

Figure 4: Curves showing the tuning of the decision threshold θ and the performance value β for the optimized GRU autoencoder-classifier. These evaluation techniques were critical in tuning the model to adapt to a severely imbalanced data set by finding the optimal θ for positive and negative predicted samples, as well as the optimal parameter β for F_β score evaluation.

and indeed its simplistic, locally-focused structure and capabilities left it trailing in multiple performance metrics.

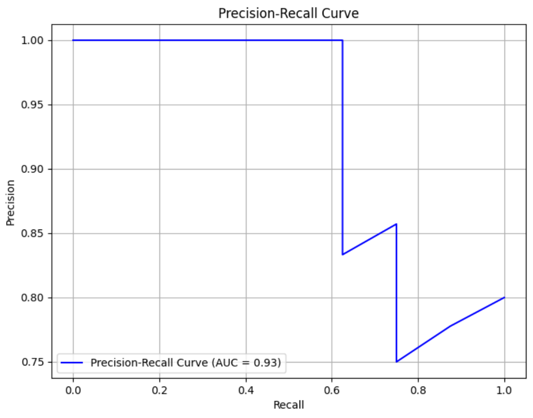
More surprising was the poor performance of the transformer, whose self-attention mechanism to simultaneously parse multiple portions of an input sequence makes it well-suited to sequential analysis tasks. The transformer, however, would benefit even more than the recurrent models from large data volumes, and it likely suffered from the small size of the available data. The transformer likely would have also performed better across longer input sequences. The 60-day duration of the signal periods may have been too short to fully leverage self-attention, which excels at distributing over long sequences rather than focusing on local patterns. LSTM and GRU may simply have proven more appropriate to the context of the problem at hand than the transformer, despite the latter's cutting-edge versatility in myriad sequential applications. LSTMs and GRUs inherently process data timestep-by-timestep, are designed to focus more on local patterns of the type found in shorter sequences, and are better suited than transformers both to capture temporal patterns inherent in stock market data and train on smaller data sets. Transformers, though they leverage positional encoding, may not be as natively effective at extracting short-term positional dependencies critical in time series data, especially when dealing with such short sequence lengths and such small data volume.

It would be difficult to alter this specific project context to better suit transformers' strengths. The Dow has nearly the greatest volume of daily trading data available for any index or stock. Any attempt to increase training data volume would inherently come at the expense of sequence length, which would have to be reduced to compensate.

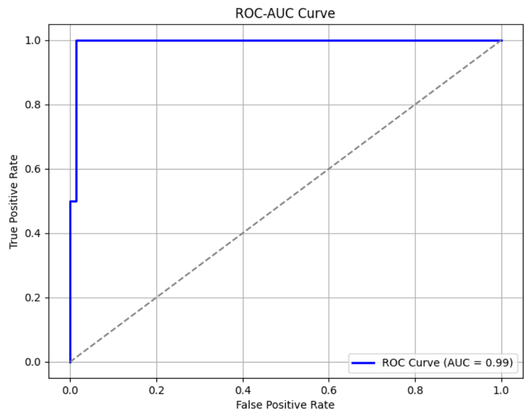
A High-Performance GRU Architecture

Exploration and comparison of the four models' performances in degraded signal detection determined that with uniform, baseline architectures, the GRU and LSTM achieved similarly strong classification outcomes, as evaluated across multiple metrics. The final step in this project was to optimize the best-performing baseline model and determine how well that optimized version could classify degraded signals.

The GRU was chosen for this step over the LSTM, although either model would have been appropriate. GRU thoroughly outpaced LSTM in recall and F_1 score, while LSTM only edged GRU in precision score. Furthermore, the high ROC-AUC score achieved by LSTM should be slightly discounted, given the imbalanced nature of the data set and the need to tune an appropriate θ beyond the default of 0.5.



(a) The precision-recall curve with $AUC = 0.93$, demonstrating high precision and a propensity to become less consistent and introduce more false positives at higher recall levels.



(b) The ROC-AUC curve with $AUC = 0.99$, demonstrating strong discriminatory power between healthy and degraded signals. The gray dashed line represents classification performance no better than random guessing.

Figure 5: Performance curves to evaluate the optimized GRU autoencoder-classifier. The model achieved high true positive rate with low false positive rate, meaning it excelled at identifying degraded signals without incorrectly labeling healthy ones. With the high decision threshold utilized ($\theta = 0.879$), the model made few, highly confident predictions of anomalous behavior, but struggled to maintain such performance at higher levels of recall. This is an expected result, given that β was tuned to 0.1, a value emphasizing high precision at the expense of recall.

Unlike the baseline architectures, the optimized GRU autoencoder-classifier was trained with class weights to address the imbalanced nature of the data set. Class weights assign a higher value to the minority class in order to compensate for its under-representation in the data and compel the model to pay more attention to it during training. Weights are calculated for each class i using the formula

$$W_i = \frac{N}{C \cdot N_i}$$

where N is the total number of data samples, C is the total number of classes, and N_i is the number of data samples in class i. With 373 healthy signals and 40 degraded signals, the class weights were 0.554 for the healthy (negative) class and 5.162 for the degraded (positive) class.

Other parameters tuned for the optimized GRU autoencoder-classifier included the decision threshold θ and performance parameter β. The optimal β value of 0.1 yielded a peak in F_β score, as shown in Figure 4b, and gave far more weight to precision than recall, corresponding to ensured reliability of all identified anomalies and avoidance of a flood of false positive results.

With the optimal β identified and F_β score replacing F_1 score as an evaluation metric, the decision threshold θ was tuned to a value of 0.879 (Figure 4a). Such a high θ prioritizes precision at the expense of recall, meaning the model requires a high level of confidence to classify a signal as degraded.

After repeated hyperparameter tuning and model validation, the final version of the tuned model included a three-layer GRU autoencoder of neuron sizes 512, 256, and 128. The first two autoencoder layers utilized the hyperbolic tangent activation function, while the final layer untilized ReLU. All layers featured dropout percentages of 0.2. The autoencoder was trained for 75 epochs using the Adam optimizer with mean squared error loss, a learning rate of 1e-4, and a batch size of 32. The dense binary classifer was composed of three layers with neuron sizes 100, 25, and 1. The first layer used the ReLU activation function, the second used hyperbolic tangent, and the third used sigmoid. The classifier was trained for 75 epochs using the Adam optimizer with binary cross-entropy loss, a learning rate of 1e-4, a batch size of 10, a tuned θ of 0.879, the identified class weights for imbalanced data, and five-fold time-ordered cross validation. The optimized model was trained, validated, and tested in the same manner as the baseline architectures to protect against

Model	**Accuracy**	**Precision**	**Recall**	**F$_\beta$ Score**	**ROC-AUC Score**
GRU	0.964	1	0.625	0.994	0.992

Table 2: Results from the optimized GRU autoencoder-classifier model. All values averaged across five-fold time-ordered cross validation.

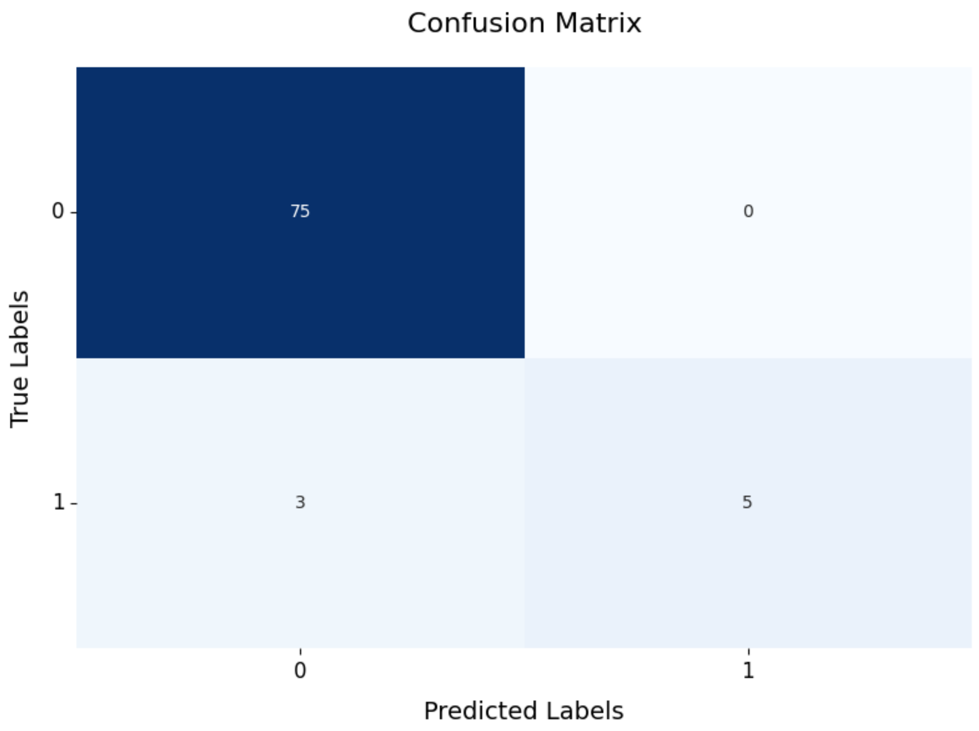


Figure 6: The confusion matrix of results from the optimized GRU autoencoder-classifier. The model correctly classified all healthy signals, and incorrectly predicted three degraded signals as healthy, aligning with the prioritization during model tuning for higher precision at the expense of recall. This meant the model was highly confident in all five correctly classified degraded signals, but also allowed three degraded signals to be incorrectly labeled as healthy.

data leakage and respect temporal ordering.

The model demonstrated strong performance across all metrics, as shown in Table 2 and Figure 5. Of note was the ROC-AUC score of 0.992, indicative of a robust model that, thanks to explicit handling of imbalanced class sizes, was a powerful discriminator across all decision thresholds, even with imbalanced data. Also of note were the perfect precision and relatively low recall, indicating the model correctly flagged all healthy signals, and reflecting the nature of the tuned values θ and β, each of which prioritize precision in lieu of high recall. The confusion matrix of classification results is shown in Figure 6, demonstrating a relatively 'safe' model that is an effective classifier of healthy signals but requires a high level of confidence to flag anomalous ones.

Conclusion

This project explored the ability of four model architectures to process patterns in time series financial data and distinguish healthy signals from degraded ones. Each simplified architecture consisted of an autoencoder component for processing sequential inputs in an unsupervised manner and a dense binary classifier for determining whether or not signals exhibited anomalous behavior signifying degradation.

The four autoencoder types evaluated included a simple recurrent neural network (RNN), long short-term memory (LSTM), gated recurrent unit (GRU), and transformer. Each model was trained under uniform conditions and a basic structure consisting of a single layer, 64-neuron autoencoder coupled with a two-layer dense binary classifier. Data inputs included 46 features derived from the daily closing price and volume of the Dow Jones Industrial Average between 1930 and 2024, partitioned into 413 60-day periods constituting signals. Median absolute deviation and rolling Z-score analysis were leveraged to jointly identify anomalous data points and label the periods containing these anomalies as degraded signals. Approximately 10% of all signal periods were identified as degraded.

The autoencoder component of the hybrid models was trained and validated on exclusively healthy signals, while the dense binary classifier component was trained on a latent representation of the autoencoder output, and validated and tested on a mix of healthy and degraded labeled signals. Model performance was evaluated across common metrics for gauging classification performance, including accuracy, precision, recall, ROC-AUC score, and F_1 score. Of the four architectures, the GRU and LSTM clearly performed the best, as their sequential processing structures were well-suited to extract both localized and long-term patterns from a small data volume comprised of relatively short signal sequences.

An optimized, high-performing model, consisting of a three-layer GRU autoencoder and three-layer dense binary classifier, was constructed and achieved outstanding performance when trained and validated in the same manner as the baseline architectures. Using class weights to handle the imbalanced ratio of healthy to degraded signals, as well a tuned decision threshold $\theta = 0.879$ and performance parameter $\beta = 0.1$, this model achieved accuracy of 0.964, precision of 1, and F_β score of 0.994, suggesting the potential of moderately complex GRU autoencoder-classifier architectures to discriminate between healthy and degraded signals within various domains. In any context where a robust set of features can be engineered and a clear, quantitative distinction can be drawn between healthy and degraded signals, a similar solution could be applied to flag anomalous data and distinguish the degraded signals. GRUs and LSTMs are by nature well-suited to many such contexts, but transformers may prove even more powerful in scenarios where the signal sequence length and data volume are great enough to take full advantage of transformers' architecture and self-attention mechanism.

While the results of this project were encouraging, they were limited and complicated by the nature of the data. A number of specific, insightful features capturing both statistical patterns, trends across different temporal scopes, and various technical financial indicators could be engineered using only the Dow's daily close price and trading volume. This made the Dow data source a rich one. However, the relatively limited volume of available data – especially after partitioning it into 60-day signal periods – likely left all models falling short of their full performance potential. The Dow is the second oldest index in the United States, featuring nearly the richest and largest set of available historical data. But partitioning this data into periods to use as signals quickly diminishes its volume. There is a finite amount of available data, and an inherent trade-off in prioritizing longer sequence length or greater data volume, both of which improve performance in neural networks: either data volume increases at the expense of sequence length, or sequence length is extended but models have access to less trainable data. The framework of this project could have been applied to a wide variety of domains in which individual signals can be collected and a subset of them exhibit anomalous, degraded behavior: industry, manufacturing, energy, medicine, climate, and others. The focus on financial data in this project reflected a personal interest of the author, but its limitations proved suboptimal for evaluating each model type at its purest and fullest potential.

More comprehensive exploration of these model types under the circumstances for which they are ideally suited is a natural extension of this work. A related extension might involve deeper examination of the implications of severe class imbalance, as found in this project's data set, and ways of dealing with it. Such methods might include data level techniques, like oversampling with SMOTE, and algorithmic level techniques, including modifying the loss function to more harshly penalize misclassification of the minority class. This project incorporated careful thought with regard to class balancing during data set splitting and classifier training. It leveraged class weights and avoided relying too heavily on accuracy as an evaluation metric, instead analyzing precision, recall, and other metrics to gauge model performance. Nonetheless, dealing with class imbalance is particularly important during classification problems such as this one, and there are additional methods for managing class imbalance worth further exploration.

A related extension could involve investigating synthetic data generation techniques to address situations where data volume is insufficient, or merely to create additional data with the goal of enabling deeper model learning and stronger performance. A generative adversarial network (GAN) is a machine learning framework designed to generate new, synthetic data that resembles an existing data set. By learning fundamental patterns in real data, the GAN can ingest random noise and apply its learning to generate synthetic data to augment the real data set, providing a greater volume and richer diversity of samples on which to train a separate model – such as a GRU autoencoder-classifier hybrid.

GANs utilize a generator-discriminator structure and 'adversarial' simultaneous training process, whereby the generator creates the synthetic data and attempts to fool the discriminator, and the discriminator attempts to correctly deduce whether data is real or synthetic. Although such models can be tricky to train, they are known to be capable of generating highly realistic data for use in a variety of domains and machine learning problems. Leveraging a GAN or another method for bootstrapping or augmenting a data set to increase its volume could prove a natural extension of this work. It would further expand the already broad array of potential use cases of this project's framework for distinguishing healthy and degraded signals to even more specific contexts, rendering data volume and limitations on model training as concerns of the past.

References

[1] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. Long short-term memory networks for anomaly detection in time series. *European Symposium on Artificial Neural Networks*, pages 89–94, 2015.

[2] Habtamu Fanta, Zhiwen Shao, and Lizhuang Ma. Sitgru: Single-tunnelled gated recurrent unit for abnormality detection. *Information Sciences*, 524:15–32, 2020.

[3] Mohiuddin Ahmed, Nazim Choudhury, and Shahadat Uddin. Anomaly detection on big data in financial markets. *ASONAM*, pages 156–163, 2017.

[4] Ashish et al. Vaswani. Attention is all you need. *Neural Information Processing Systems*, pages 5998–6008, 2017.

[5] Vitor Azevedo and Christopher Hoegner. Enhancing stock market anomalies with machine learning. *Review of Quantitative Finance and Accounting*, 60:195–230, 2023.