Exploration of Artificial Intelligence Techniques for Model Usability with Multivariate Time Sequence Data

A Dissertation

Presented in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

with a

Major in Computer Science

in the

College of Graduate Studies

University of Idaho

by

Mary L. Everett

Approved by:

Major Professor: John Shovic, PhD.

Committee Members: Robert Rinker, PhD.; Julie Beeston, PhD.; Eric Stuen, PhD.

Department Administrator: Terence Soule, PhD.

December 2023

Abstract

Artificial intelligence techniques are prevalent in many industries, but adoption barriers still exist for smaller organizations. Additionally, current deep learning artificial neural network designs seldom account for an evolving data landscape. However, particularly in the realm of precision agriculture, increasing artificial intelligence usage is an important step in expanding production efficiencies and potentially reducing climatic impacts. This dissertation looks at existing popular artificial intelligence techniques and identifies two candidates for further exploration: modularity in deep learning for increased data flexibility and quantitative association rule mining (QARM) in genetic algorithms for more explainable model outputs. Experiments using autoencoders as shared feature spaces for data broken up into different modalities were conducted for the deep learning models to assess the impact on modular networks on overall outputs. Quantitative association rule mining genetic algorithms were also run with and without a novel sequence extension to mine interesting datastream associations and create simple prediction models. Both techniques used two datasets: a weather dataset from the Jornada Basin Research Center and a biomedical dataset (VitalDB) of patient medical data. Ultimately, the modular deep learning models performed as well as the monolithic models with the added advantage of having a shared feature space for more flexibility when it came to adding specific sites for the Jornada Basin data. For the biomedical data, the modular models (which didn't have a concept of being broken up by location) did not provide an advantage over monolithic models. For the Jornada Basin data, the quantitative association rule mining genetic algorithm models performed slightly below the neural network predictors but had explainable output, and the sequence extension was more robust to sub optimal parameters than the non-sequence models.

Acknowledgments

I first want to thank my major professor and advisor, Dr. John Shovic. Dr. Shovic picked me up out of a Real Time Operating Systems class that I almost dropped during my undergraduate career in order to have more time to apply for jobs. I would have never attended graduate school without his encouragement, and he has been available and supportive all the way through. John, thanks for taking a chance on a student that didn't even have a computer science degree.

I want to also thank my graduate committee. Dr. Bob Rinker has helped me become a better and more informed researcher while also helping me navigate administrative challenges. Dr. Julie "BC" Beeston inspires me daily with her love of math and her excellent ability to communicate hard concepts simply. Dr. Eric Stuen helped me understand the stories behind statistics way back during my undergraduate degree and was willing to put up with a graduate computer science student who wanted economic feedback.

Additionally, I would like to thank Dustin and Jaclyn Mommen of Laurel Grove Wine Farm for supporting this research and for supporting the advancement of sustainable agriculture technology worldwide. Their contributions were not limited to funding and their feedback and willingness to partner in creating usable tools for farmers was instrumental in developing these models. Their vision of a better food system has helped guide this project from the get-go.

The research team working on SCARECRO was also a tremendous help in getting to the point of applying artificial intelligence - Nikolai Tiong, Garrett Wells, Jordan Reed, James Lasso, Zach Preston, Lacey Hunt, and Walter Neils. The SCARECRO team is intelligent, hard-working, and fun to be around.

I would also like to thank Dr. Terry Soule, who took the time to personally help me understand how to create and improve the genetic algorithms used in this paper. I was fortunate enough to have two classes with Dr. Soule and have used the lessons learned in many aspects of my research due to his excellent teaching.

I would like to thank Tammy St.John-Tesky for proofreading and helping prepare the document. Tammy also has the daunting task of keeping our office running, and nothing in this dissertation or basically any other research aspect would have been accomplished without her help.

Finally, I am very grateful for the help Dr. Joe Abate and Dr. Dave Barnes in taking the time to help me set up and interpret the biomedical dataset and results. I am assuming that there are very few people who, while trying to warm up with coffee after a 2-mile swim, are willing to be asked questions about surgery for liver patients. I was fortunate to know two people who are, in fact, in that group. Additionally, Joe, I appreciate your ability to search medical literature at the speed of light and explain it to someone whose medical prowess begins and ends with lifeguard training. Thank you both for the gifts of your time and talents!

Dedication

It took many, many people to get me here. This is dedicated first and foremost to my dad and mom, Scott and Beth Everett, and my siblings, Sophie, Peter, Alyssa, Rachel, and Marisa Everett. My family have been my number one supporters throughout everything in my life, including this dissertation. I love and appreciate you all. I also want to thank my extended family, especially my aunt Julie and uncle Brandon, and my grandma and grandpa, for being kind enough to let me stay with them during my graduate and undergraduate degrees, and my aunt Becky, for encouraging me throughout the process. My church family, my community group, and my incredible swim team have been some of the biggest blessings to my life as this dissertation has come together - this is dedicated to you as well. Finally, I want to thank the Biodiesel Education Program (Dr. Dev Shrestha, Dr. Brian He, and John Crockett) for taking a chance and hiring me seven years ago, which started a love for engineering. John – thanks for telling me (for a year!) that I needed to take a computer science class. I think you might have been right.

Contents

List of Figures

List of Acronyms

AE	Autoencoder	14
AI	Artificial Intelligence	1
ANN	Artificial Neural Network	1
CNN	Convolutional Neural Network	11
IoT	Internet of Things	26
GA	Genetic Algorithm	16
GPUs	Graphics Processing Units	1
LSTM	Long Short Term Memory Network	13
PA	Precision Agriculture	2
QARM	Quantitative Association Rule Mining	18
RNN	Recurrent Neural Networks	13
TPUs	Tensor Processing Units	1
MSE	Mean Squared Error	43

1 Introduction

The introduction section of this dissertation aims to give a broad overview to the types of questions explored in the work as well as the motivations behind them. The introduction sections will go over general and specific motivations for the presented research interest.

The last decade has seen a significant uptick of interest in machine learning usage across virtually every industry and sector. Artificial Neural Network (ANN)s in particular have been explored and applied to a variety of problems including image recognition and classification and natural language processing.

However, adoption of Artificial Intelligence (AI) is still slow by smaller organizations, and there exist barriers to creating general models that can adapt with the data being collected in a dynamic rather than static matter. Additionally, neural network approaches, while often powerful, tend to be difficult to analyze and explain results with regard to the specific function actually being fitted to the data.

This dissertation looks at modularity in artificial intelligence systems as well as techniques to make certain artificial intelligence approaches increasingly usable and explainable. The general motivation for this work, explained more fully in the subsequent subsection, involves making artificial intelligence more usable by smaller organizations. The more focused motivation of this work involves increasing the adoption of artificial intelligence techniques in the precision agriculture domain.

1.1 General Motivation

The dominant artificial intelligence methods in use today typically involve supervised learning, which often necessitates pre-collecting large amounts of labeled data for the task of interest. After data collection, design of an effective neural network is more of an art than a science, requiring extensive knowledge of the problem domain or significant trial and error processes, or both. Transfer learning on good existing network designs has mitigated this problem somewhat. Deep artificial neural networks can contain millions of parameters, so it is often useful to train them on specialized hardware like Graphics Processing Units (GPUs) or Tensor Processing Units (TPUs). This equipment can be quite expensive. Artificial neural networks have achieved powerful performance on a variety of tasks, but because of the knowledge required to design and operate them, their usage is largely restricted to researchers as well as larger companies and organizations. Additionally, a deep neural network can be performant but still a "black box" model when it comes to interpreting the inner workings of the model. The sheer number of parameters and difficulty understanding the underlying logic of the model can make it difficult to trust for users.

This dissertation is primarily interested in exploring (1) ways to make artificial intelligence networks easier to implement in the presence of new data, and (2) exploring artificial intelligence alternatives to deep learning networks for more explainable outcomes.

This dissertation takes note of the efficiency of the biological human brain and explores concepts that may be transferable to the artificial realm. The principle biological inspirations in this dissertation utilize modularity, but the biological concepts of evolution are also used in genetic algorithms to efficiently search a solution space.

1.2 Precision Agriculture Motivation

The definition of Precision Agriculture (PA) varies widely, but this proposal will use the term in the most general form, referring to the incorporation of new technology into agriculture practices. Precision agriculture here includes the following systems and practices:

- Site-specific management of ongoing real-time data.
- Drone/UAVs.
- Satellite Imaging.
- Remote Sensing.
- Guidance Systems.
- Automation.
- Robotics.

Research into the current state of the precision agriculture industry has illuminated several drivers for the increased adoption of precision agriculture technology including (1) increased projected demand on the food system (2) climate impacts, and (3) potential for increased profits.

Increased demand on the food system

A 2012 UN Article titled "Feeding the World Sustainably" writes: "According to estimates compiled by the Food and Agriculture Organization (FAO), by 2050 we will need to produce 60 per cent more food to feed a world population of 9.3 billion" [3]. This is an often-quoted statistic in the conversation about the future of agriculture, and concerns over the ability of the world food system to produce adequately have been reflected in governmental policy priorities. The need for greater efficiency in production has been identified as key since the agricultural production land worldwide is expected to decrease in size rather than increase [4].

Precision agriculture has been evaluated as a piece of the solution in realizing these necessary efficiency and yield improvements. The literature surrounding precision agriculture has noted:

- The most common goal surrounding precision agriculture discussions was increasing yield [5].

- Of farmers who had adopted precision agriculture technologies, “72% report a positive yield impact from data-driven seeding rate decisions; 81% from fertilizer decisions and 85% from drainage decisions”[6].

The world food supply problem poses challenges that precision agriculture is positioned to help address.

Climate Impacts

There is little debate that the agriculture industry, particularly in the United States, has come with some negative environmental impacts:

- “Nonpoint-source pollution links agricultural practices with offsite movement of nutrients, pesticides, and sediment”[7].
- “The agricultural sector contributes to the production of 25% of CO_2 , 50% of CH_4 , and 70% of N2O emissions in a global basis summing up to nearly 13.5% of the total global anthropogenic GH emissions”[8].
- There have been concerns over over-application of fertilizer like nitrogen [8] and the contamination of drinking water by farming chemicals, affecting potentially 54 million people in the U.S. [7].

Precision agriculture methods like site-specific treatment and variable rate technologies are expected to better mitigate adverse environmental impacts. A 2021 precision agriculture US Congressional task force report stated that the USDA estimated variable-rate technologies could bring down fuel consumption by 40%, water use by 20-25% and chemical application by 80% [9]. Precision agriculture is also noted to reduce water usage [8], at a 4% level in its current implementation and up to 21% with full adoption in the U.S. [9].

Potential for Increased Profits

Worldwide, there is a strong case for precision agriculture driving increased profitability. A European Parliament Think Tank study noted of a review comprised of 234 studies, [1988-2005] "precision agriculture was found to be profitable in an average of 68% of the cases"[10]. Another study stated that resource saving was an important precision agriculture consideration (in addition to legal compliance, revenue factors, and saving time/labor) for experts in the field of Swiss Outdoor vegetable production. [11]. The United States market reflects the optimism for profitability of the precision agriculture industry, with $670 million raised by startups in 2017 for the sector [5].

Summary of the Current State of the Precision Agriculture Market

The United States is recognized as the current leader in Precision Agriculture research and implementation [12][13], followed by China, Brazil, Spain, Germany, Australia, and Italy. Additionally, "The increased government activities to assist the adoption of modern agriculture technologies and enhanced infrastructure will facilitate the precision agriculture market growth in North America over the forecast period [2022-2026]"[14]. The United States has considerable government support of precision agriculture initiatives and is recommending additional programs for incentive's adoption and innovation in this field [9].

Currently, hardware is the dominant force in the precision agriculture market [15][13], but software and services are expected to experience high growth [15][13]. For the hardware market, guidance systems were identified as having the highest global revenue [16]. When segmenting by application, yield monitoring had the highest revenue share [16][13]. Several market research companies have identified smartphones usage as an important aspect of precision agriculture [17][18].

Summaries of market research reports were evaluated to obtain insight on market potential. Forecast periods ranged from 2024 [19]-2032 [20], with expected CAGR [Compound Annual Growth Rate] of the market ranging from 7.9%[21] to 14.95%[17], with an average of 12.17% among the reports [22][15][17][16][21][20][18][23][14][19][13].

Farmer Precision Agriculture Adoption Barriers

Despite increased interest in research and technology development for precision agriculture, farmer adoption of these techniques has lagged behind industry growth. [10][5]. There is a substantial body of literature addressing what factors impact a farmer's ability to adopt precision agriculture, which include farm size, farmer age, innovativeness, and technical skills, and high cost of precision agriculture implementation [24][22][20][25] [10][11][19][26][16][21].

Of these, this dissertation selects costs and skill gaps as main focus items. Cost particularly is often singled out as the most important adoption factor [10][14], and these two factors have the most straightforward path for improvement.

Large farms [24][11] or higher-value crop producers [10] are usually the producers who most often overcome the high-cost barrier to entry regarding precision agriculture. Precision agriculture systems often require high capital purchases with unclear returns on investment. Some of the costs farmers expecting to use precision agriculture must contend with include [10][11]:

- equipment costs.
- information storage costs.
- hardware and software costs.
- costs related to switching over existing systems.

Internet connectivity has also been singled out in several studies as an implementation cost/barrier [9][11][6], as 25% of U.S. farms are not internet-connected [9], and precision agriculture often relies on connectivity as a prerequisite for use.

The skill/knowledge gap between the developers of precision agriculture systems and their end users (farmers) is another important barrier, because the average U.S. farmer is over 50 years old [9] and IT personnel is unusual on small farms. Technology providers need to keep these factors in mind when designing for their user. Data interoperability between providers and usability of farmer decision support systems is a related issue [27][28][5] regarding this gap as it remains a difficult management task.

Food Distribution System

While there are many improvements to be made in precision agriculture to increase food production, it is important to note that increasing supply and efficiencies are not the only factors involved in world food security. The food distribution system is highly complex and interrelated, and is a major influence on what food is available at what time and what place. While the focus of this dissertation is on developing technology with the goal of assisting food production, the research is also designed to be general to other domains, ideally including supply chains and logistics of transporting and distributing food. Increasing research in the food distribution system has been identified as a currently lacking area when it comes to food security [29].

Precision Agriculture Motivation Summary

Gaps exist between the current state of precision agriculture and the state of precision agriculture adoption, which beg questions about which stakeholders the increased technology interest is focused on [5]. One study states that "By positioning limited adoption rates as a challenge to be overcome by turning to other types of users, or a function of farmers not understanding the benefits of PA, we question if these tools are really for farmers versus the developers and various stakeholders that promote them "[5].

There are certainly contexts where farmers have expressed interest in precision agriculture technology. However, it is essential to evaluate how these tools will help farmers and the food supply, rather than just the bottom line of the technology organizations providing them.

Artificial intelligence has been positioned as a game-changing technology and necessary component of solving the world food crises as well as creating efficiencies and increased profitability to farmers. If artificial intelligence is to be adopted by this industry, however, it needs less barriers to entry in terms of skilled operation in addition to being trustworthy and explainable. Current AI offerings do not broadly meet these criteria. There are needs for models that are easier to set up, maintain, modify, and understand.

1.3 Introduction Summary

This dissertation aims to explore methods of making artificial intelligence more accessible to aid small business and organizations but especially to aid in the adoption of precision agriculture by farmers and grower. The next section will explore existing work in this field and its relation to the broad research motivations.

2 Review of Work

Since this dissertation focuses on exploring more usable methods of integrating artificial intelligence with a non-research-oriented community, several neural network types and techniques have been researched for potential impact. One non-neural network but more explainable technique has also been explored. This section will give an overview of the existing techniques and/or concepts most relevant to this work including Artificial Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks, Autoencoders, Evolutionary Computation for Deep Learning, Evolutionary Computation for Quantitative Association Rules, Biologically Inspired Networks, and Modular Neural Networks. It will also discuss studies that have made strides in related problem domains such as Multimodal Data Fusion Networks and Recurrent Neural Network and Autoencoder combination networks. This section will finish with work that evaluates artificial intelligence and deep neural networks with regard to application areas (Business Cases and Precision Agriculture).

2.1 Artificial Neural Networks

ANNs are perhaps the most popular Artificial Intelligence tool currently in existence. ANNs are not a new technique (the basis for the design was developed in the 1940s [30]), but developments in computational power, memory, and availability of large datasets have caused an explosion of interest in their use over the last decade.

ANNs are loosely analogous to a human brain architecture. Neural networks consist of nodes, which are organized into layers. For a basic, fully connected feedforward ANN, each node in a layer is connected to every node in the layer before and after. An input layer passes data to the network, typically in vectorized form, while the output layer produces information relevant to whatever task the ANN is trained for.

The method of connection between nodes in a network consists of weight and bias values. Each input to the node is multiplied with a specific weight, and then the bias term is added. This value is then processed by an activation function to introduce non-linear learning capabilities to the network. For x, a vector of inputs, w a weight vector, b a bias vector, and θ an activation function, the output of a node n is expressed in relation to the inputs in Equation 1:

$$n = \theta(x * w + b) \tag{1}$$

where $x{*}w$ is an element-wise multiplication followed by summation.

Networks typically incorporate a cost/loss/error function to measure the performance of the network on a given input. The learning capability of ANNs comes with their ability to adjust their weight values to minimize the error of the network. It is a nontrivial task to determine what error should be assigned to an individual weight in the network in order to adjust it. The backpropagation algorithm is a popular choice for determining this and takes the derivative of the overall network error with respect to a particular weight. The weight is then adjusted based on the error during the update step:

$$w_{new} = w_{old} - \eta * \epsilon \tag{2}$$

where w_{new} is the new weight value, w_{old} is the old weight value, η is the learning rate, and ϵ is the error value. For systems that use backpropagation, this error value is the derivative of the total error with respect to that of the given weight.

The weights and bias terms are parameters automatically learned by a neural network. However, there exist many hyperparameters that researchers must set before training which can have a significant impact on ANN performance. The number of nodes in a particular layer and the number of layers in the network are hypothesized to be related to the capacity of the model. The learning rate (which scales the amount of change to a given weight at a training pass) can affect the training speed of the model as well as its ability to converge to a solution. Choices of activation and cost functions have different effects on the model and can contribute to the "vanishing" or "exploding" gradient issues where updates to weights can get unhelpfully large or small during training. Figure 1 illustrates the basic structure of an artificial neural network.

While there exist tools and tricks to help mitigate these problems (the Adam Optimizer for the learning rate, dropout for overfitting, etc.). crafting neural networks that

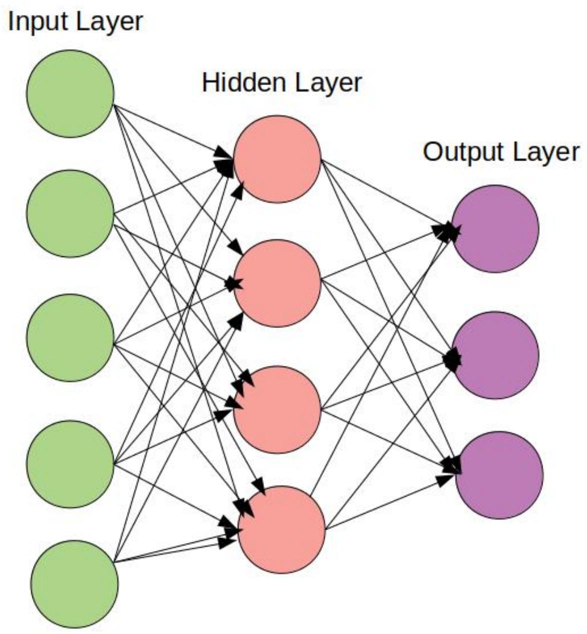


Figure 1: The basic structure of an artificial neural network. There are often multiple hidden layers.

will perform well on a given task is not always straightforward. Network designers often use rules of thumb or transfer learning, where good performance on one network for a task can be used as a baseline for another task.

Additionally, most ANNs require large amounts of representative training data to achieve satisfactory performance on a given task. For example, with regard to the Vertex AI product, the Alphabet company (parent to Google) recommends providing at least 1000 examples per class [31].

For a simple feed-forward fully connected neural network, data is fed to the network as a vector. This prevents a network from learning spatial correlations on more structured data. Another form of ANN, a Convolutional Neural Network (CNN), helps mitigate this problem.

2.2 Convolutional Neural Networks

CNNs are highly performant networks where spatial correlations can be taken advantage of in the data. Images tend to be the most popular inputs for CNNs, as well as data that can be structured into matrix form.

CNNs are composed of convolutional layers, typically followed by fully connected layers in the case of classification tasks. Convolutional layers are composed of filters, which are matrices of weights that are convolved over the input in a sliding window fashion to create feature maps, which are inputs to the next layer. Figure 2 illustrates the basic convolution operation.

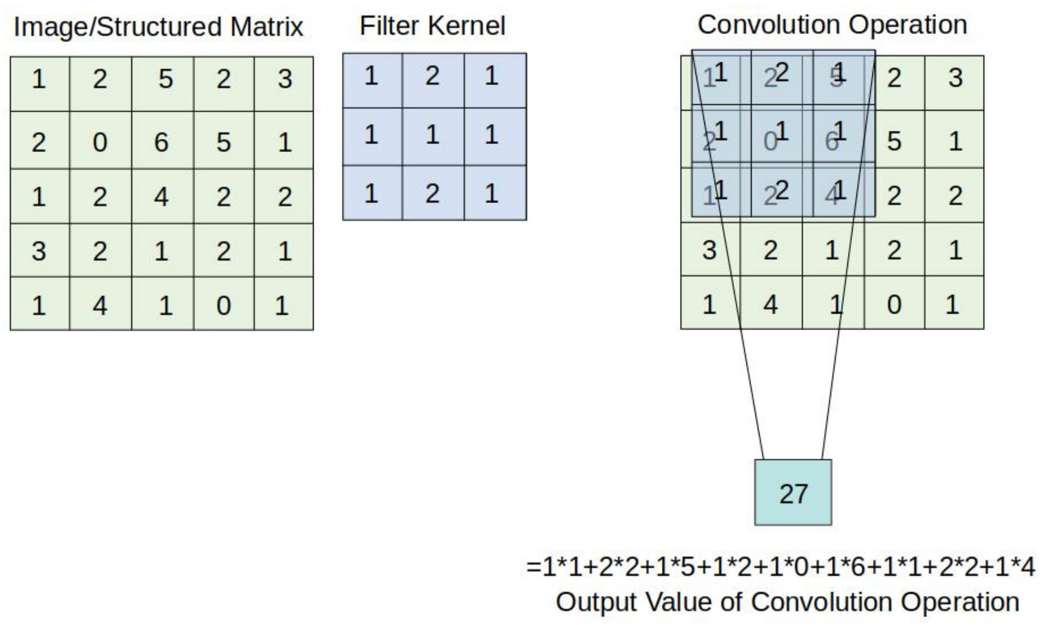


Figure 2: Convolution of a filter kernel with a structured matrix

Different channels of input are multiplied with different kernels within a filter, but channel-wise maps are usually concatenated so there is only one output per filter before the feature map is passed to the next layer. Convolution typically reduces the matrix input size throughout the feedforward process and is flattened into a vector before being passed to the output fully connected layer.

There are many varieties of CNNs which have been applied to a variety of problem spaces. CNNs (and ANNs in general) do not necessarily have to follow connection rules limited to layers just before or after the current layer. Residual Networks [32] introduced skip connections into their network architecture to incorporate information less locally restricted by convolutional operations, to great success.

For tasks that require output images/matrices (such as image segmentation), the fully connected network can be omitted. In this case, convolutional layers are usually followed by upsampling layers to restore input size. Fully Convolutional Neural Networks [33] such as this have the additional handy property of being able to handle multiple size inputs as their feature maps are never flattened to vectors.

CNNs and vanilla ANNs have achieved considerable success when training samples are independent of each other. However, neither network handles inputs well if they are dependent on other inputs. For this, Recurrent Neural Networks (RNN)s are explored.

2.3 Recurrent Neural Networks

RNNs are networks which can handle sequences of data (where inputs are dependent on each other). RNNs divide the inputs into timesteps, then "unroll" a network architecture into inputs, a hidden layer, and outputs. The hidden layer for each timestep is also part of the input to the next timestep. A weight matrix for calculating the hidden layer and outputs is shared across all timesteps. The network can be "unrolled" to as many timesteps as applicable, and therefore, RNNs can theoretically handle as many inputs and outputs as needed. Figure 3 shows a basic RNN construction.

In practice, plain RNNs suffer from the vanishing gradient issue where inputs from long-ago timesteps are forgotten by the network. This limits the length of a sequence an RNN can process. An improvement of the base RNN was found in the Long Short Term Memory Network (LSTM). This network incorporates an input and forget gate along with previous hidden states and is able to retain information from previous timesteps for longer periods of time. LSTMs are practically still limited in input length but are generally able to extend it well beyond the capabilities of a basic RNN.

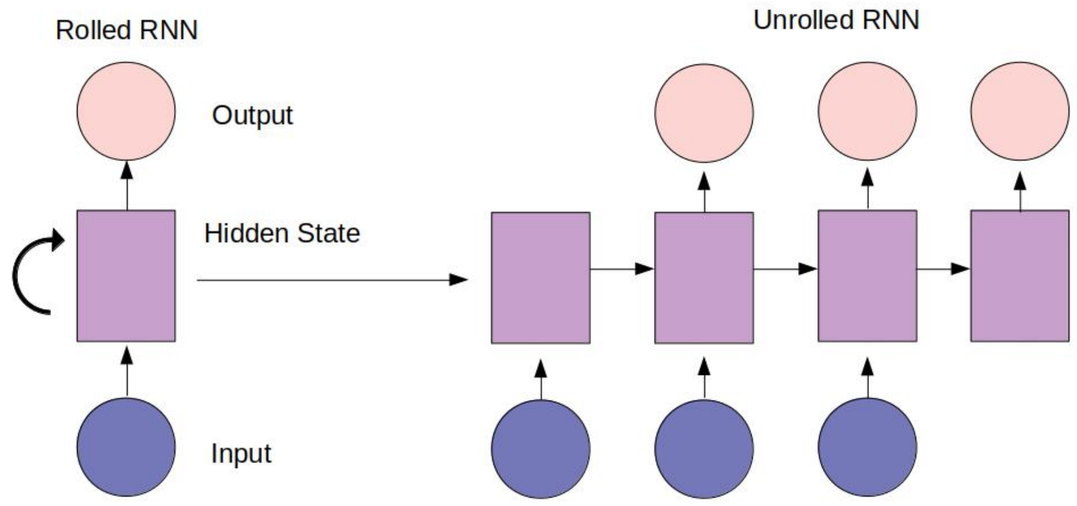


Figure 3: Basic RNN structure, rolled and unrolled.

Another network capable of handling sequential and/or time-varying data includes the Transformer network, which incorporates principles of attention (such as Squeeze and Excitation Networks [34]) to the network with excellent results. At the time of writing, transformer networks are exceptionally resource and data intensive to implement and are not explored as a practical method for expanding general AI usage. However, this dissertation aims to structure the proposed networks so that a transformer network could easily be added later.

2.4 Autoencoders

Outputs at different layers of ANNs are often referred to as "features" or "feature maps". The intuition behind features is that layers of the network are building high-level information from low-level inputs as information travels throughout the network. Therefore designing a neural network architecture is a process of crafting the most useful and representative set of features for the task at hand. However, manually trying to determine feature sets by network design and trial and error experimentation is an arduous process.

Autoencoder (AE)s are a potential solution to this problem. AEs are structures (often implemented by ANNs, but that is not a necessary requirement) where the input and the output are exactly the same: the point of the network is to re-create the input. While this at first sounds pointless, the key functionality of AEs is found in their structure: an undercomplete AE is one in which one or more of the subsequent network layers has less nodes than the input layer. This bottleneck layer (which can also be labeled as the latent space of the hidden representation) is assumed to be a compressed feature representation. Everything before the latent layer is referred to as the encoder, with everything after comprising the decoder. If the decoder can re-create the original input with only the latent layer, the latent layer is assumed to be a good compressed representation of the input. The compression aspect is handy for resource-constrained networks or problems where the input space is huge, but one of the more interesting properties of latent space is that it is assumed to not only be a smaller encoding of the input, but one with good enough features to represent the input. Figure 4 shows a basic AE network structure.

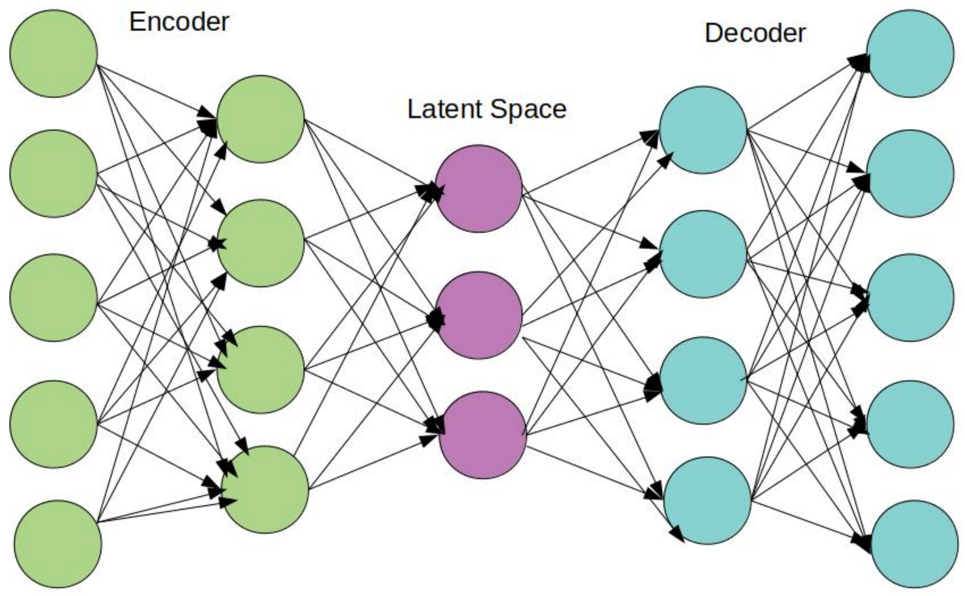


Figure 4: An autoencoder network

AEs can learn without labeled data (unsupervised learning), so they can learn a useful feature set with less data annotation. Other neural networks can be built on top of latent space representations, without needing as computationally intensive pre-processing layers.

Latent space has plenty of prior usage in robotics. In [35], AE latent space was used to train a robot to throw a ball, noting that the reinforcement learning technique used to train the robot converges faster and more stably in latent space. Similarly, [36] applied latent space to visual servoing and found it an effective representation. [37] used reinforcement learning and deep AEs to complete basic visuomotor tasks with a robot. An assistive robot was controlled by AEs mapping contexts to latent actions and decoding latent actions into different actions in [38]. Visual Action Planning was carried out on a robot using latent space to identify transitions between regions and generated valid visual action planes [39], and a Mars Rover was trained to use latent space to opportunistically explore areas [40]. Variational AEs were used to transfer learn between real and synthetic images to allow a robot to take advantage of simulation data in [41]. Latent space was used to create a low dimensional robot state in [42] to control soft robots.

In [43], latent space was used effectively as a search space for routing problems but mentioned that the structure of their latent space was important to performance. A deep AE was also applied to solving traffic congestion problems [44] to predict subsequent time steps of traffic levels. [45] used stacked AEs to diagnose gearbox faults. Patient Health Data and latent space encodings of that same data was used to predict outcomes based on ICU data in [46].

2.5 Evolutionary Computation - Deep Learning

Evolutionary Computation covers a range of topics and techniques, but this section evaluates those relating to design of ANN architectures. Evolutionary computation uses principles of evolutionary algorithms applied to design and optimization problems. One set of techniques includes Genetic Algorithm (GA). Typically GAs follow these basic steps:

1. A population of individual solutions to a problem is created.

2. Candidate solutions are evaluated on the basis of fitness, usually related to how well they solve the problem at hand. The idea of fitness requires input from domain experts as they are best suited to to determining evaluation metrics for a given industry.

3. Individuals are usually selected on the basis of fitness to continue to the next generation. The next population can be direct copies of the previous generation, or they can crossover multiple individuals to create new ones. Mutation is also usually introduced to allow individuals to vary.

4. This process is repeated for a set number of generations, for a specific performance metric, or until a different stopping criteria is reached.

Genetic and evolutionary algorithms have been applied to the problem of automatically designing effective neural networks rather than handcrafting them. One disadvantage of these algorithms is that they are often quite computationally expensive. [47] introduced Progressive Neural Architecture Search, which looked for performant CNNs by searching for increasingly complex structures while learning a surrogate model. The surrogate model helped guide the search of the space in question. This is followed by Efficient Progressive Neural Architecture Search [48] which shares sample architecture weights to reduce the need to train architectures from scratch. The similarly named Efficient Neural Architecture Search [49] searched for "an optimal subgraph within a large computational graph". A policy gradient is used to select an optimal subgraph. [50] uses a HyperNet to generate weights of the model based on its architecture, searching for ANN models in one shot while [51] uses lower-level architecture motifs as network building blocks to search for neural architectures. [52] used Sequential Model-Based Optimization (SMBO) to search for neural architectures specifically for data fusion.

2.6 Genetic Algorithms and Quantitative Association Rules

Evolutionary computation is not limited to neural networks, and the second part of this dissertation looks at evolutionary computation particularly with regard to quantitative association rules.

Quantitative association rules are useful ways of finding correlations between parameters, especially in large datasets or problem domains where a brute-force search would not be attainable. These rules link parameters at certain values with a parameter of interest at a value of interest and quantify the relationship. The relationship quantification looks at the frequency of co-occurrence and statistical importance with a few measures including support, confidence, and lift, which are explained more fully in subsequent sections.

Quantitative Association Rule Mining (QARM) techniques have been explored as a neural network alternative [53] [54] due to their highly explainable output, which is useful regarding trust in artificial intelligence models.

While techniques exist to mine quantitative analysis rules without use of a GA [53] [55], there has been considerable success in using genetic algorithms to find these informative associations [56] [57] [58] [59], including multi-objective approaches [60] [61] [62]. The applicable domains for such searches are vast and cover everything from COVID-19 patient analysis [60] to manufacturing defect analysis [53] and ocean dynamics [63].

There has been some prior work in exploring quantitative association rules for time sequences as well. In [64], this technique was used to forecast next time steps based on previous time steps. [65] actually evolved a single time sequence for associated parameters along with parameter values for its take on the algorithm, which was able to successfully predict ozone based on meteorological data. In [63], time data in a few different forms were utilized alongside other methods to explore ocean dynamics (though time did not appear to be an explicit part of the algorithm). However, the realm of exploration of quantitative association rules with regard to time sequence data appears to still be in the early stages of development.

2.7 Biological Neural Networks

We often look to biology for neural network design processes, in part because often the motivation underlying artificial intelligence research is to automate tasks humans are capable of solving well. Object and person recognition, navigating, planning, and understanding language fall into this category. AI research at the time of this writing is nowhere near the level of generalization power that a human is capable of learning.

While ANNs are roughly based on biological brain construction,the analogy quickly breaks down in implementation. To begin, it is hypothesized that human brains contain billions of neurons and connections, which cannot be physically implemented in hardware at the time of this writing. Human brains are also more efficient than ANNs by orders of magnitude. It takes roughly the same amount of power to light a lightbulb as for humans to solve complex tasks [66], which is much more power-hungry than ANNs.

Many modern ANNs also do not closely align with human brain design principles. Some of these limitations are practical: humans synthesize a tremendous amount of information on a daily basis coming from a variety of sensors - taste, touch, sight, hearing, skin temperature, balance, and pain, to name very few. Not all organizations have the capability to collect a similar breadth of information, and many artificial neural networks are trained on just one kind of information stream (such as images). However, since human data collection is multimodal, it seems reasonable that more sophisticated artificial intelligence designs could benefit from different streams of information.

Another disconnect between the artificial and biological brains includes network structures. Human brains are believed to be highly modular [67], from which we get diagrams of brain functional areas such as the primary visual cortex and frontal cortex. In [68], it is noted that brains are modular at both a lower level, with synapses clustered into dendrites, and at a higher level, with the brain functional areas.

Aside from the functional areas, the human brain is hypothesized to include "connection modules" or groups of neurons whose only purpose is to integrate information across the brain [69]. When your brain is performing a more computationally complex task, brain module activation stays about the same, but connector module activity increases

[69]. Current ANNs tend to be monolithic, and rarely seem to incorporate specialized modules for localized data processing.

Finally, the development and training of ANNs and the human brain differ. It is hypothesized that the main structure of the human brain is more or less set in its design by birth [67] while the connections between neurons are strengthened and weakened over time. This perhaps lends itself to an evolutionary algorithm designing a network structure while conventional methods are used to train this structure (though for many reasons this method is not always feasible for an application). Humans also begin life with neurons hyper-connected, and connections are pruned back over time.

One study sought to incorporate biological design principles into the development of an autonomous robot bike [70]. It had a visual and audio modality with different network structures for each, as well as hybrid state machine for decision making.

2.8 Modular Neural Networks

Modular neural networks incorporate a more biologically likely design. In fact, [71] evaluated neural networks and measured modularity by partitioning neurons into sets where within-set connectivity is high and between-set connectivity is low. They found network topology (pruning particularly) increases clusterability of a network. [68] recommends breaking down the Modular Neural Network Design into 3 stages:

1. Task Decomposition: the overall task is broken up into simpler ones.
2. Training Modules: the simple modules learn their sub-tasks.
3. Multi-Module Decision Making: the local decision making is integrated into global decision making.

In [72] six module operators were evaluated to Neural Network design, including splitting (independent networks), substituting (related to transfer learning), augmenting (adding new networks), inverting (switching the order the networks are run), porting (applying networks to different contexts), and excluding (having the ability to remove networks). In [73] the modularity of biological nervous systems is noted, and modular neural network operations are categorized by domain, topology, formation, and integration. This study discussed techniques for all of these operations. [74] focuses on the design principles of replication and decomposition. Replication allows the reuse of useful modules, where decomposition allows breaking a complex problem into simpler ones. They separately trained modules on subsets of data and were able to better able to recognize noisy images than a monolithic network, though they comment that the monolithic network may be better at statistically neutral problems.

[75] used a decision tree structure where non-terminal nodes were expert neural networks. In [76] a modular neural network was used for visual question answering where individual modules were "independent and composable". [77] also applied Modular Neural Networks to Visual Question Answering . They explored several modular architectures related to sub-tasks and found an intermediate degree of modularity application was most effective for their application. [78] looks at modular neural networks from an interpretability standpoint, as monolithic networks are often difficult to understand. They used Gated Modular Neural Networks and a Mixture of Experts Architecture applied to the Fashion MNIST dataset to try and understand what an individual module learned. This study had some interesting insight on the behavior of jointly training experts and trying to decompose tasks while preventing one expert from starving the rest. [79] and [80] found that modularity in neural networks can assist in mitigating catastrophic forgetting of networks, and [81] introduced a Cooperative Modular Neural Network which used a voting scheme among multiple modules which could affect the responses of other modules. [82] applied problem decomposition to a modular neural network to classify handwritten Hindi alphabets.

2.9 Data Fusion Networks

Data Fusion networks take in more than one type of data and fuse them together at some point during processing. Fusion is often broken up into three potential levels:

1. Early fusion or data level fusion: The data goes through little or no preprocessing before combination with other data in the network. Figure 5 is an example of this.

2. Middle fusion or feature-level fusion: The data is preprocessed before combination and processed together following combination. Figure 6 is an example of this.

3. Decision level or late fusion: Data goes through nearly entirely separate processing streams and is only evaluated together at the output of the network. Figure 7 is an example of this.

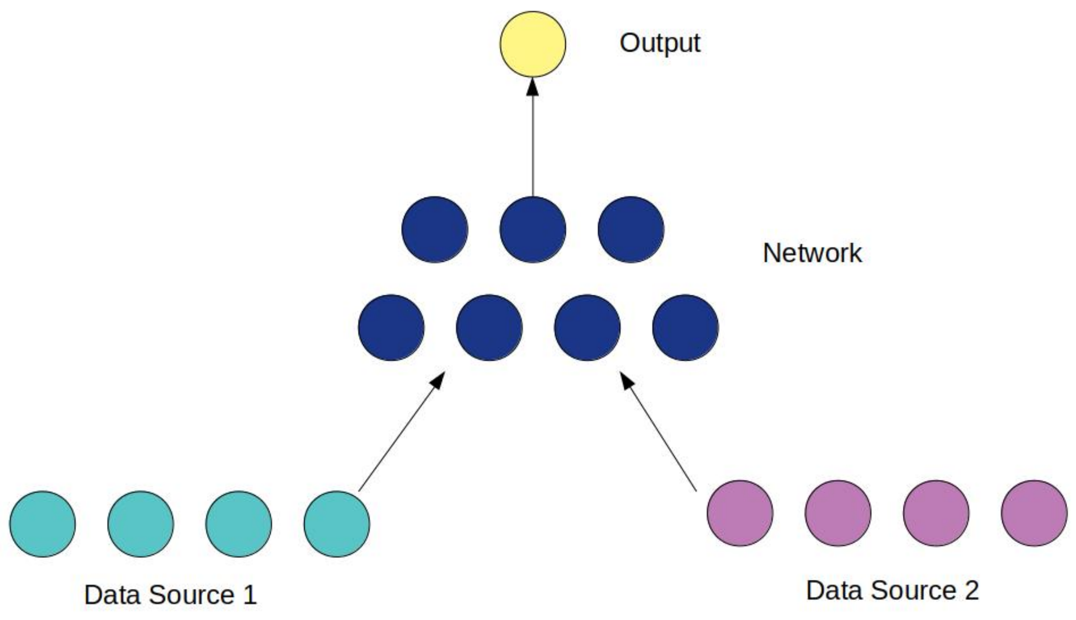


Figure 5: Early or data level fusion network.

[83] looked at data fusion for fault diagnosis of a planetary gearbox. They found they had the highest accuracy with data-level fusions of features but noted feature-level fusion still had good performance. [84] used data fusion along with squeeze and excitation units to add an attention mechanism and found that feature level and decision level fusion performed the best for their model. [85] used decision level fusion and weighted majority voting among modules for good results on classifying uterine EMG signals. [86] used a multimodal model for dynamic hand gesture recognition by augmenting their dataset with optical flow. They regularize their loss to increase performance. [87] looked at early and late fusion for semantic video analysis. They discovered that late fusion in general worked better, though in the cases where early fusion worked better it did so more significantly.

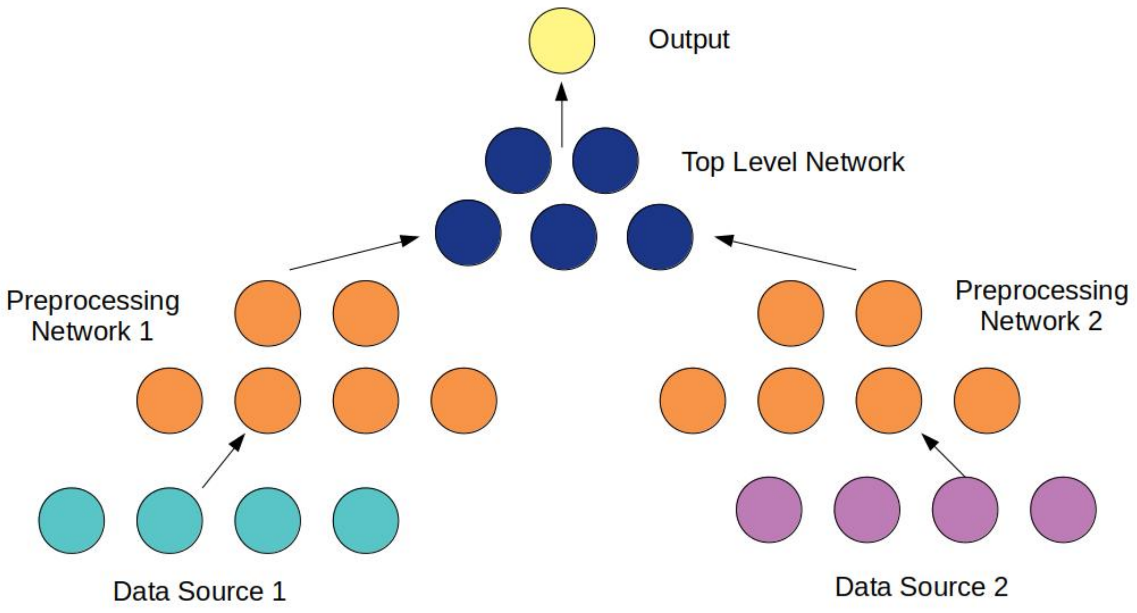


Figure 6: Intermediate or feature level fusion network.

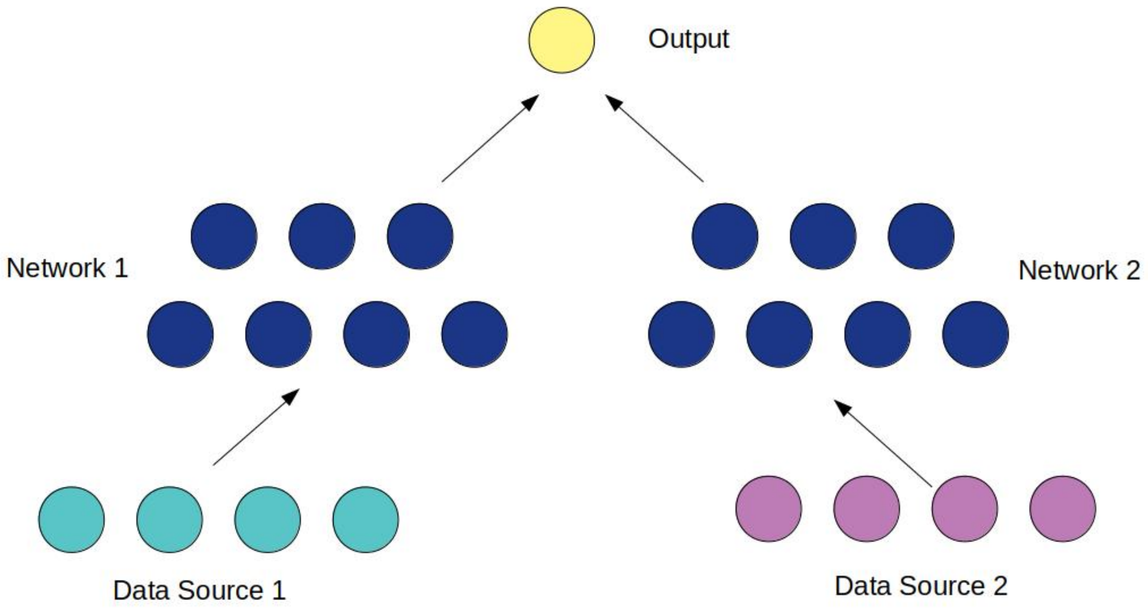


Figure 7: Late or decision level fusion network.

[88] also applied data fusion to video data, in the area of action recognition. [89] fused side face and gait information in videos to identify humans. In [90], multimodal image data was fused and sent through a fully connected recurrent neural network to classify videos.

[91] introduced Gated Multimodal Units - units internal to an ANN designed to find intermediate representations. They used combinations of data from different data modalities, which was tested on the IMDb movie plot and poster dataset. [92] used a "Tensor Fusion Network"with three components: modality embedding subnetworks, tensor fusion layer, and sentiment inference subnetwork. This is an example of feature level fusion, though their tensor fusion layer was more sophisticated compared to simple concatenation of preprocessed inputs. [93] used data fusion along with a CNN incorporating atrous convolution for fault diagnosis. They said of their FAC-CNN network: "Theoretically, the adaptive fusion layer can learn to extract features from unlimited multi-channels to obtain the best one-dimensional representation of the fused data". [94] stacked multiple shared layers between different modalities to create their data fusion network, and used an encoder-decoder architecture that decodes into different modalities. [95] trains their fusion network with some examples having a modality purposefully zeroed out in order to accommodate potential missing values. [96] introduced CentralNet, which combines independent modality networks with a central network and automatically identifies the best fusion levels for combining data.

Attention mechanisms have also been applied to multimodal deep learning, as is the case in [97] and [98]. [97] uses modality-dependent attention weights and leverages pretrained models.

2.10 Recurrent Neural Networks and Autoencoders together

A lot of sensor-streaming/IoT data is time series data. There is a significant body of work concerned with combining the unsupervised learning properties of an AE with the sequential processing abilities of a RNN.

There are two main ways this combination can occur: either as separate modules (typically with AE features feeding the RNN) or with one module integrated with another (typically the AE is composed of recurrent cells). We will begin with the case of an AE composed of recurrent cells.

A RNN inside an AE

An AE composed of LSTM cells was used in [99] for machine prognostics. Anomaly detection for RNNs in AEs appeared to be a popular use case. This was the case for Indoor Air Quality Detection in [100], attack detection for additive manufacturing in [101], audio anomaly detection in [102], network anomaly detection in [103], solar radiation prediction in [104], dialect analysis [105], anomaly detection in ECG readings [106], supply chain manaement [107], remaining useful life prediction [108], and time series forecasting [109].

An AE followed by a RNN

In [110], smartphone sensor data went through a convolutional AE followed by an LSTM network and two fully connected layer for human activity recognition. A stacked AE was used as stage one and an LSTM as stage two for data processing for intrusion detection in [111]. A traffic flow prediction model [112] also opted for the AE followed by LSTM architecture, as did a motion recognition in video sequences model [113], and flight testing [114] (although this model was implemented with Field Programmable Gate Arrays). Attention in the form of squeeze and excitation was added to a similar structure in [115]. Lung ultrasound data was was put through an AE followed by a CNN followed by an LSTM to predict COVID-19 severity in [116], with a similar architecture used for Human Activity Recognition in the DeepSense model [117].

An AE and LSTM were trained in parallel in [118] to forecast coronavirus global spread.

2.11 Business Case Issues with AI

While there is a vast body of research into artificial intelligence and neural networks, very little of it focuses on practical integration of techniques to broader industry and organizational stakeholders. At the moment, training an AI system remains a fairly specialized skill. There are some known obstacles to AE integration for smaller companies and organizations. [119] notes that "Research on AI in IS [Information Systems] is still largely unexplored". The same study notes that much existing research suffers from the lack of a cohesive definition of what artificial intelligence is.

2.12 Internet of Things and Machine Learning Applied to Agriculture

There is an existing body of work applying Internet of Things (IoT) and machine learning technology to agriculture. This section surveys these studies and their outcomes.

On the application side, [120] discussed popular usage areas for deep learning in agriculture including identification of weeds, land cover classification, plant recognition, fruits counting, and crop type classification. In much of the literature covered in this section, diseases and pests classification and monitoring were popular applications as well.

General AI in Agriculture

In [121], VGG and RetinaNet were used to localize and classify banana diseases, and [122] used a bidirectional RNN and LSTM with weather data to predict pests and diseases. [123] used a CNN on different light representations of plant images to classify leaf diseases. [124] used colorimetric space and vegetation indices together with a CNN to detect vine diseases. They found a combination of the YUV and RGB color spaces worked well. [125] also used deep learning for vine disease detection on visible and infrared spectrum images, and [126] and [127] used UAV multispectral images for grapevine disease detection. [128] used a self organizing feature map for grape leaf pixel segmentation to detect grape leaf diseases. Fuzzy decision trees were used in [129] to predict coffee rust based on weather station data. A probabilistic neural network aided [130] in plant classification after using Principal Component Analysis (PCA) to identify 12 features. [131] made use of annotated regions of interest and deconvolution to classify tomato diseases and pests (in 9 categories) in real time.

Wireless Sensor Networks in Agriculture

In [132], Zigbee and Arduino-based Wireless Sensor Networks (WSN) were used for rose greenhouses in Ecuador. They tested linear regression, neural networks, and support vector machines (SVMs) to predict data, with SVM as the top performer. [133] used a mote-based AgriSens WSN for groundnut pest and disease prediction, with Gaussian Naive Bayes classification. [134] used an expert system for variable rate application of fungicide, combining a wireless sensor network and AI backend of GLAIR cognitive architecture. [135] is a WSN in a vineyard located in Spain for monitoring mildew, with the incorporation of a variety of mildew models, while [136] uses a Zigbee-based WSN to monitor diseases conditions, and a Hidden Markov Model to model them. They alerted conditions via SMS messages. [137] used a Naive Bayes Kernel model to assist crop yield in India and [138] also uses WSN with Arduino, Raspberry Pi 3 and Zigbee to monitor pests and diseases. They used KNN, logistic regression, random forest regression, and linear regression to model data. [139] uses a sensor network for tomato blight detection, using real-world images in combination with PlantVillage dataset. [140] used a mote-based WSN to monitor for Apple Scab disease and [141] used a mote system in an Oregon vineyard to predict powdery mildew based on temperature pressure. [142] used a WSN for monitoring the field as well as the wine cellar in a vineyard. [143] also used a WSN for vineyard management. [144] used Zigbee wireless sensors and a mesh network for environmental monitoring.

Data Fusion in Agriculture

In [145], the state of machine learning for detecting crop diseases was surveyed, including data fusion techniques. They noted one challenge of data fusion was that it is “only worthwhile if it increases quality of predictions and relevance of decision based on data combination". [146] used a data fusion network (although they did not use this term) to predict cotton quantity in Greece. Each input was processed by a separate network and features were merged by a concatenate layer, while [147] applied data fusion for rice prediction yield on two kinds of deep features. [148] checked input and feature level fusion for soybean prediction yield which fared better than other methods when more input variables were available. In [149] an extreme learning machine with multi-sensor fusion was applied for phenotyping soybeans. [150] combined remotely-sensed and weather data to forecast powdery mildew, while [151] used a Multi-Context Fusion Network with a CNN backbone for crop disease prediction and outperformed state of the art methods with a 97.5% identification accuracy. Both spatial and temporal context information was incorporated into their model. Plant Classification used global and local leaf features from 2 different CNNs along with early and late fusion in [152], with better success in late fusion.

Precision Agriculture Systems

Some of the current precision agriculture offerings on the market include:

- **FarmBeats:** Microsoft has created (though not yet fully open-sourced) a system, FarmBeats, for precision agriculture, citing the lack of affordable technology as cause for limited adoption of data-driven agriculture by farmers [153]. FarmBeats includes leveraging low-cost hardware (including a specialized radio that can utilized unused TV white spectrum bands [154], a multimodal artificial intelligence system. The FarmBeats Deep Microclimate Prediction Network [155] fuses a forecast with local IoT information at multiple scales using an AE and incorporates a CNN and LSTM for their final output. FarmBeats also encompasses methods of reconstructing satellite images obscured by clouds [156], processing drone data more effectively [157] and decreasing latency for low orbit earth satellites [158].

- **AgStack** [159]: AgStack has been put into motion by the Linux Foundation and is supported by a multitude of partners and members, including OpenTEAM, farmOS, and OurSci. They are developing a comprehensive open-source agriculture infrastructure system. However, they appear to be in the beginning stages of development. Right now, they are working with LF Edge and EdgeX to put IoT tools into AgStack.

- **OpenTEAM** [160]: The Open Technology Ecosystem for Agricultural Management is also working on a suite of precision agriculture tools, though their focus is primarily on soil health. They are supported by several organizations including the USDA. At this moment, they appear to have primarily focused on networking and developing policies among their stakeholders. In their "collabathons", they have discussed technology concepts such as Ag Data Wallet and application programming interface switchboard.

- **OurSci** [161]: Our Sci has several free precision agriculture tools, though also appear to be early in development. They work on both hardware and software. They have a free to use SurveyStack tool for surveys, and are working on projects for soil health, bionutrients, a research farm, and more.

- **vite.net** [162]: This system looked at agriculture decision support systems in regard to usability for farmers and developed a website as an improvement over existing offerings. The site integrates many sources of information particularly for grape growing.

- Other open-source tools include the Trellis Framework, Tania, AgroSense, LiteFarm, AgriXP, and Tamboro.

University of Idaho Precision Agriculture SCARECRO System

In Spring 2022, the University of Idaho began developing a precision agriculture system (more narrowly focused in remote environmental sensing) with Laurel Grove Wine Farm in Winchester, Virginia. The main design principles behind the system include being:

- free (or low-cost),
- open-source,
- farmer-focused, and
- flexible.

The system is not expected or created to be a one-stop end-all precision agriculture solution but is meant to act as a tool for future development and eventual integration with more complete systems (for instance, AgStack). The system aims to speed up research and adoption of precision agriculture remote sensing.

The SCARECRO remote sensing system is composed of the following components:

- sensors (hardware component on-site),
- aggregators (hardware component on-site),
- gateways (hardware component on-site),
- a middle agent and a weredog (software component in the cloud),
- a database (software component in the cloud),
- a dashboard (software component in the cloud), and
- artificial intelligence models (software component, eventually in the cloud).

All components except the artificial intelligence models (the focus of this dissertation) currently have an implementation at Laurel Grove Wine Farm.

One important aspect of the system design is that it is provisionless: no device in the system needs knowledge of other devices in order to function. This was an important choice for our development team as reconfiguring devices in the field can be costly and time consuming. As a result, any sensor can connect to an aggregator or gateway, and any aggregator can connect to any gateway. If a sensor or aggregator is within range of multiple aggregators or gateways, there is potential message duplication, which is handled naively by the middle agent. This has a nice side effect of being more stable to hardware loss if a sensor or aggregator can reach more than one aggregator or gateway. This scheme does require the individual sensors and devices to be identified and geolocated, which was already a necessary requirement for managing site-specific data.

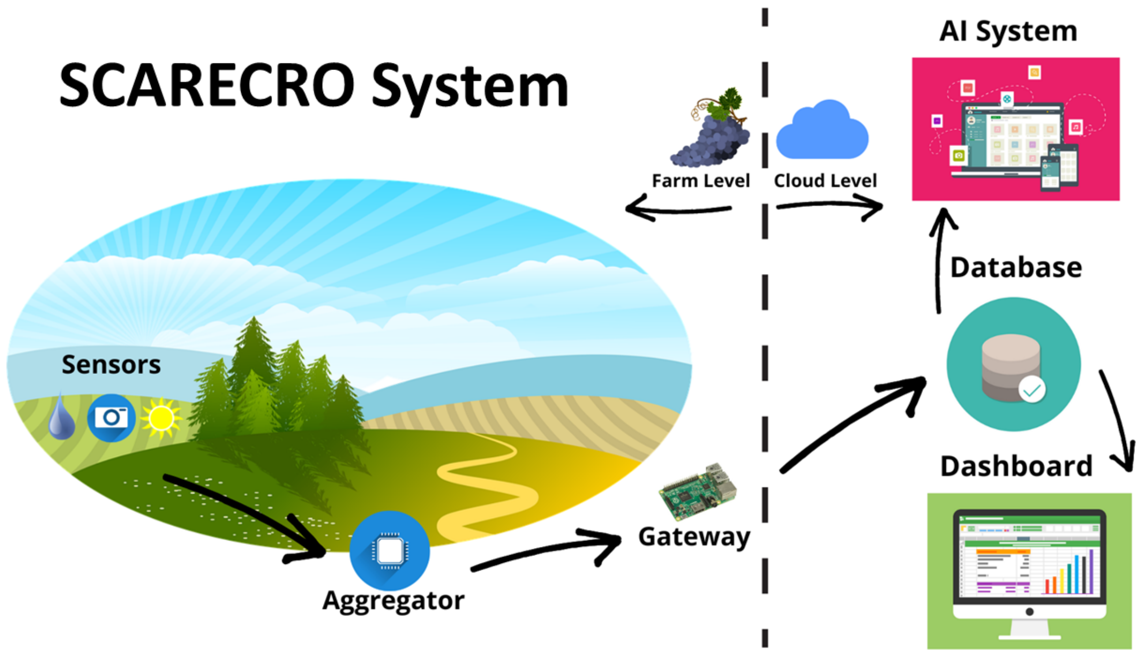


Figure 8: An Overview of the SCARECRO system

Sensors: For the overall system, nearly any type of sensor can be integrated with minimal extra development. The current system uses Minew s1 Bluetooth temperature/humidity sensors, a generic soil pH sensor, soil moisture sensors, and SwitchDoc Labs weather racks, thunderboards, and solar chargers. Sensors can be wired to a gateway or aggregator. Sensors can communicate their information on a variety of protocols to either an aggregator or a gateway.

Aggregators: Since the main communication methods in this system are wireless, there are concerns over wireless communication costs since the current system is using a cellular network from farm to cloud. Aggregators were introduced to extend the range of gateway nodes and provide a middle link between sensors farther away from gateways without increasing connection costs. Aggregators read from sensors and pass information along to gateways via a local WIFI connections out of a hotspot run by the gateway. This allows more sensor information at greater distance without connecting additional devices to the cellular network.

The current implementation of this (called the Datagator) is a esp32-based Firebeetle board with custom PCB design to extend the number of interfaces to different sensors. The resulting board can read Bluetooth and wired sensors and communicate information to the gateway via the local WIFI connection. Any microcontroller that can communicate on a protocol the gateway can understand could act as an aggregator for a system implementation. There are about eleven aggregators in use at Laurel Grove Wine Farm.

Gateways: The gateway reads sensor information from sensors and aggregators, stores a local copy, and sends the information to the middle agent. Gateways can perform data cleaning if necessary for a sensor reading. Gateways keep a local copy of the received data for a configured time in case of a connection loss; if such a loss is detected the local copy can be sent at a later time to the middle agent and database. At the moment, gateways are implemented by Raspberry Pi microcontrollers: our research team likes this device because it is highly extensible and compatible with a variety of interfaces and sensors. The seven implemented gateways at Laurel Grove Wine Farm use a cellular connection to pass sensor data along to the middle agent. Gateway software was written to be highly extensible to new sensor types: a driver that communicates sensor information in the form of a data value dictionary requires minimal code to be integrated into the system. The gateway can accept sensor information on multiple communication protocols: right now, it listens on 433 MHz radio, WIFI, and wired interfaces. The gateway software is around 90% complete and includes features to automatically reboot at connection losses and turn on a cooling fan if overheated.

Middle Agent: The middle agent is single point for all sensor information to flow through before entry into the database. This allows gateways to be decoupled from the database and provides the elimination of duplicate sensor readings. Right now the middle agent implements duplicate elimination naively: it only accepts one message from one sensor instance within a configured time frame. This also allows the farmer to control the data rate into the database (which is not always possible on a sensor level, particularly for inexpensive sensors). Right now, the middle agent and weredog runs on a cloud EC2 instances using AWS, though it could easily be implemented in local hardware.

Database: The database is the home of all remotely sensed data. Right now, this is implemented by Mongodb Atlas, a cloud no-sql database. Its non-relational data schema allows us to be more flexible without sensor readings. With appropriate drivers, other databases (or local rather than cloud instances) could be implemented into the system.

Dashboard: The dashboard is a crucial part of getting collected data to the actual end user (the farmer) so it can be utilized. The dashboard allows real-time monitoring and mapping, as well a historical data visualization and graphing. The dashboard displays data from both remote sensors and other data sources (such as open weather APIs). Right now, the dashboard is implemented as a webpage created in React, library, running on an AWS EC2 instance.

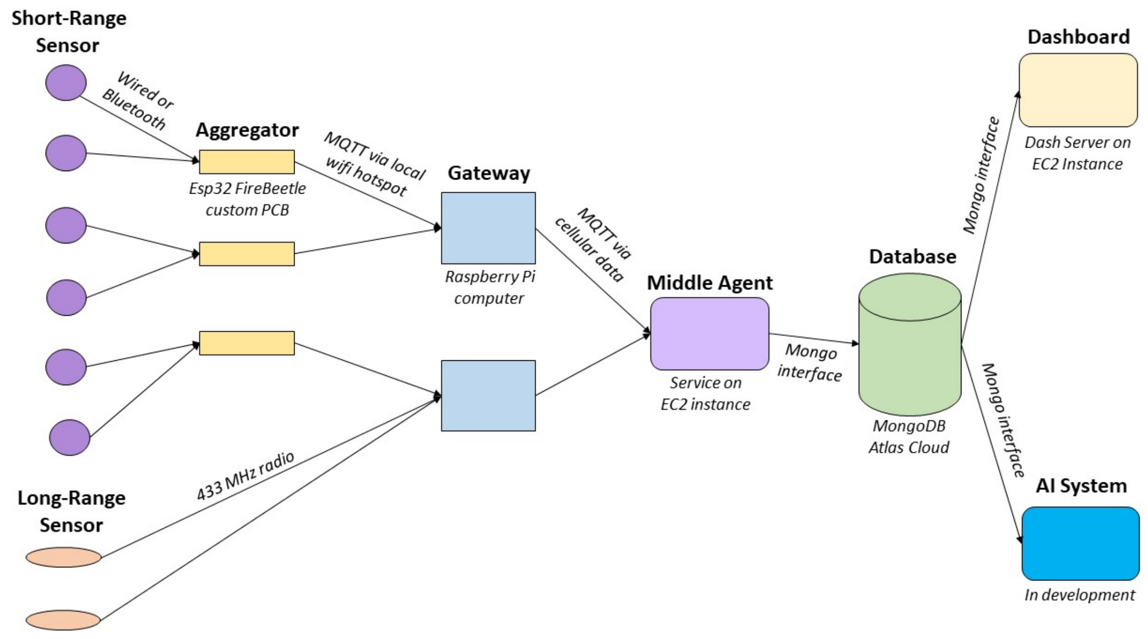


Figure 9: SCARECRO block diagram

Artificial Intelligence Models: The ultimate goal of the SCARECRO system is to collect sufficient data to aid in farmer analysis to make informed decisions about growing techniques and processes. The artificial intelligence models will leverage the data in the cloud database to provide insights to the grower, with an interface eventually developed via the dashboard. This is the only component not currently implemented in the system and is the subject of this dissertation.

Figure 9 illustrates the block diagram of the SCARECRO system.

The SCARECRO system was designed with high-level AI processing in mind. It aims to make high fidelity data collection automatic and simple to implement to aid the ability of AI analysis to assist farmers in making high-level decisions.

2.13 Review of Work Summary

Overall, there have been many studies on improving various data collection and artificial intelligence techniques related to the motivations of this study. There have also been many attempts at improving precision agriculture specifically. This section of the dissertation evaluated relevant existing work and studies to determine what new approaches could be promising to the field. The following section narrows down the focuses of the research in this work and takes a look at the datasets that can be used to evaluate them.

3 Dissertation Focus Areas and Datasets

This section outlines the two research focus areas of the work, and why these areas are applicable to making artificial intelligence more accessible and/or explainable. This section then describes the datasets chosen to evaluate the research focus areas and why they they were selected.

3.1 Focus Areas

There are a vast array of artificial intelligence techniques that could potentially serve precision agriculture adopters and/or smaller business and organizations, only a few of which are outlined in this dissertation. However, this dissertation will focus more narrowly on two areas:

1. Exploring trade-offs of modular flexibility regarding deep learning neural network performance, and
2. Exploring more explainable quantitative association rule techniques compared to deep learning approaches.

The first focus acknowledges the power and performance of deep neural networks and seeks to examine ways they might be structured to better accommodate a dynamic data environment. The second focus examines an alternative to neural networks that yields more explainable results. These approaches were chosen for study due to their potential to aid artificial intelligence adoption and system construction by smaller or less technically savvy organizations or farms.

3.2 Datasets

Two datasets will be used for the experiments: a microclimate dataset, and a biomedical dataset. The datasets are described below, followed by a justification for the choice of these datasets.

Dataset Descriptions

Microclimate Dataset: Jornada Basin [1]: The Jornada Rangeland research program provides free microclimate data on the Jornada basin (located in the Chihuahuan desert), including daily summaries of temperature, humidity, wind speed, solar, radiation, and more across 30 geo-located sites. This dataset is perhaps most related to the current SCARECRO project in terms of remote environmental modeling. The data are provided as csv files and are free to download.

The regression task for this dataset involves predicting weather readings from past weather readings. Predicting the presence or absence of frost will be the prediction task for this dataset.

Biomedical Dataset: VitalDB [2]: VitalDB is a dataset of recorded vital signs from surgery patients in Seoul National University Hospital in South Korea. The patient data includes recordings from monitors during the surgery, as well as patient clinical data and lab results from before, during, and after surgery. For these experiments, a subset of this vast database will be used: 38 patients undergoing liver transplant surgery. The regression tasks for this dataset will involve predicting vital signs ahead of time, while the prediction tasks will include whether or not the operation is an emergency operation, as well as glucose and discharge risk factors for these types of patients.

Both the Jornada Basin Dataset and the VitalDB dataset had more data available than was usable in the scope of this project. Part of the experimental setup was evaluating which types of data, in what formats, would be used in the resulting neural networks. The datasets also had differences in the data formats, which necessitated slightly different slates of models for each. The data formatting, types of models, and experimental setups are described below.

Dataset Choice Justification

Both of the chosen datasets collect time data from a variety of features over set periods. However, these datasets are otherwise very dissimilar, with different domains, manners of collections, and temporal resolution. The differences in the chosen datasets are purposeful: in order to evaluate the models, it was important to note whether or not performance was generalizable across domains or different time scales. The Jornada Basin dataset has an obvious connection to precision agriculture in that it is composed of mostly weather features, and models scrutinized subsequent time steps for the presence or absence of frost. The VitalDB is a medical dataset, very different from precision agriculture. Aside from being a new application of the models, the VitalDB dataset has a very fine temporal resolution (on the level of seconds rather than days), which allows investigation of model performance on more than one time scale. While these two datasets do not form a comprehensive set of domains for experimentation, their differences are sufficient to obtain an idea about the generalizability of the proposed models.

3.3 Focus and Datasets Summary

This section described the two specific focus areas of the dissertation and gave an overview of the type and purpose of the datasets chosen to evaluate these areas. The next section outlines proposed experiments for the datasets and the methodology of setting them up.

4 Methodology and Proposed Experiments

The focus of this section of the dissertation is to develop intuition on artificial intelligence deep neural network design for dynamic environments that can be implemented by small or medium size organizations with multiple streams of data to process. This section is split up by focus area, with deep learning modular networks explored first and genetic algorithms second. Each focus first covers the principle research questions guiding the experiments and neural network or algorithm design. It then covers potential datasets that will be used for experimental validation, the neural network designs incorporated into the experiments, and the steps for running each experiment. Finally, the metrics for evaluating network design performance in the experiment are discussed.

4.1 Deep Learning Focus

This subsection is covers the methodology and design related to making modular neural networks.

Research Questions

Four research questions relating to neural network design and data processing are presented below:

1. **What is the best method of incorporating new data into an existing neural network design?** Neural networks are often restricted to a certain input size, and new information cannot always be fitted to an existing neural network structure. If a new data stream should be incorporated, often the network must be retrained. This can be difficult for a system to handle if the new data stream was not available at the same time as the other data values. There are several ways to investigate this problem, including:

    (a) Re-training a complete neural network system for new input;

    (b) Creating a neural network structure that can handle different types of inputs added or subtracted; or

   (c) Retraining a top-level model on an existing feature set, rather than re-training an end-to-end model.

2. **How modular can we make a multivariate system?** There have been notable successes in training modular neural networks and training networks with multivariate data. Modularity in systems is very useful as it lends itself to a plug-and-play design without needing to bother with interrelated dependencies if a data stream is added, subtracted, or changed. However, as modular as the biological human brain is hypothesized to be, it is an incredibly interconnected organ. Ideally, a system can leverage independently preprocessed data features, but at some point, it is likely that higher capabilities come with fusion before a decision.

3. **Can we delegate multivariate (multiple types of input data) decision tasks to exist on top of an existing feature set?** High-level decision training will ultimately necessitate some type of training label or feedback. However, it would be beneficial to obtain a powerful feature set in an unsupervised manner, allowing for smaller, more tailored decision networks reliant on this feature set.

These research questions will be evaluated with 2 datasets on several different neural network designs.

Network Designs

To investigate the research questions, 4 network designs and 2 training modifications are proposed. These network designs include using base models and current state of the art approaches as comparisons to modified approaches incorporating various degrees of preprocessing for eventual data fusion. The training modifications include adding data streams and incorporating spatial locality principles. These designs incorporate functionality discussed previously in the literature review, including using an AE as a preprocessing technique with another network (likely an LSTM). The difference of this approach has to do with adding data streams as well as investigating several levels of data fusion with the AE preprocessor.

Network Design 1: Base Models The first network designs for all 3 datasets are baseline comparative models. A base approach for single and multivariate data streams will be trained to provide a comparison for subsequent models. These models will be end to end, with the multivariate data as input and the network directly giving the output for a specific task. Figure 10 illustrates this design.

This design is meant to model a handcrafted (per task) network design for a particular stream of data and use case. It is hypothesized that tweaking a base network of this type can produce good results but may be time-consuming to fine-tune. It is also hypothesized that with more data incorporation, this network design will be slow to retrain.

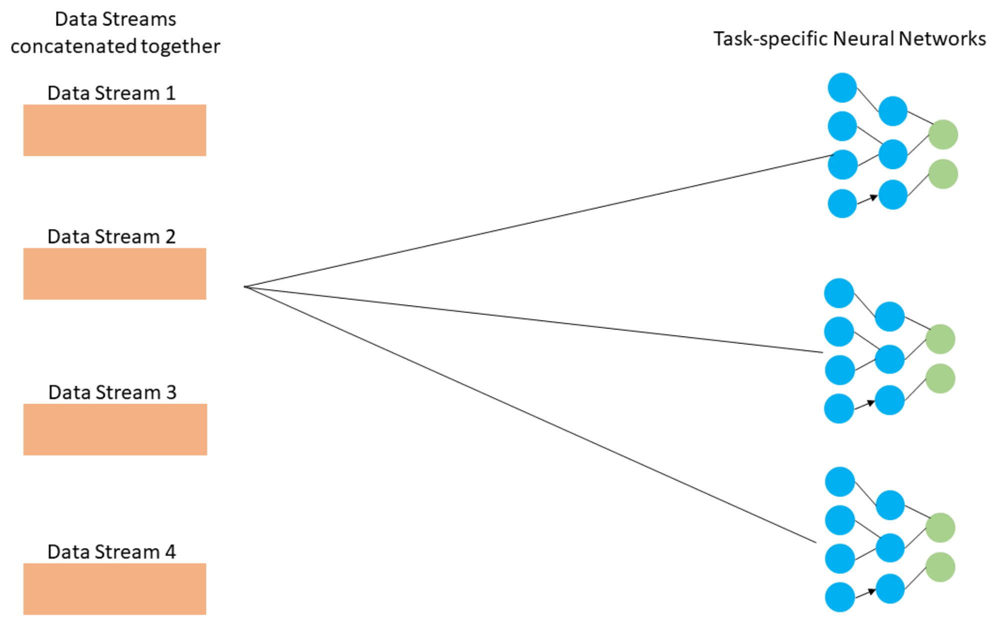

Data Streams concatenated together
Data Stream 1
Data Stream 2
Data Stream 3
Data Stream 4
Task-specific Neural Networks

Figure 10: Network Design 1: Base Models.

Network 2: Modular Neural Networks, Separate Latent Spaces For this network design, each data stream is preprocessed separately via autoencoder unsupervised learning. Latent space features are concatenated as inputs to task-driven neural networks. Figure 11 illustrates this design.

This design is very modular. If this design performs well, it is hypothesized that adding data streams (which would be preprocessed as well) would involve re-training only the higher level task-driven neural networks, rather than the preprocessing AE networks.

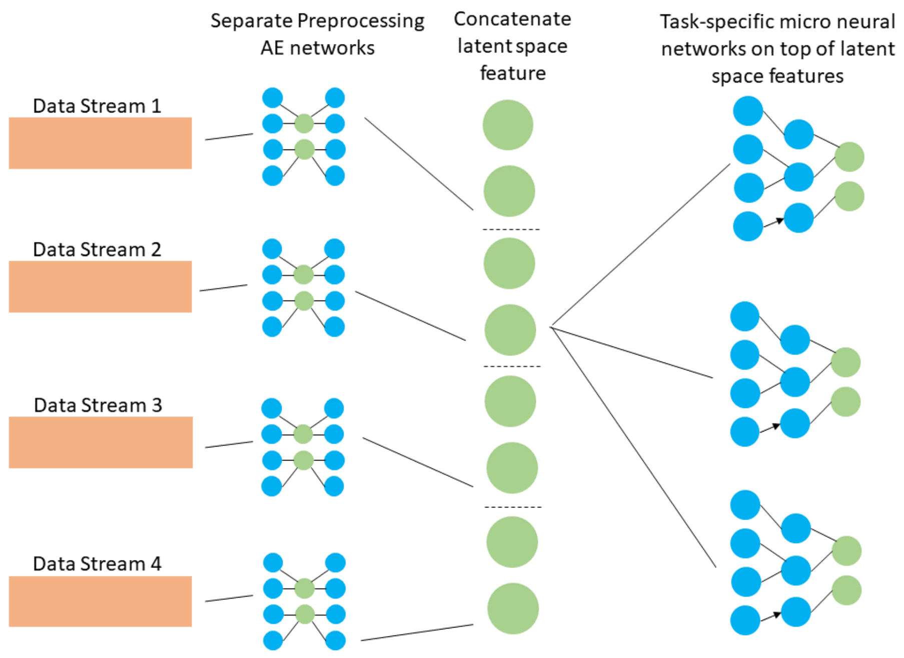

Separate Preprocessing AE networks
Concatenate latent space feature
Task-specific micro neural networks on top of latent space features
Data Stream 1
Data Stream 2
Data Stream 3
Data Stream 4

Figure 11: Network Design 2: Modular Neural Networks with Separate Latent Spaces.

Network 3: Modular Neural Networks, Shared Latent Space In this network design, data streams are preprocessed separately via an AE. An additional network processes the features into a shared latent space representation. The latent space representation will be used as input to task-driven neural networks. Figure 12 illustrates this design.

This design is also modular, but to a lesser degree as it incorporates a shared fusion space. It is worth investigating how the shared fusion space would develop (and potentially need to be re-trained) as more data streams are added.

Network 4: Non-Modular Neural Networks, Shared Latent Space For this network design, data is pre-processed together into a shared latent space. The latent space representation will be used as input into task-specific neural networks. Figure 13 illustrates this design.

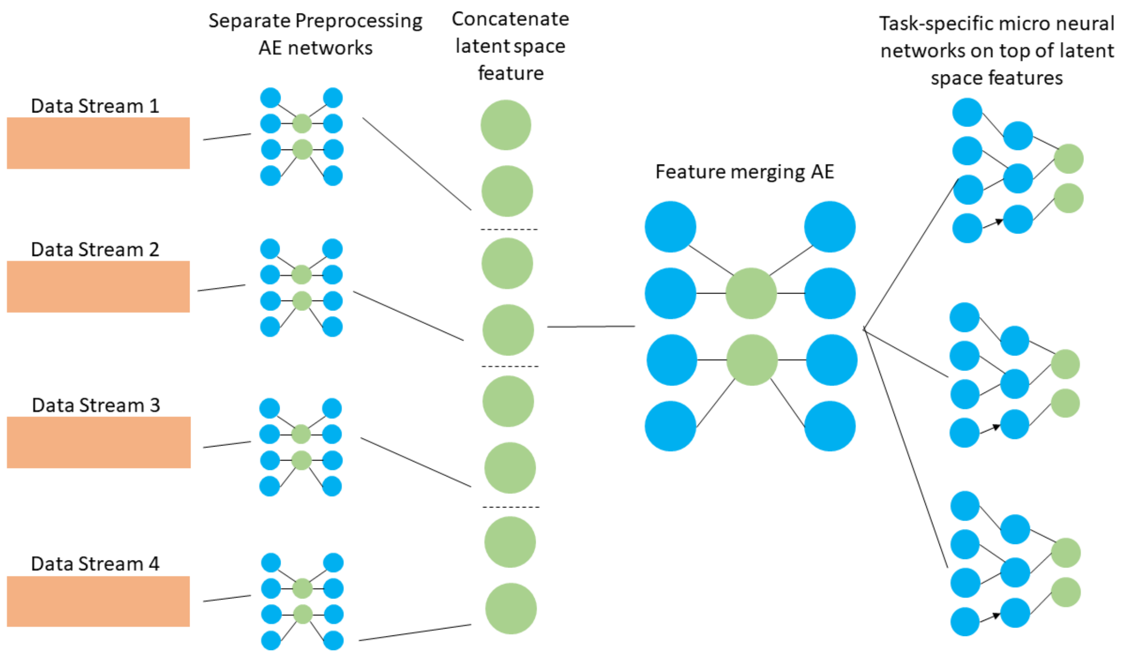

Separate Preprocessing AE networks
Concatenate latent space feature
Task-specific micro neural networks on top of latent space features
Data Stream 1
Data Stream 2
Data Stream 3
Data Stream 4
Feature merging AE

Figure 12: Network Design 3: Modular Neural Networks with Shared Latent Spaces

This design does not separate data streams by different preprocessing networks but preprocesses all data streams together. This design trains task-specific networks on top of a shared latent space. For future work, it would be worth investigating if adding data streams requires retraining both the shared latent space and the task-specific networks, or the shared latent space alone.

Adding Additional Data Streams All network types will be evaluated by withholding and then adding a new data stream. Performance of these networks will be scrutinized with regard to the potential difficulty of retraining or re-integrating the network. Decreasing costs associated with adding data to existing trained models is one of the main goals for the research.

Model Metrics

After training the different network models, there needs to be metrics in place to evaluate them. The model metrics will be used to compare different network design tasks performance with each other for the same dataset. The performant models will be com-

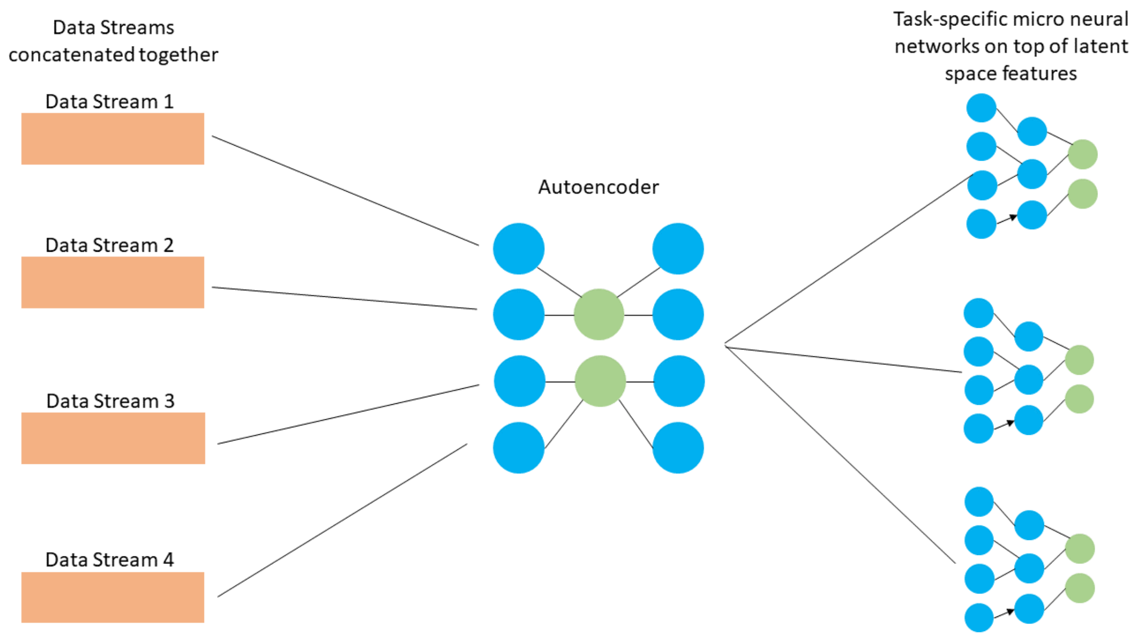

Data Streams concatenated together
Data Stream 1
Data Stream 2
Data Stream 3
Data Stream 4
Autoencoder
Task-specific micro neural networks on top of latent space features

Figure 13: Network Design 4: Non-Modular Neural Networks with Shared Latent Spaces

pared across datasets to see if different network designs have similar results across data domains.

Three different types of model metrics are needed for training and evaluation. To begin, a method is needed to determine if the preprocessing autoencoder networks create good latent space representations of the data. After this, the task-specific metrics need to be established. There are two types of tasks: forecasting or regression, where a new data sample is created based on input data sample(s) and classification or prediction, where a class is identified for input data. These specific tasks will vary by dataset. The different metrics for these tasks are explained further below.

Autoencoder Metric: The primary evaluation of performance for an autoencoder is reconstruction error, which will be modeled with Mean Squared Error (MSE). Mean Squared Error (named aptly: it takes the squared mean difference between the actual observation value and predicted observation value) allows for quantification of the deviation of the reconstructed input with the actual input.

For a vector x with elements x_1 to x_m, and a reconstructed vector y with elements y_1 to y_m, Equation 3 gives the calculation for MSE.

$$MSE = \frac{1}{n}\sum_{n=1}^{m}(x_n - y_n)^2 \tag{3}$$

Regression Task Metric: Forecasting tasks compare predicted values with actual values. For forecasting tasks, the MSE metric will also be used.

Prediction Task Metric: For classification tasks, a slightly different metrics will be used to evaluate model performance. Tracked metrics will include the accuracy and loss of the model, as well as true and false positives and negatives, precision, recall, and the overall F1 statistic (2 multiplied by precision * recall divided by precision plus recall) [163].

Experimental Design

To evaluate the proposed network designs on the chosen datasets with regard to the research questions, several experiments will be conducted. A given dataset will be trained on each of the four network models at least twice: once without training modifications, and once with adding in a new data stream.

The experiments will be conducted in the following steps:

1. Prediction tasks will be identified for each dataset.

2. The dataset will be taken and broken into a training, test, and validation set.

3. The dataset first will be trained with an LSTM network approach, (Network Design 1) without unsupervised learning. The model will be evaluated on the appropriate metrics (as explained in the previous section) for the task at hand. A model will be trained for each prediction task separately (if more than one is identified), and one model will be trained for all prediction tasks simultaneously.

4. The dataset will be split into unimodal data streams, and each stream trained on the Network Design 1.

5. The dataset will be split into unimodal data streams, and each stream will be trained on an autoencoder neural network in an unsupervised manner. The latent spaces will be concatenated together and used as input to task-specific neural networks (Network Design 2). The models (and the AE) will be evaluated on the appropriate metrics for the task at hand.

6. The dataset will again be split into unimodal streams and separately trained on an autoencoder. The latent space features will then be fed as input to a feature merging autoencoder, and its latent space features will be used to train the prediction networks (Network Design 3) The same metrics will be used to evaluate the models.

7. The dataset will NOT be split into separate modalities but used as input to one autoencoder network (all data streams will be preprocessed together rather than separately). The latent space features will be used to train the prediction network and the model will use the same metrics as before.

8. One modality will be taken out of the dataset, and the experiments will be repeated. The modality will then be added back into the network, and performance will be compared to the initial full-data run.

4.2 Genetic Algorithms Focus

This subsection looks at general principles and research questions behind the explainable AI focus of this dissertation applied to QARM and genetic algorithms.

One issue with neural network approaches is that there are limits on the explainability of neural network models. While there has been some work in studying feature activation and other methods of determining neural network behavior, networks are still by and large mostly black box models. While performant, the sheer number of parameters in a neural network makes it difficult to comprehend exactly what function is being fitted to the data.

In many contexts, a model does not necessarily need to be understood (in terms of where the many weights and biases landed) to be useful. There are other contexts, however, where a black box model is not preferred. In such cases, non-neural network explainable artificial intelligence approaches should be explored.

Quantitative association rules take the form outlined in Equation 4.

$$feature_1[v_{11}, v_{12}], feature_2[v_{21}, v_{22}], ..., feature_n[v_{n1}, v_{n2}] \rightarrow outcome_o[v_{o1}, v_{o2}] \quad (4)$$

Where:

1. $feature$ refers to a specific feature in the dataset,
2. v_{n1} refers to the lower bound of the value of feature n (inclusive),
3. v_{n2} refers to the upper bound of the value of feature n (inclusive),
4. *outcome* refers to a specific feature of the dataset of interest for correlation.

Essentially, a rule in this format states a correlation between specific features in certain value ranges and an outcome feature between another value range. In some cases, a particular value of a feature is sought, in which case its possible for $v_{n1} = v_{n2}$.

A rule is simply a statement with no inherent meaning about the quantification or usefulness of the relationship between the right side of the rule (rule antecedent or rule body) and the left side of the rule (rule consequent or rule head). In order to determine if the rule is a "good" rule, there must be a way of determining how often the stated relationship occurs compared to the rest of the data. There are several metrics of determining this. Three of the more common include support, confidence, and lift.

We'll define:

1. *num_whole_rule*: The number of times the antecedent and consequent appear together (are correlated) in the dataset, within the bounds set for each feature in the rule.
2. *num_dataset*: The number of records in the entire dataset.

3. *num_antecedent*: The number of times in the dataset the items in the antecedent appear between its set bounds in the rule.

4. *num_consequent*: The number of times in the dataset the items in the consequent appear between its set bounds in the rule.

The equations for support, confidence, and lift are given in Equations 5, 6, and 7.

$$supp(rule) = \frac{num\_whole\_rule}{num\_dataset} \tag{5}$$

$$conf(rule) = \frac{num\_whole\_rule}{num\_antecedent} \tag{6}$$

$$lift(rule) = \frac{conf(rule)}{supp(consequent)} \tag{7}$$

Support gives the percentage of how often every item in the rule occurs out of every item in the entire dataset.

Confidence gives the occurrence of the entire rule over just the antecedent; i.e., out of all the occurrences of the antecedent, how many times did the consequent also occur?

Lift gives the ratio of the confidence of the rule over the support of just the rule head. This gives a measure of how much more likely than expected the antecedent and consequent occur together. A value close to 1 indicates that this is a normal co-occurrence, and the rule is not very interesting. A positive value very different from 1 indicates that this is a more unexpected occurrence, and the rule is more likely to be correlated.

High support, confidence, and lift bring credibility to the idea that the antecedent and consequent are associated with each other.

There are different ways of performing QARM to obtain rules, but this dissertation in particular looks at doing so with GAs. The GA operations with QARM will be presented in subsequent sections.

The advantage of rules mined with a QARM GA is that the output is explainable (it looks at combinations of factors between different value ranges) and very easy to implement for real-time predictions (as it is a series of comparisons).

Sequence Extension

While there has been plenty of prior work associated with QARM GAs, less attention has been focused on using this information for time-sequenced data. One work did apply these concepts to time-sequenced data when it came to QARM GA, evolving a time barrier separately from the features to note the ranges in which the features occurred together [65]. This dissertation extends this work by evolving a time range for each feature separately.

A rule for this extension therefore takes the form in Equation 8.

$$\begin{aligned} &feature_1[v_{11}, v12][t_{11}, t12], feature_2[v_{21}, v_{22}][t_{21}, t22], ..., \\ &\quad feature_n[vn1, v_{n2}][t_{n1}, tn2] \rightarrow outcome_o[v_{o1}, v_{o2}] \end{aligned} \tag{8}$$

Where t_{n1} refers to the feature n sequence lower bound before the consequent occurrence, and t_{n2} refers to the feature n sequence upper bound before the consequent occurrence.

Determining the consequent support for this extension is exactly the same as before, as there is no consequent time evolution. For determining the entire rule support, the entire rule is considered present, if, for the consequent, each feature n is present in the associated bounds at least t_{n1} time steps ago and up to t_nn2 steps ago. For determining only the antecedent support, every time the features in the dataset occur in the time and value patterns noted with regard to a record is considered an occurrence of the antecedent.

Research Questions

This aspect of the dissertation mainly looks at the performance of QARM GA compared to neural network approaches as well as the performance of different QARM GA methods compared to each other. The research questions are:

1. What is the performance of a QARM GA predictor compared to a neural network predictor?

2. How modular can we make a QARM GA?

3. For time-variable datasets, is there an advantage to using a sequence extension on QARM?

QARM GA Process

There are many ways to set up a QARM GA. In the models used for this dissertation, the GA was broken up into the following classes:

QARM GA Parameter

A QARM GA Parameter refers to a single parameter within a rule. A parameter has the following properties:

1. upper bound: if not provided, it is calculated as the maximum value of the parameter within the dataset;
2. lower bound: if not provided, it is calculated as the minimum value of the parameter within the dataset;
3. current upper bound: the current value upper limit of the parameter;
4. current lower bound: the current value lower limit of the parameter;
5. mean: the mean of the parameter in the dataset; and
6. stdev: the standard deviation of the parameter from the mean in the dataset.

   If the experiment includes a sequence extension, the following are also present:

7. sequence upper bound: the maximum value a sequence can take, typically user-specified. The lower bound is assumed zero;
8. current sequence lower bound: the minimum limit of the time range the parameter is currently taking; and
9. current sequence upper bound: the maximum limit of the time range the parameter is currently taking.

The following operations can be performed on a parameter:

1. Mutation: The mutation changes either one of the value bounds or one of the sequence bounds. If the parameter is part of the sequence extension, there is an equal chance of choosing a value or sequence mutation. One of the values has a random amount added or subtracted.

2. Value bound change: Part of a mutation, this picks a random amount between zero and the mutation amount parameter multiplied by the standard deviation of the parameter and adds or subtracts it to the chosen bound. If there is a restriction on the range of the parameter, the new range is checked.

3. Sequence change: Part of a mutation, this adds or subtracts one or two timesteps to the chosen sequence bound. If there is a restriction on the sequence range on the parameter, the new range is checked.

4. Initialization: A parameter is randomly initialized between its bounds (to be within range, if required).

QARM GA Rule

A QARM GA Rule is made up of one or more parameters in the parameter class (as the antecedent). The rule knows the consequent information and occurrences as well, in order to be able to calculate all rule metrics. Additionally, the rule has the following properties:

1. whole rule number: the number of times in the dataset the rule occurs;

2. antecedent number: the number of times in the dataset the antecedent; occurs

3. consequent number: the number of times in the dataset the consequent; occurs

4. applicable records: the number of potential items in the dataset; (adjusted appropriately for sequence size, if applicable);

5. support: support for the rule;

6. confidence: rule confidence value;
7. lift: rule lift value;
8. antecedent indexes: the indexes in the dataset where the rule antecedent can be found;
9. consequent indexes: the indexes in the dataset where the rule consequent can be found;
10. whole rule indexes: the indexes in the dataset where the whole rule (antecedent and consequent) can be found; and
11. fitness: calculated fitness of the rule. This is typically a weighted combination of one or more of the previous properties.

Additionally, the following operations can take place on the rule:

1. Initialization: The rule is randomly initialized to have one or more parameters. The number of parameters at the rule initialization may be limited by the initial rule limit parameter. The parameters forming the rule are randomly initialized as well;
2. Mutation: The mutation can either add or subtract a parameter or change the bounds of a parameter. The add/subtract percent and change percent parameters are integers representing the percent change of that particular mutation occurring. If add/subtract is chosen, a random new parameter is either added to the rule (randomly initialized) or a randomly chosen existing parameter is subtracted. If the rule only has one parameter, adding is the only option. If the bounds change mutation is selected, a parameter is randomly chosen within the rule and its bounds are changed; and
3. Fitness calculation: the rules support, confidence, and lift are calculated, as well as its overall fitness. The fitness function is chosen based on the fitness function index parameter. If the sequence penalty or range penalty parameters are true,

the index of the exact sequence penalty or range penalty are chosen and applied to the fitness. These penalized a rule from having too large of a domain. The penalty is applied based on the average range and average sequence length if more than one parameter is active. Fitness is calculated after initialization and after mutation. If after mutation an antecedent support is zero, the mutation is retried up to max mutation tries (another setting). If it is still unsuccessful, the mutation is not applied.

QARM GA Population

A population in a QARM GA run for these experiments is made up of rules. It has the following properties:

1. rules list: This is the list of rules that a population contains. The number of rules is set in the population size setting;
2. top rules list: This is a list of rules with the highest fitness across all generations. The size of the top rules list is set by top rules setting;
3. mutation rate: The rate at which to mutate members of the population;
4. generations: The number of generations to run the population;
5. tournament size: The size of the selection tournament pool for the next generation;
6. dominance: if true, this removes dominated rules with lower fitness from the population; and
7. reseed from best: if true, this reseeds some of empty population slots from the top rules list.

Additionally, the population has the following operations:

1. Initialization: The population randomly creates a list of rules to fill a list of rules to the length set by the population size parameter;

2. Update top rules: the population sorts its rules by fitness and compares the top number of rules in the current generation to the top number of overall rules. If the rules are not the same, it adds the more fit rules from the current generation to the top rules;

3. Update and kill dominated rules: If dominance is true, this kills rules whose value bounds are already contained within other rules and whose fitness is lower;

4. diversify top rules: If true, this keeps only the highest fitness rule with the same types of active parameters in a population's top rules list (This is a somewhat greedy technique to diversify but keep good performers);

5. tournament selection: this randomly chooses a number of rules (set by the tournament size setting) from the population and copies the highest fitness rule to the new population;

6. Mutation: This chooses a number of random rules equal to the mutation rate percent from the population and mutates them;

7. generation run: This performs the following operations:

    (a) Updates and kills dominated rules;

    (b) Sorts the rules population by fitness;

    (c) Diversifies the top rules, if set;

    (d) Updates the top rules;

    (e) If the reseed from best setting is set, population members are replaced with a 10% of being from the top rules and a 90% chance of being randomly initialized. If the reseed from best setting is not set, the population members are replaced by randomly initialized new population members;

    (f) a tournament selection creates a new population from the existing population; and

    (g) the new population is mutated.

8. save: rules (either all rules or top rules) are saved to a javascript object notation file; and

9. experiment run: a population of rules is created, a number of generations equal to the generation setting is run, and all rules in the last generation as well as the top rules are saved to a javascript object notation file.

QARM GA Predictor

A QARM GA Predictor is given a test dataset and a list of top rules. For each rule, the predictor:

1. Determines where in the dataset the features (according to the rule antecedent) would predict the consequent;

2. Checks those predictions against the true consequent values in the dataset; and

3. Outputs accuracy, precision, recall, true positives, false positives, true negatives, false negatives, and the F1 score for each rule.

QARM GA Parameterization

There are many possible parameters for a QARM GA that are set by a user. The following are parameters for the QARM GA:

1. mutation rate: the percent of population members that will be mutated in a generation;

2. mutation amount: the percent of the standard deviation of a mutation parameter that will be upper limit of the amount of change for a parameter on a give mutation;

3. range restriction: If set, this is the percent of standard deviation of the parameter that the total range of the value upper and lower bound must be within;

4. range penalty: If true, this applies a range penalty to the fitness. The actual penalty applied is set by the range penalty index. The idea of range amplitude can be found in [61];

5. initial rule limit: The initial limit of number of parameters that can be in a rule when a population is created. Note that more can be added beyond this limit. This aids in a quick initialization;
6. add subtract percent: The percent chance that a mutation will add or subtract a parameter;
7. change percent: The percent chance that a mutation will change a parameter bound;
8. max mutation tries: The maximum number of tries to mutate a parameter before giving up;
9. population size: The number of members in the population;
10. top rules: The number of top rules to be kept across generations for a population;
11. generations: The number of generations for a population to run;
12. tournament size: The number of competitors in the pool for tournament selection;
13. dominance: If true, this kills rules with the same parameters and same bounds but lower fitness than an existing rule. The idea of dominance and non-dominated rules can be found in [62] and [61];
14. sequence limit: The upper limit on how far back the sequence can go for a parameter;
15. sequence penalty: If True, applies a penalty to the fitness based on how long the sequence is. The exact penalty to be applied is set by sequence penalty index. This is extended from the idea of penalizing the range amplitude;
16. diversify top rules: If true, this only keeps the highest fitness rule in a list of rules with the same active parameters;
17. reseed from best: if true, missing population members have a chance of being replaced from the top rules list;

18. fitness function index: The index of the fitness function used to calculate the overall fitness of the rule;

19. sequence penalty index: The index of the sequence penalty to apply to the rule; and

20. range penalty index: The index of the range penalty to apply to the rule.

A Note on Sequence Antecedent Calculation

It should be noted that the calculation of the antecedent number when it comes to sequences is not always intuitive, since each parameter can take on their own sequence values. In this subsection, the way the model code is running to make these calculations is explained more fully. The process to calculate the antecedent number is as follows:

1. First, a parameter is taken out of the rule;

2. Second, every index for where the parameter is within its value bounds is found in the dataset;

3. Next, the sequence value range is calculated for that parameter;

4. Next, for each index of the parameter, every value within the sequence range is added to create a new index list. For example, if the parameter occurs within its values at index 2, 3, and 5, and if its sequence range is [2, 3], then the new index list looks like [2+2, 2+3, 3+2, 3+3, 5+2, 5+3] for a new index list of [4, 5, 5, 6, 7, 8];

5. The new index list is reduced to the set of its elements. For our previous example, the new index list looks like [4,5,6,7,8]. These represent that records in the database for which the parameter fulfills both the sequence and values requirements. In other words, for indexes 4,5,6,7, and 8, the parameter occurs within its values 2 or 3 timesteps ago for all of them.

6. For any other parameters, the process of getting the fulfilled index list is repeated;

7. The lists of all parameters are then intersected to get the final fulfilled indexes. For instance, for our previous example, if one other parameter was present and this parameter fulfilled indexes [4, 7, 9, 10], the final fulfilled indexes would be [4, 7] for this rule; and

8. The length of this final list is the number of antecedents. The number of possible antecedents is found by taking the maximum lower bound of the sequence and subtracting it from the number of records in the database. (Since an antecedent can't be filled earlier than this).

This approach can be implemented in python with numpy arrays, which was a considerable boost to performance compared to repeated dataframe queries. This approach also makes calculating the whole rule support much faster, as the consequent indexes are merely intersected with the fulfilled rule indexes. This speed of this approach depends on the dataframe being organized by timestep with no missing timesteps, however.

Metrics

Since the QARM GA results can be used to create predictions, the metrics used to evaluate their performance will be the same used for the deep learning prediction task, namely, the F1 score. False and true positives and negatives as well as overall accuracy will also be tracked.

Computational Complexity

Genetic algorithms computational complexity compared to neural networks is difficult to determine, as both depend on size and hyperparameter choice. It should be noted, however, that the models in these experiments ran more slowly than the neural network models but required much less up-front data wrangling to implement. Additionally, the predictors that are ultimately output by the models are much less computationally intensive than neural networks to create predictions. What parameters are more important to a user when it comes to training and implementation timing, is worth further investigation.

4.3 Methodology Summary

This section outlined the methodology and experimental design for both the modular deep learning networks and the explainable QARM genetic algorithm models, and discussed the research questions behind each and the methods to evaluate them. The deep learning models use 4 network types in 4 separation schemes to investigate the impact of modularity with regards to datastreams and location. The genetic algorithms use sequence and non-sequence models on just prediction tasks with sets of hyperparameters to find highly correlated rules in the data. Both focus areas have specific metrics to evaluate their importance against the research questions. The next section looks a how the experiments will be set up for each task.

5 Experimental Setup

This section describes how the neural network and GA experiments were set up and run. In particular, it goes over the data formatting, data separation, experiment models and model groupings, and experiment parameters for each focus area. As before, this section of the paper separates the information by focus area, describing the modular deep learning models first and the QARM GA models next. The section also describes in detail some of the analysis techniques used and assumptions made when determining the final models performance in the results section.

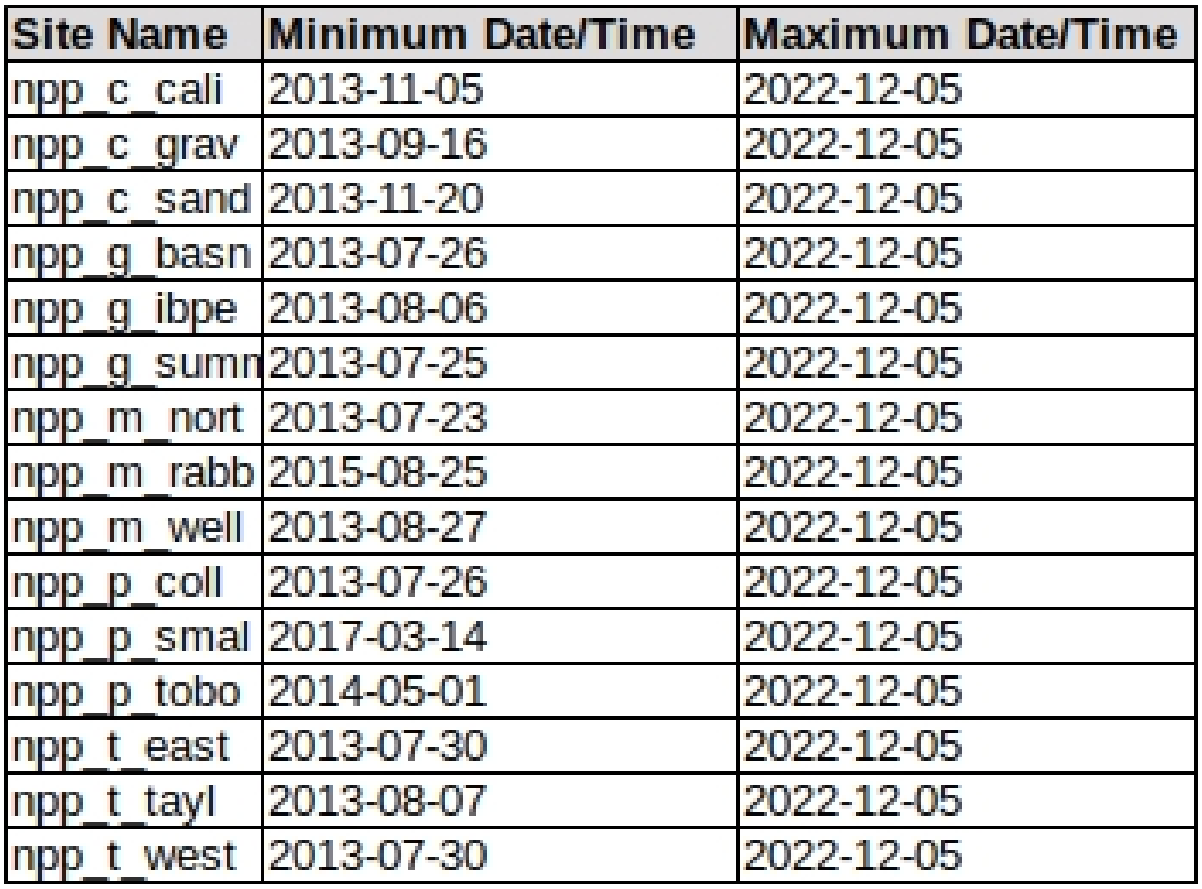
5.1 Jornada Basin: Deep Learning Models

The Jornada Basin is located in the Chihuahuan desert in New Mexico. The Jornada Basin Long Term Ecological Research Project maintains a substantial data catalog of ecological information, including weather data, net primary production, vegetative cover, rodents, and more. For this dissertation, the 15 NPP meteorological sites were chosen, which included the following sites: 'c_cali', 'c_grav', 'c_sand', 'g_basn', 'g_ibpe', 'g_summ', 'm_nort', 'm_rabb', 'm_well', 'p_coll','p_smal', 'p_tobo', 't_east', 't_tayl', and 't_west'. The first reporting date for any of the sites was July 23, 2013, and the last date used for all the sites was December 5, 2022. Refer to Figure 14 for exact start and end dates for all sites.

The amount, kinds, and start/stop time for different weather data at the different sites varied. Four data streams were chosen for the neural network models, encompassing 15 total features. The features are outlined in the Figure 15 below.

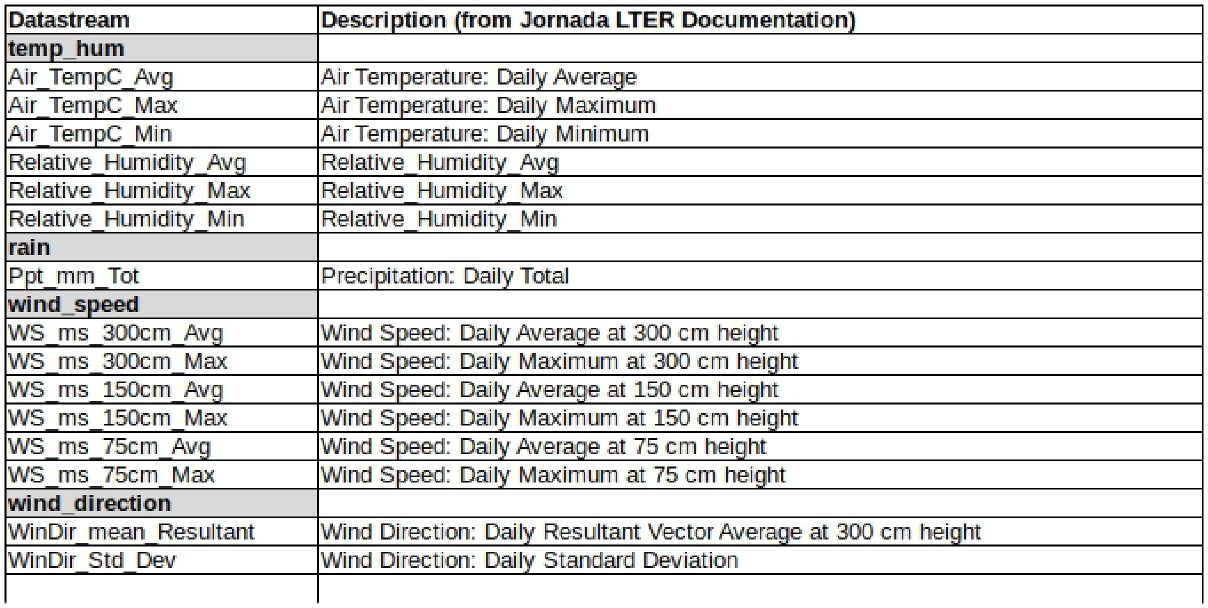
The deep learning models in this dissertation examine the impact of adding and subtracting different datastreams, which is why the features are broken up according to Figure 15. While these features were likely collected at the same weather station, the datastream split is a reasonable estimation of a component-level separation.

The Jornada Models also made use of several separation schemes, separating modalities out by datastream, location, or both. Figure 16 outlines the separation schemes for these models.



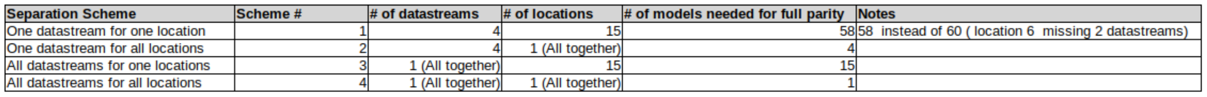
Site Name	Minimum Date/Time	Maximum Date/Time
npp_c_cali	2013-11-05	2022-12-05
npp_c_grav	2013-09-16	2022-12-05
npp_c_sand	2013-11-20	2022-12-05
npp_g_basn	2013-07-26	2022-12-05
npp_g_ibpe	2013-08-06	2022-12-05
npp_g_sumn	2013-07-25	2022-12-05
npp_m_nort	2013-07-23	2022-12-05
npp_m_rabb	2015-08-25	2022-12-05
npp_m_well	2013-08-27	2022-12-05
npp_p_coll	2013-07-26	2022-12-05
npp_p_smal	2017-03-14	2022-12-05
npp_p_tobo	2014-05-01	2022-12-05
npp_t_east	2013-07-30	2022-12-05
npp_t_tayl	2013-08-07	2022-12-05
npp_t_west	2013-07-30	2022-12-05

Figure 14: NPP site names for Jornada Basin dataset with start and end dates of observations.



Datastream	Description (from Jornada LTER Documentation)
temp_hum	
Air_TempC_Avg	Air Temperature: Daily Average
Air_TempC_Max	Air Temperature: Daily Maximum
Air_TempC_Min	Air Temperature: Daily Minimum
Relative_Humidity_Avg	Relative_Humidity_Avg
Relative_Humidity_Max	Relative_Humidity_Max
Relative_Humidity_Min	Relative_Humidity_Min
rain	
Ppt_mm_Tot	Precipitation: Daily Total
wind_speed	
WS_ms_300cm_Avg	Wind Speed: Daily Average at 300 cm height
WS_ms_300cm_Max	Wind Speed: Daily Maximum at 300 cm height
WS_ms_150cm_Avg	Wind Speed: Daily Average at 150 cm height
WS_ms_150cm_Max	Wind Speed: Daily Maximum at 150 cm height
WS_ms_75cm_Avg	Wind Speed: Daily Average at 75 cm height
WS_ms_75cm_Max	Wind Speed: Daily Maximum at 75 cm height
wind_direction	
WinDir_mean_Resultant	Wind Direction: Daily Resultant Vector Average at 300 cm height
WinDir_Std_Dev	Wind Direction: Daily Standard Deviation

Figure 15: Datastream features and descriptions for Jornada Basin LTER NPP meteorological sites, taken from the meta data documentation in their catalog [1]



Separation Scheme	Scheme #	# of datastreams	# of locations	# of models needed for full parity	Notes
One datastream for one location	1	4	15	58	58 instead of 60 (location 6 missing 2 datastreams)
One datastream for all locations	2	4	1 (All together)	4	
All datastreams for one locations	3	1 (All together)	15	15	
All datastreams for all locations	4	1 (All together)	1 (All together)	1	

Figure 16: Separation schemes for the Jornada Basin Dataset and the number of models needed for each scheme to reach parity of the full system.

Data Formatting

Not all locations in the Jornada Basin dataset have every datastream at every time, so the data was re-formatted in several ways. When there was a missing value in the dataset, it was forward filled (using the last valid observation to fill a future one) using the pandas fillna() method. If a feature was not present for a location (or barely present in the timeframe considered), that feature would be eliminated for a given model.

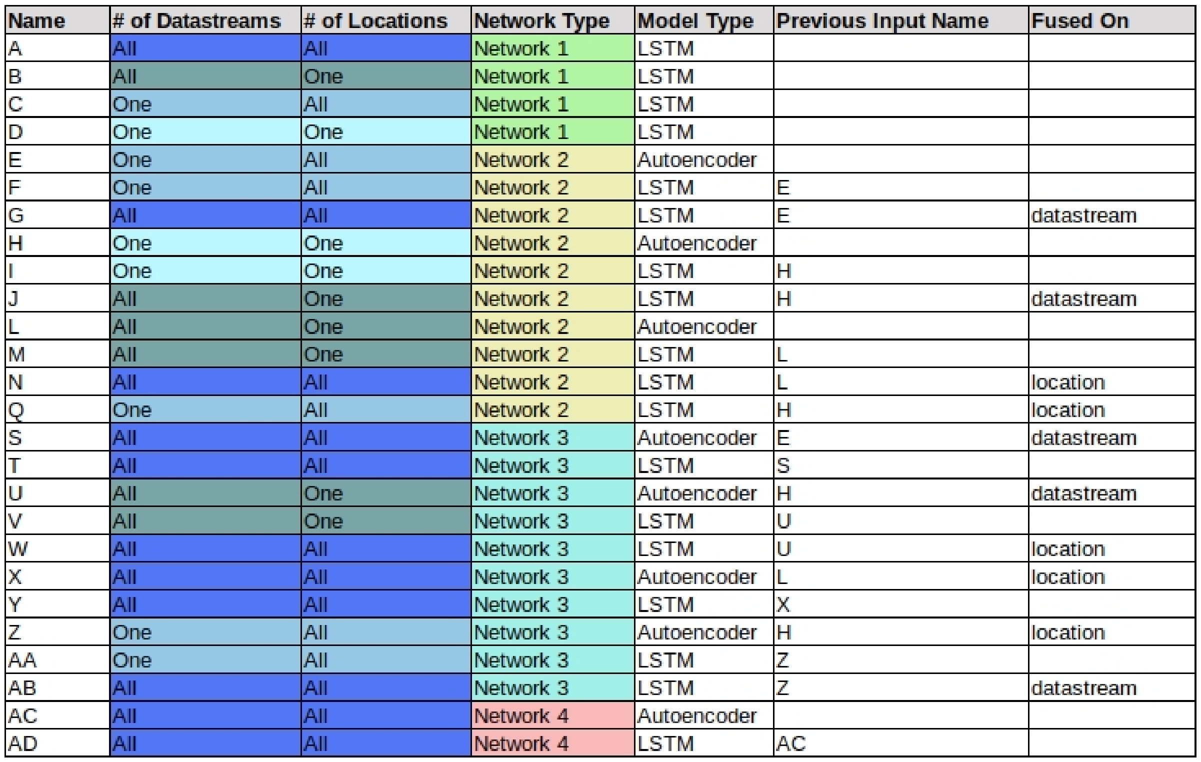
The Jornada Basin dataset is an interesting dataset in that most locations have a fair amount of data reported at the same time. Because of this, data features were predicted per-location. For example, the wind speed was predicted for the c-sand site and the wind speed was predicted separately for the g-basn site, rather than having one wind speed prediction for both sites. This technique made the most sense from site-specific precision agriculture perspective.

All data features were normalized to a value between 0 and 1 before being run through the model.

Models to Run

For the deep learning models, there are two ways to look at the kinds of models to run: in terms of what network they used to process the data, and in terms of how datastreams and locations were broken up. Most models eventually predicted all datastreams at all locations, but some models predicted one datastream at all locations, all datastreams at one location, or one datastream at one location. These models were run in order to look at aggregate performance of individual models after separating the prediction tasks. Figure 17 displays the descriptions of the models that could be run in a particular slate.

In Figure 17 the Number of Datastreams column refers to how many of the broad datastream categories were included for that model letter (1 or all). The Network Type refers to the networks in Figures 10 through 13 and the Model Type indicates whether or not an LSTM or autoencoder model was run for that letter. The Previous Input Name refers to prior autoencoder input, and the "Fused" on category is relevant for models predicting all of one datastream or location using input from autoencoders that only



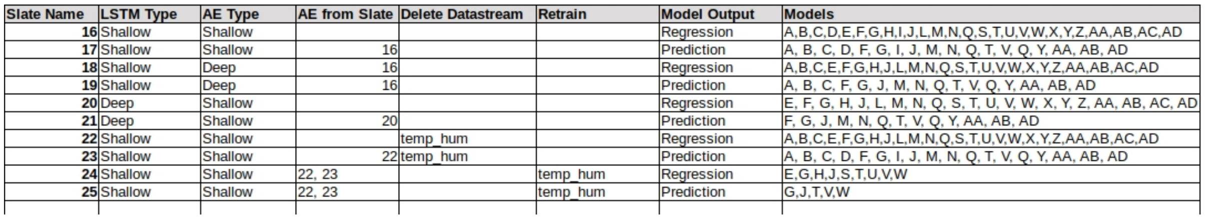
Name	# of Datastreams	# of Locations	Network Type	Model Type	Previous Input Name	Fused On
A	All	All	Network 1	LSTM		
B	All	One	Network 1	LSTM		
C	One	All	Network 1	LSTM		
D	One	One	Network 1	LSTM		
E	One	All	Network 2	Autoencoder		
F	One	All	Network 2	LSTM	E	
G	All	All	Network 2	LSTM	E	datastream
H	One	One	Network 2	Autoencoder		
I	One	One	Network 2	LSTM	H	
J	All	One	Network 2	LSTM	H	datastream
L	All	One	Network 2	Autoencoder		
M	All	One	Network 2	LSTM	L	
N	All	All	Network 2	LSTM	L	location
Q	One	All	Network 2	LSTM	H	location
S	All	All	Network 3	Autoencoder	E	datastream
T	All	All	Network 3	LSTM	S	
U	All	One	Network 3	Autoencoder	H	datastream
V	All	One	Network 3	LSTM	U	
W	All	All	Network 3	LSTM	U	location
X	All	All	Network 3	Autoencoder	L	location
Y	All	All	Network 3	LSTM	X	
Z	One	All	Network 3	Autoencoder	H	location
AA	One	All	Network 3	LSTM	Z	
AB	All	All	Network 3	LSTM	Z	datastream
AC	All	All	Network 4	Autoencoder		
AD	All	All	Network 4	LSTM	AC	

Figure 17: Descriptions of the different types of general models run in the Jornada Basin dataset. One or more of these models were run on each slate of experiments, with modifications specific to that slate.

used one. In these cases, the prior autoencoder inputs must be fused, either with regard to the datastream or the location.

Slates of experiments

The 26 model letters were run in different experimental slates. Not all slates ran all model letters. For instance, when adding a subtracted feature back into the data, only the models that broke up by a data feature made sense to retrain, since a fully trained version of the other model types already existed in a previous slate. Figure 18 describes the model slates for Jornada Basin.



Slate Name	LSTM Type	AE Type	AE from Slate	Delete Datastream	Retrain	Model Output	Models
16	Shallow	Shallow				Regression	A,B,C,D,E,F,G,H,I,J,L,M,N,Q,S,T,U,V,W,X,Y,Z,AA,AB,AC,AD
17	Shallow	Shallow	16			Prediction	A, B, C, D, F, G, I, J, M, N, Q, T, V, Q, Y, AA, AB, AD
18	Shallow	Deep	16			Regression	A,B,C,E,F,G,H,J,L,M,N,Q,S,T,U,V,W,X,Y,Z,AA,AB,AC,AD
19	Shallow	Deep	16			Prediction	A, B, C, F, G, J, M, N, Q, T, V, Q, Y, AA, AB, AD
20	Deep	Shallow				Regression	E, F, G, H, J, L, M, N, Q, S, T, U, V, W, X, Y, Z, AA, AB, AC, AD
21	Deep	Shallow	20			Prediction	F, G, J, M, N, Q, T, V, Q, Y, AA, AB, AD
22	Shallow	Shallow		temp_hum		Regression	A,B,C,E,F,G,H,J,L,M,N,Q,S,T,U,V,W,X,Y,Z,AA,AB,AC,AD
23	Shallow	Shallow	22	temp_hum		Prediction	A, B, C, D, F, G, I, J, M, N, Q, T, V, Q, Y, AA, AB, AD
24	Shallow	Shallow	22, 23		temp_hum	Regression	E,G,H,J,S,T,U,V,W
25	Shallow	Shallow	22, 23		temp_hum	Prediction	G,J,T,V,W

Figure 18: Descriptions of different model slates run with described modifications and parameters. The Models column refers to the models run on from Figure 17.

In Figure 18, model letters D and I are absent after the first two slates. Early results from the first two slates supported removing these models from subsequent slates (as the models were computationally complex and not performant). This will be discussed further in the results section of the dissertation.

Experiment Parameters

There were two types of models for the experiments: LSTMs and AEs. For both model structures, there were modifications depending on the experimental slate, which incorporated slightly different hyperparameters. This section discusses the different model hyperparameters depending on the experimental slate for a given model.

Each individual model was run with several variations in order to control for variability in some of the structure and hyperparameter choices. Each dataset would be run with two different amounts for input days (30 or 60) and two different amounts for output days (1 or 7), for 4 total pure dataset variations. Each LSTM model had three structure variations possible (base 8, 32, or 64). For AE models, only one structure variation (70% of input nodes) was trained.

There were a handful of static hyperparameters which did not change across the model structure or slate. These included the test set split (20% for all), the validation set split (20% for all), and the batch size, which was 32 samples.

The shallow LSTM model had one LSTM layer (with the default Keras tanh activation function) followed by a 20% dropout layer. The input and output layers where shaped to the desired formats but had no additional processing. The final activation of the LSTM model used the Rectified Linear Unit function, and the Adam optimization scheme. It would run for 100 epochs and using the MSE function as a loss function. For each run, it kept track of the MSE. Shallow LSTM models ran for 100 epochs.

The three variations of the shallow LSTM model used different numbers of nodes in the first and only LSTM layer in the model structure. The first variations used 8 nodes (base 8 structure), the second 32 (base 32 structure), and the final used 64 nodes (base 64 structure).

The shallow AE used only one latent layer (aside from identical input and output layers), with the number of nodes in the latent layer determined as 70% of the number of input features. The dense latent layer and final layer used a Rectified Linear Unit activation, and the loss function (and only metric tracked) for the AE was MSE. The AE also used the Adam optimizer. The shallow AE model ran for 100 epochs.

Slates 18 and 19 were the only slates to use a deep LSTM model. There were two differences between the deep LSTM model and the shallow model: model layer structure and number of epochs. The deep LSTM model ran for 150 epochs. The first variation (base 8 structure) had one LSTM layer with 24 nodes, followed by a 20% dropout layer, followed by an LSTM layer with 8 nodes, followed by a 20% dropout layer. The second variation (base 32 structure) had an LSTM layer with 64 nodes, followed by a 20% dropout layer, followed by an LSTM layer with 48 nodes, followed by a 20% dropout layer, followed by an LSTM layer with 32 nodes, followed by a 20% dropout layer. The final variation (base 64 structure) had an LSTM layer with 128 nodes, followed by a 20%

dropout layer, followed by an LSTM layer with 96 nodes, followed by a 20% dropout layer, followed by an LSTM layer with 64 nodes, followed by a 20% dropout layer. Figure 19 provides the structures for the deep and shallow LSTM variations.

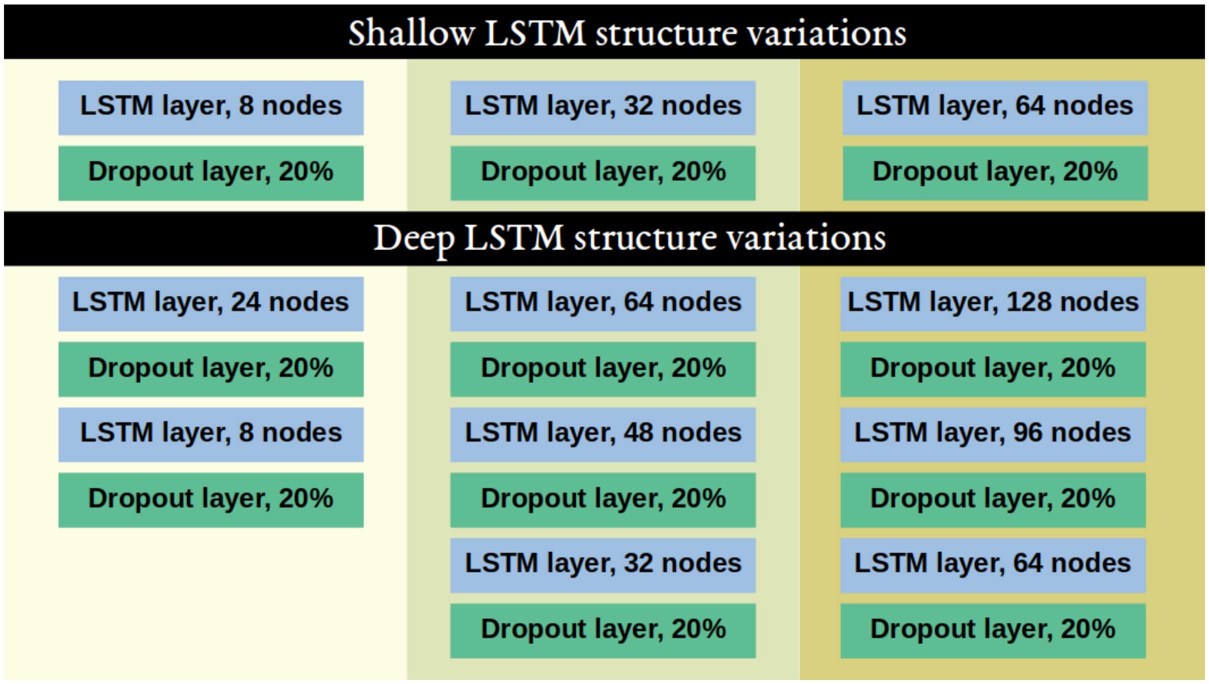


Figure 19: Model structures of deep and shallow LSTMs and variations.

Slates 20 and 21 were the only slates to use deep AE models. Like the deep LSTM model, the deep AE varied from the shallow model only in number of epochs run and model structure. The deep AE ran for 150 epochs and incorporated an additional dense encoding and dense decoding layer. Each of these layers used the rectified linear unit activation function and had 70% of the input nodes multiplied by 2. Figure 20 provides the structures for the deep and shallow AEs.

Slates 17, 19, 21, 23, and 25 used prediction models. The LSTM prediction models differed from the base shallow model only in the final activation function (sigmoid, in this case) and the metrics it tracked. The prediction LSTM tracked the MSE, but also the Binary Accuracy, Precision, Recall, True Positives, True Negatives, False Positives, and False Negatives. Since the Jornada Basin dataset is relatively unbalanced in terms of number of observations which actually record a frost event, the F1 statistic was used

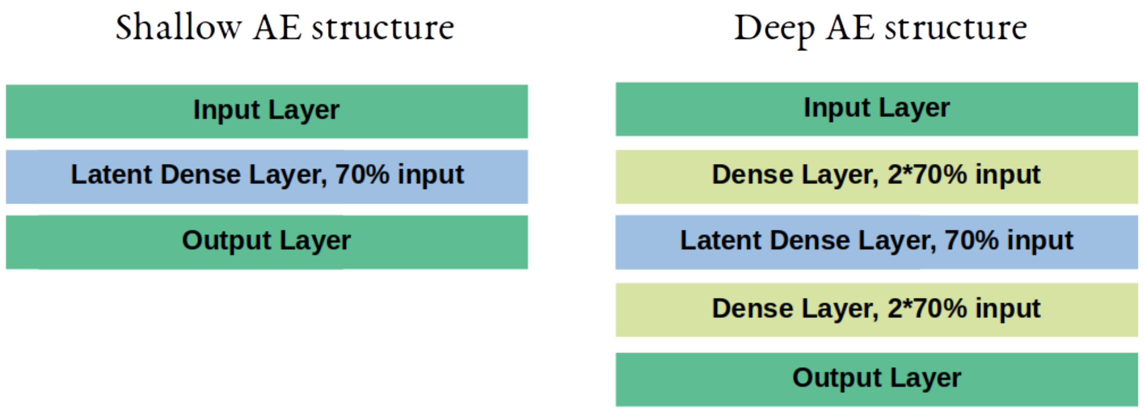


Figure 20: Model structures of deep and shallow AEs.

as the primary performance metric for predictions models.

It is not very typical to use MSE as the loss function for an LSTM model, but it was used here since many models had multiple binary predictions (presence or absence of frost), rather than having multiple categories of output.

The prediction models letter which used AE inputs re-used the AEs from the corresponding previous regression slate.

During training, all models used callbacks to monitor performance. Early stopping was incorporated for all models with a patience of 10 (model training would stop if the validation loss did not improve after 10 epochs). All models saved the best run over the total training and loaded the best model before evaluating it.

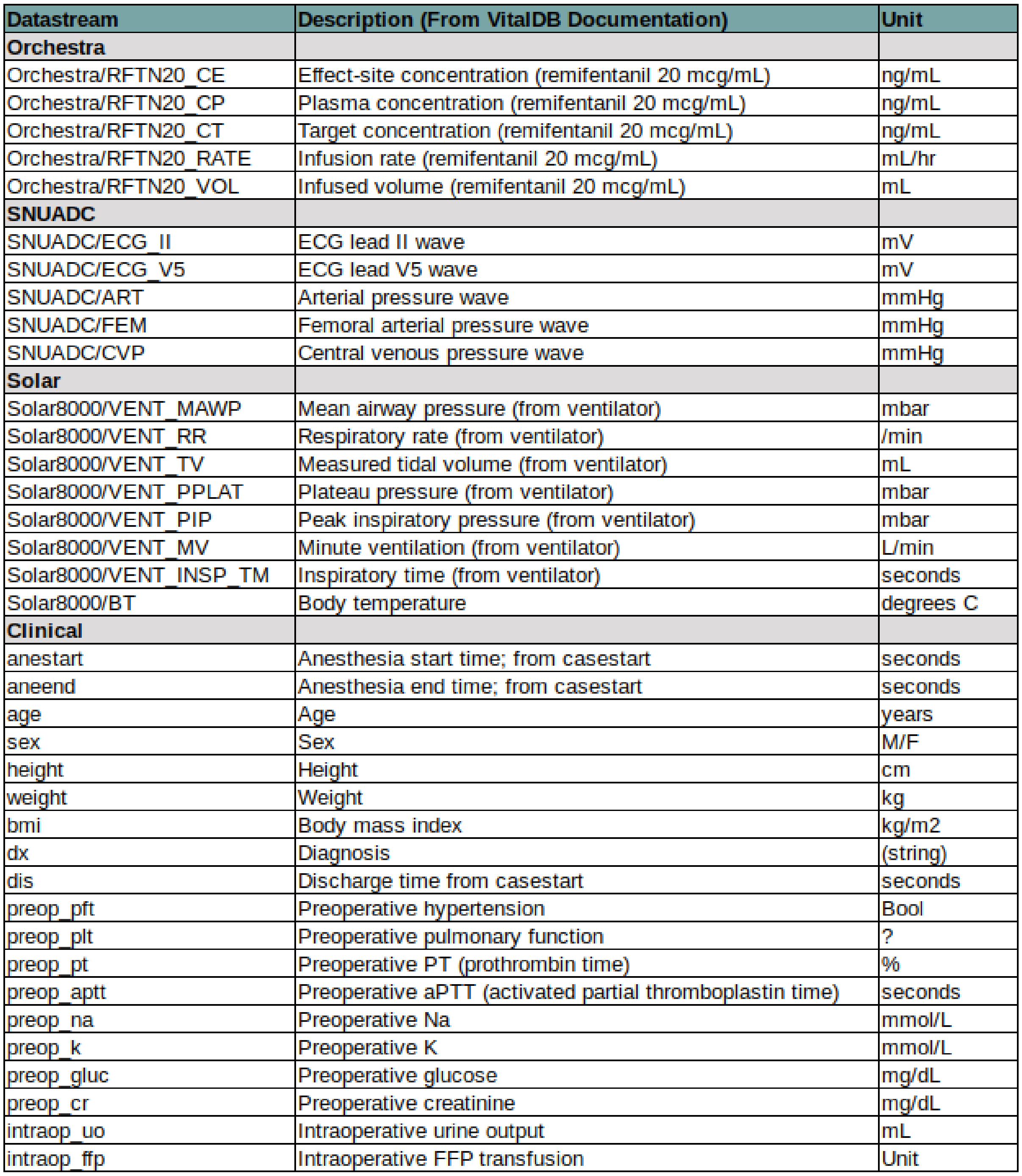
5.2 VitalDB: Deep Learning Models

The VitalDB dataset [2] is composed of non-cardiac surgery patient information and surgical patient monitoring recordings collected from Seoul National University Hospital in South Korea. There is an immense amount of data available from a total of 6,388 patient cases. The dataset includes 11 different patient monitors, clinical information, and lab results from each case. Patient monitoring data is timestamped and, in some cases, can be sampled at different rates.

Since this dissertation is investigating multivariate time-stamped data, a subset of cases were taken to include those with patient monitoring data from the Orchestra, Tram-Rac 4A (SNUADC) and Solar8000M patient monitors. A subset of clinical data was also given as input. The clinical data, since it is static, was not included as output for the predictions (since it is always an identity mapping from a previous observation), so there were a total of four input and three output datastreams for regression tasks for this dataset. Figure 21 displays the input features for the VitalDB experiments, broken up by heading of their associated datastream.

The resulting subset of VitalDB data includes 38 patients, all of whom are undergoing liver transplant surgery. These patients were selected since they contained all the data features of interest for this study. The diagnoses for these patients included Hepatocellular Carcinoma, Hepatitis B, Cirrhosis, Wilson Disease, Alagille Syndrome, and Biliary Atresia.

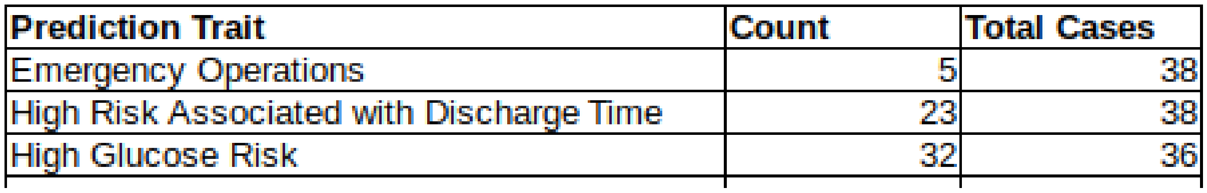
Identifying appropriate prediction tasks for these datasets was tricky in that the researcher is not overly familiar with the medical field or surgical operations. In order to accomplish this, the researcher informally consulted with two professionals in the medical field to identify potentially useful input parameters and gain insight on what might constitute a useful prediction. The professionals identified useful input parameters and advised searching the literature on liver transplant patient outcomes. From the literature, negative patient outcomes were noticed when the patient's postoperative hospital stay lasted longer than 14 days [164]. Negative patient outcomes were also associated with the patients glucose level spiking above 150 at any point during surgery [165]. From the patient data, these two risk factors were created. The "Discharge Mortality Risk "factor "dis_mortality_risk "was assigned 1 if the patient's discharge time exceeded 14 days from time of first recording patient info. This is not perfectly analogous to postoperative stay since it also includes the time in surgery but is a very close approximation. The "Glucose Risk" factor "gluc_risk "was assigned 1 if at any point the patient's glucose lab



Datastream	Description (From VitalDB Documentation)	Unit
Orchestra		
Orchestra/RFTN20_CE	Effect-site concentration (remifentanil 20 mcg/mL)	ng/mL
Orchestra/RFTN20_CP	Plasma concentration (remifentanil 20 mcg/mL)	ng/mL
Orchestra/RFTN20_CT	Target concentration (remifentanil 20 mcg/mL)	ng/mL
Orchestra/RFTN20_RATE	Infusion rate (remifentanil 20 mcg/mL)	mL/hr
Orchestra/RFTN20_VOL	Infused volume (remifentanil 20 mcg/mL)	mL
SNUADC		
SNUADC/ECG_II	ECG lead II wave	mV
SNUADC/ECG_V5	ECG lead V5 wave	mV
SNUADC/ART	Arterial pressure wave	mmHg
SNUADC/FEM	Femoral arterial pressure wave	mmHg
SNUADC/CVP	Central venous pressure wave	mmHg
Solar		
Solar8000/VENT_MAWP	Mean airway pressure (from ventilator)	mbar
Solar8000/VENT_RR	Respiratory rate (from ventilator)	/min
Solar8000/VENT_TV	Measured tidal volume (from ventilator)	mL
Solar8000/VENT_PPLAT	Plateau pressure (from ventilator)	mbar
Solar8000/VENT_PIP	Peak inspiratory pressure (from ventilator)	mbar
Solar8000/VENT_MV	Minute ventilation (from ventilator)	L/min
Solar8000/VENT_INSP_TM	Inspiratory time (from ventilator)	seconds
Solar8000/BT	Body temperature	degrees C
Clinical		
anestart	Anesthesia start time; from casestart	seconds
aneend	Anesthesia end time; from casestart	seconds
age	Age	years
sex	Sex	M/F
height	Height	cm
weight	Weight	kg
bmi	Body mass index	kg/m2
dx	Diagnosis	(string)
dis	Discharge time from casestart	seconds
preop_pft	Preoperative hypertension	Bool
preop_plt	Preoperative pulmonary function	?
preop_pt	Preoperative PT (prothrombin time)	%
preop_aptt	Preoperative aPTT (activated partial thromboplastin time)	seconds
preop_na	Preoperative Na	mmol/L
preop_k	Preoperative K	mmol/L
preop_gluc	Preoperative glucose	mg/dL
preop_cr	Preoperative creatinine	mg/dL
intraop_uo	Intraoperative urine output	mL
intraop_ffp	Intraoperative FFP transfusion	Unit

Figure 21: VitalDB dataset input features, broken up by datastream [2].

data exceeded 150. There were three patients that did not have any glucose lab data to work with, and so were not included in this particular prediction assessment. Finally, the “Emergency Operation ”status of the patient was chosen as a third prediction factor, which is provided natively in the dataset. Figure 22 outlines the different prediction features with counts of how many cases express each.



Prediction Trait	Count	Total Cases
Emergency Operations	5	38
High Risk Associated with Discharge Time	23	38
High Glucose Risk	32	36

Figure 22: Counts of cases with particular prediction factors.

Data Formatting

Similar to the Jornada Basin dataset, data preparation and cleaning was needed to make the dataset usable by the models. All categorical variables (which were numerous in the clinical data input) were encoded to numerical categories. Data was also all normalized to a value between 0 and 1 depending on the max or min value of the particular feature across all cases.

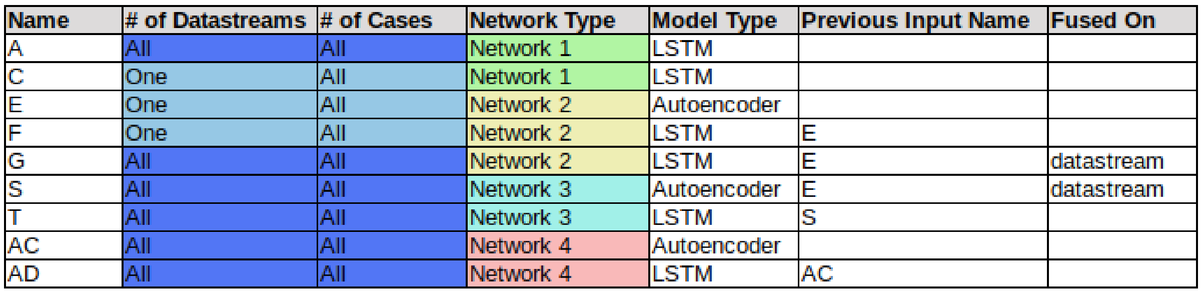
Some data features were recorded as numerical values, and others were waveforms that could be sampled at different rates. For these experiments, all data is sampled at 1 second intervals. Missing data is first forward filled using the pandas fillna() method (meaning a gap in data is replaced with the previous value) and then backward filled for remaining gaps (where a gap is data is replaced with the next valid value).

Not all features began recording at the same time, so the dataset is trimmed to only include the continuous period of time where all patient monitors were recording at the same time. The average amount of recorded observations (at the 1 second rate) for the cases was 19554.92, with a maximum of 32728 and a minimum of 5837. For model inputs, 30 seconds were used to generate either 10 seconds of output or predictions. The input and output values were determined based on the informal conversations with the medical professionals. For the ultimate models, one thirtieth of the data was used as input in order to be able to run the models.

Models to Run

The VitalDB dataset differs from the Jornada Basin dataset in that the surgeries took place at different times, for different lengths of time. For Jornada Basin, locations were mostly reporting the same kinds of data at the same time intervals. The VitalDB set does not have this property, and therefore all models that tried to predict or separate out by individual location are eliminated for these experiments. This also means no datastreams are predicted specifically for any one case but are generalized across all cases.

The same basic types of models were run for the VitalDB dataset as the Jornada Basin Dataset, without the models that relied on location. For consistency, the alphabet character model identifiers remained the same. Figure 23 displays the 9 types of models the VitalDB dataset ran.

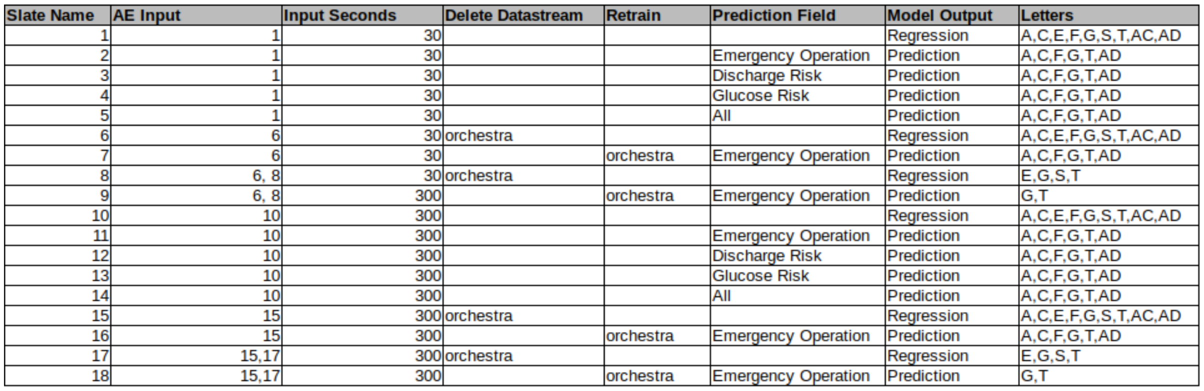


Name	# of Datastreams	# of Cases	Network Type	Model Type	Previous Input Name	Fused On
A	All	All	Network 1	LSTM		
C	One	All	Network 1	LSTM		
E	One	All	Network 2	Autoencoder		
F	One	All	Network 2	LSTM	E	
G	All	All	Network 2	LSTM	E	datastream
S	All	All	Network 3	Autoencoder	E	datastream
T	All	All	Network 3	LSTM	S	
AC	All	All	Network 4	Autoencoder		
AD	All	All	Network 4	LSTM	AC	

Figure 23: VitalDB models to run.

Slates of experiments

13 slates of experiments were run for the VitalDB dataset. Slate 1 ran the basic regression models for all Network types and splits of the data, while slates 2-5 focused on predictions. Slates 6-9 focused on the retraining problem with regression tasks, while slates 10-13 did the same with a prediction task. Figure 24 displays the slate information for VitalDB.



Slate Name	AE Input	Input Seconds	Delete Datastream	Retrain	Prediction Field	Model Output	Letters
1	1	30				Regression	A,C,E,F,G,S,T,AC,AD
2	1	30			Emergency Operation	Prediction	A,C,F,G,T,AD
3	1	30			Discharge Risk	Prediction	A,C,F,G,T,AD
4	1	30			Glucose Risk	Prediction	A,C,F,G,T,AD
5	1	30			All	Prediction	A,C,F,G,T,AD
6	6	30	orchestra			Regression	A,C,E,F,G,S,T,AC,AD
7	6	30		orchestra	Emergency Operation	Prediction	A,C,F,G,T,AD
8	6, 8	30	orchestra			Regression	E,G,S,T
9	6, 8	300		orchestra	Emergency Operation	Prediction	G,T
10	10	300				Regression	A,C,E,F,G,S,T,AC,AD
11	10	300			Emergency Operation	Prediction	A,C,F,G,T,AD
12	10	300			Discharge Risk	Prediction	A,C,F,G,T,AD
13	10	300			Glucose Risk	Prediction	A,C,F,G,T,AD
14	10	300			All	Prediction	A,C,F,G,T,AD
15	15	300	orchestra			Regression	A,C,E,F,G,S,T,AC,AD
16	15	300		orchestra	Emergency Operation	Prediction	A,C,F,G,T,AD
17	15,17	300	orchestra			Regression	E,G,S,T
18	15,17	300		orchestra	Emergency Operation	Prediction	G,T

Figure 24: VitalDB slates to run.

Experiment Parameters

As with the Jornada Basin Dataset, there were two basic types of networks run for VitalDB models: LSTMs and AEs.

Both networks had several hyperparameters in common. Both ran for 150 epochs, incorporating early stopping with a patience of 10 based on the validation loss. The best model, based on the validation loss, was loaded after training before evaluation. Both used a batch size of 128 due to the much larger data size. Both models used a test split of 20% and a validation split of 20%. Both models also made use of the Adam optimizer from the Keras library.

For regression tasks, LSTM models kept track of the MSE, MAPE, and MAE metrics, using MSE as the loss function. Regression LSTMs also used the Rectified Linear Unit function as the final activation, and the tanh function as the default LSTM activation. Prediction LSTM models kept track of the MSE, Binary Accuracy, Precision, Recall, True Positives, True Negatives, False Positives, and False Negatives metrics for each run, using MSE as the loss function. Prediction LSTMs also used the sigmoid function as the final activation.

There were three variations of LSTM structure run for each model. The first had an LSTM layer with 48 nodes followed by a 20% Dropout layer, an LSTM layer with 32 nodes, followed by a 20% Dropout layer, and an LSTM layer with 16 nodes followed by a 20% Dropout layer. The second used the same structure but with 192, 128, and 64 nodes in the LSTM layer, respectively. The final used 384, 256, and 128 nodes. Figure 25 displays the different LSTM structures for the VitalDB dataset.

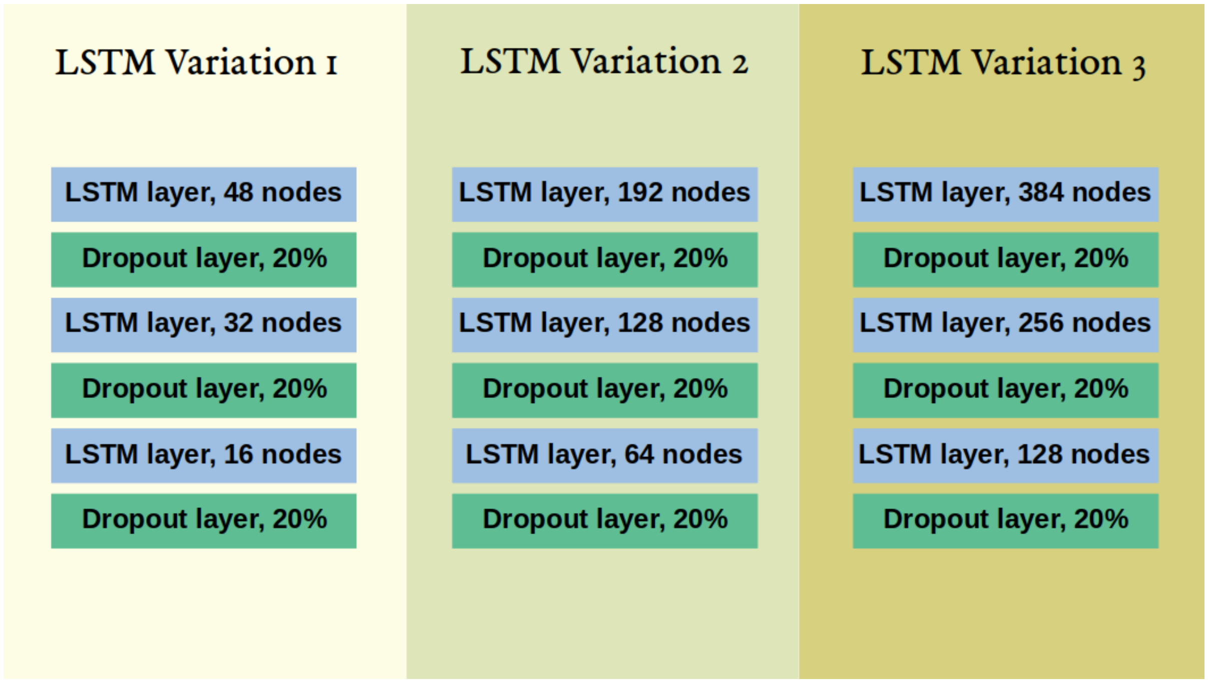


Figure 25: LSTM structure variations for VitalDB dataset

The AE models were reused between prediction and regression tasks. All AE models had a simple structure of three layers - one dense encoding layer composed of 70% the number of input nodes multipled by three, a dense latent space layer with 70% of input nodes, and one decoding layer with 70% of the input nodes multipled by 3. All dense layers used the Rectified Linear Unit activation function. Rectified Linear Unit was also used as the final activation function. Figure 26 displays the AE structure for the VitalDB dataset.

5.3 Jornada Basin - QARM GA Models

The QARM GA models for the Jornada Basin Dataset only ran a subset of similar models from the deep learning tasks. Since only positive association rules would be mined, these experiments looked only at the frost prediction task.

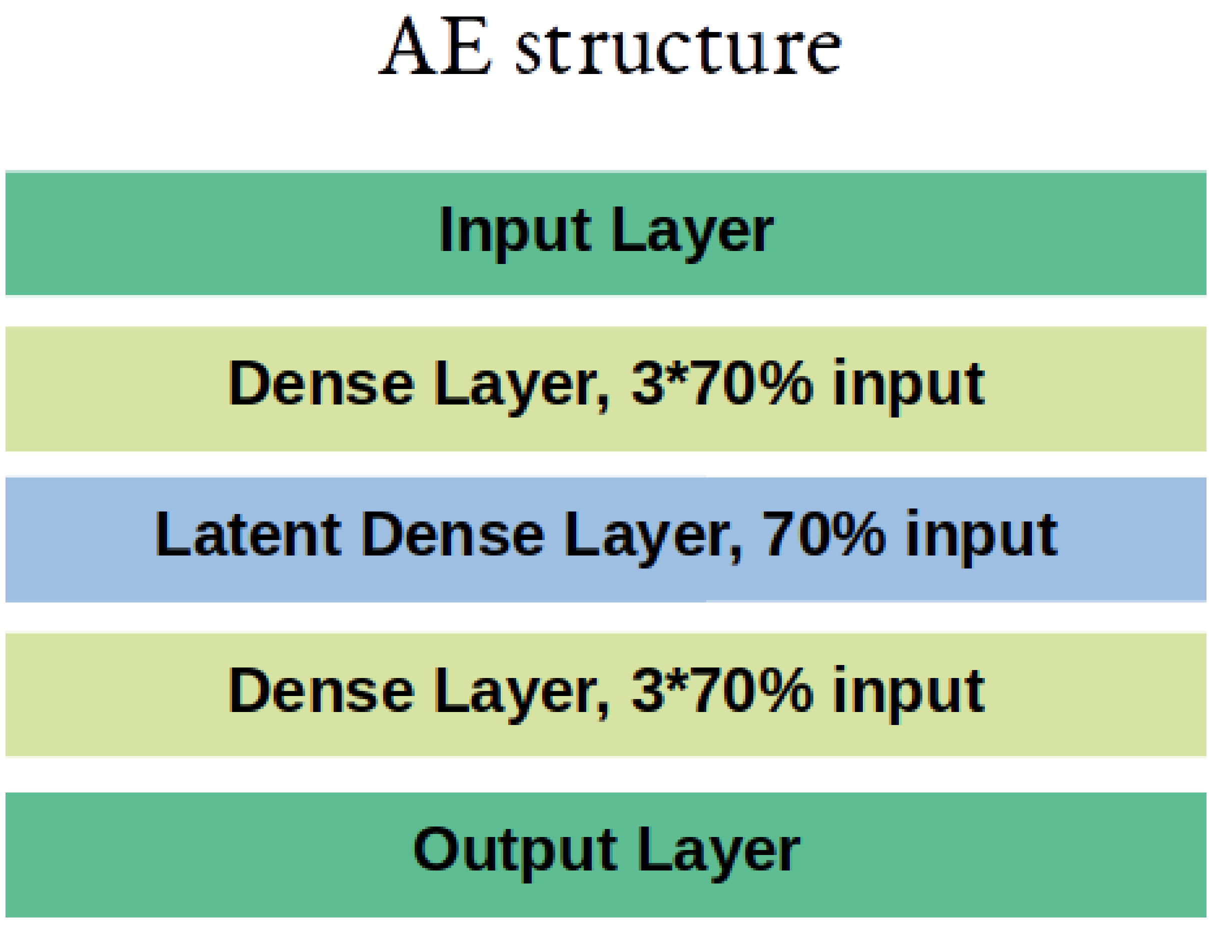


Figure 26: LSTM structure variations for VitalDB dataset.

Data Formatting

For the QARM GA, the data formatting was similar to the deep learning models, with a few exceptions. The data was not normalized for these algorithms. Additionally, the prediction columns of the dataframes used were shifted up one, so the prediction for the next day would be on the same row as the current day. This was done for convenience of allowing sequences to begin at zero. Filling “N/A ”data was not necessary for this task.

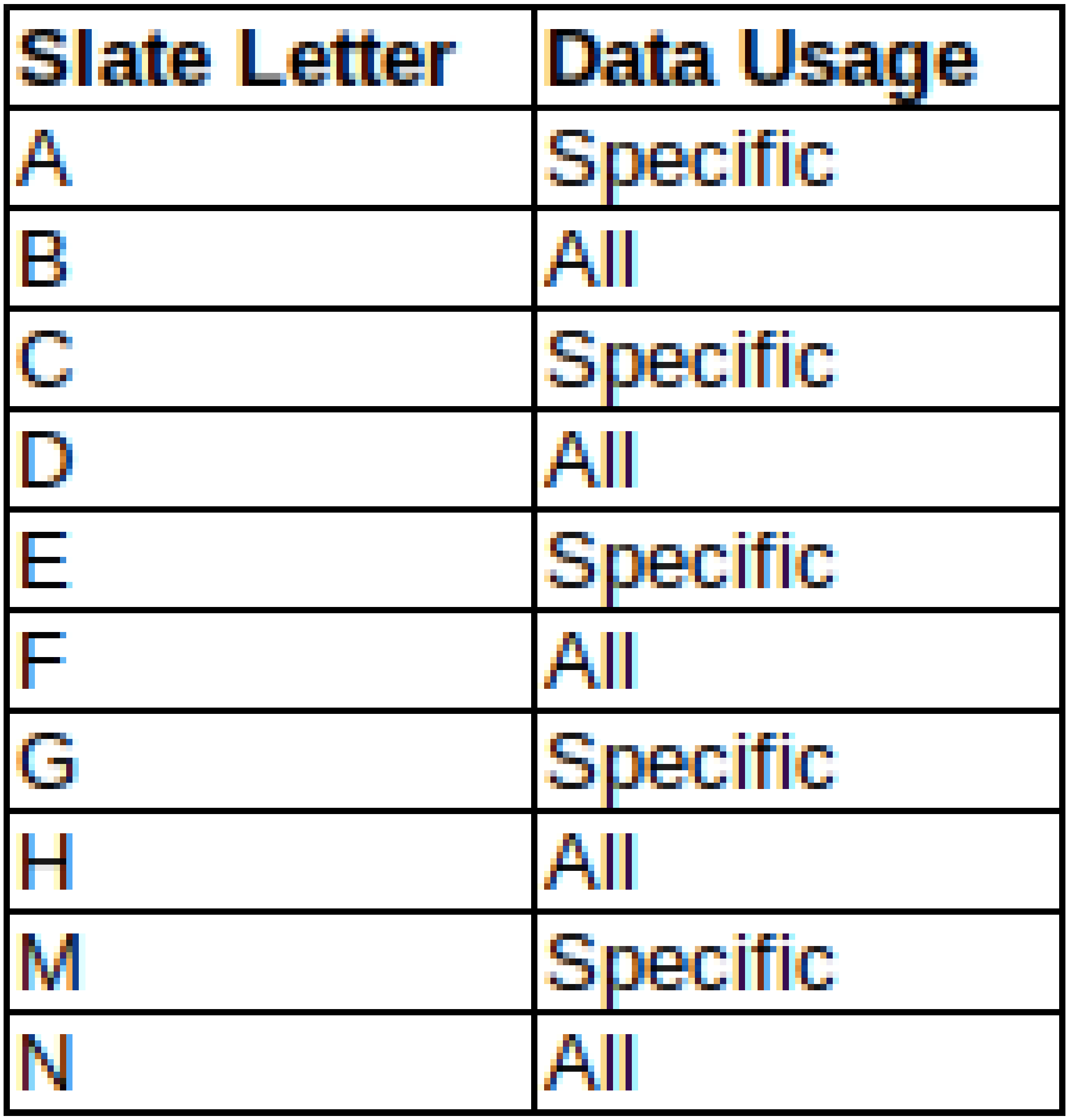
Models to Run

For the QARM GA Models, experiments were run according to a slate letter. Each slate letter ran 4 distinct sets of parameters, and each run was repeated 3 times. So, each letter ran a total of 180 genetic algorithms (15 sites * 4 sets of parameters, * 3 runs). Each slate letter had a non-sequence version, where the normal QARM GA was run using only one day prior of data for frost prediction, and a sequence version, where the number of past days the model considered was evolved per parameter in the dataset. Additionally, since this dissertation wanted to consider site-specificity versus using all available datastreams when it came to site-specific management, some letters used just the site data to predict the frost occurrence for that specific site, and others used all site data to predict the frost data for the specific site. Figure 27 displays which slate letters used all site data and which used specific site data.

Unlike the deep learning models, data-specific QARM GAs were not run for this set of experiments.

Experiment Parameters

While experiments were run with different sets of parameters, some of the parameters remained the same across runs. All models had a mutation rate and mutation amount set to 20, and no models restricted the range of the parameter. The models had an initial rule limit of 2, a 30% chance of adding or subtracting a parameter, and a 70% chance of changing a parameter bound when a mutation occurs. There is maximum limit for mutation tries of 5. All sequence experiments used a sequence limit of 12 (days prior).



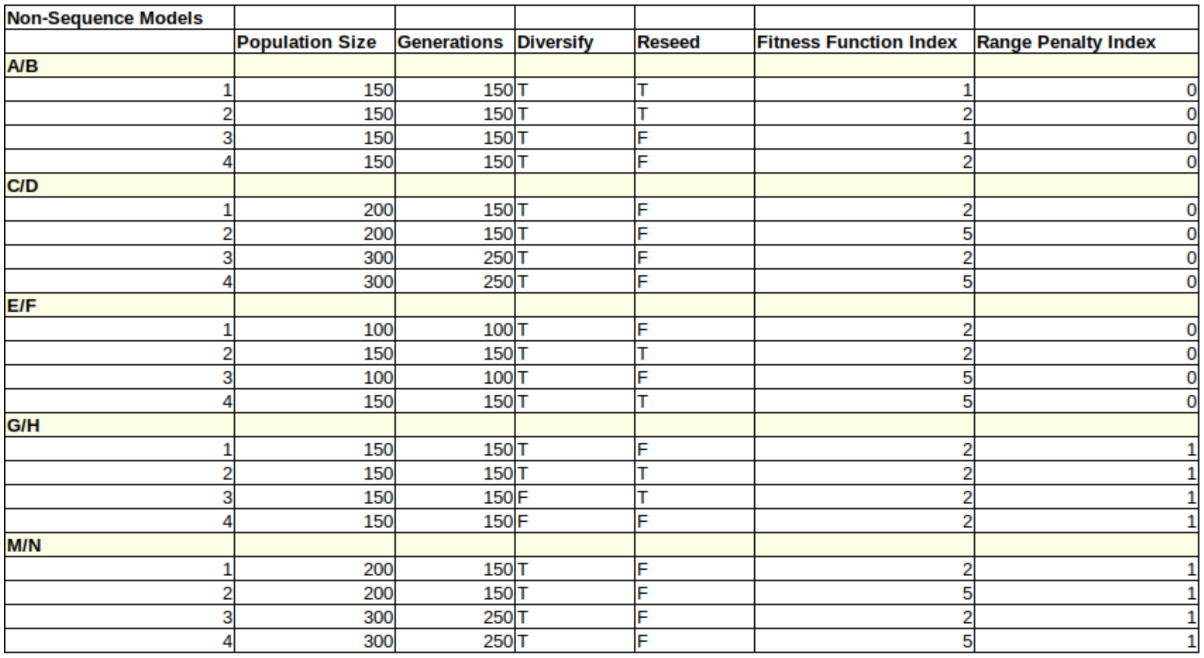
Slate Letter	Data Usage
A	Specific
B	All
C	Specific
D	All
E	Specific
F	All
G	Specific
H	All
M	Specific
N	All

Figure 27: Slates Letters and whether they used all or specific site data to make prediction

There was a selection tournament size limit of 4 population members. Killing dominated parameters was set to true.

Slates of Experiments

For the experiment sets, the population size, number of generations, diversification, reseeding from best, fitness function index, and range penalty indexes varied from model letter to model letter. For sequence models, the sequence penalty index also varied. Figure 28 gives the letters and parameters for the non-sequence models. Figure 29 gives the same information for the sequence models.

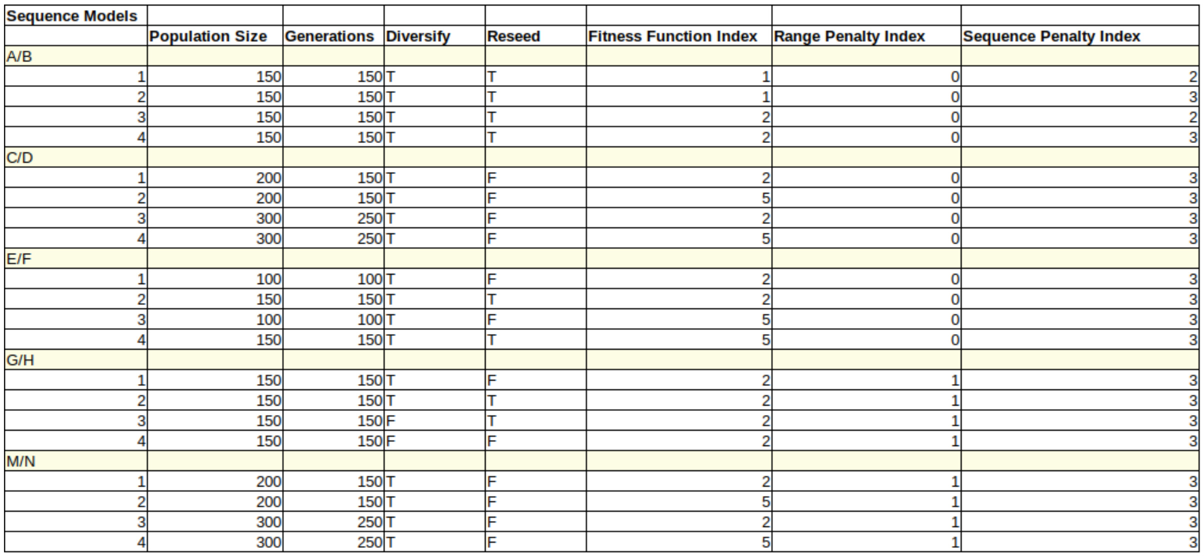


Non-Sequence Models						
	Population Size	Generations	Diversify	Reseed	Fitness Function Index	Range Penalty Index
A/B						
1	150	150	T	T	1	0
2	150	150	T	T	2	0
3	150	150	T	F	1	0
4	150	150	T	F	2	0
C/D						
1	200	150	T	F	2	0
2	200	150	T	F	5	0
3	300	250	T	F	2	0
4	300	250	T	F	5	0
E/F						
1	100	100	T	F	2	0
2	150	150	T	T	2	0
3	100	100	T	F	5	0
4	150	150	T	T	5	0
G/H						
1	150	150	T	F	2	1
2	150	150	T	T	2	1
3	150	150	F	T	2	1
4	150	150	F	F	2	1
M/N						
1	200	150	T	F	2	1
2	200	150	T	F	5	1
3	300	250	T	F	2	1
4	300	250	T	F	5	1

Figure 28: Slates Letters and parameter values for non-sequence models

For the sequence and non-sequence models, corresponding letters have the same parameters except for models A and B, since the sequence letters varied sequence penalty indexes while the non-sequence letters varied the reseeding parameter.

The equations for the respective range penalties, sequence penalties, and fitness functions are outlined below.



Sequence Models							
	Population Size	Generations	Diversify	Reseed	Fitness Function Index	Range Penalty Index	Sequence Penalty Index
A/B							
1	150	150	T	T	1	0	2
2	150	150	T	T	1	0	3
3	150	150	T	T	2	0	2
4	150	150	T	T	2	0	3
C/D							
1	200	150	T	F	2	0	3
2	200	150	T	F	5	0	3
3	300	250	T	F	2	0	3
4	300	250	T	F	5	0	3
E/F							
1	100	100	T	F	2	0	3
2	150	150	T	T	2	0	3
3	100	100	T	F	5	0	3
4	150	150	T	T	5	0	3
G/H							
1	150	150	T	F	2	1	3
2	150	150	T	T	2	1	3
3	150	150	F	T	2	1	3
4	150	150	F	F	2	1	3
M/N							
1	200	150	T	F	2	1	3
2	200	150	T	F	5	1	3
3	300	250	T	F	2	1	3
4	300	250	T	F	5	1	3

Figure 29: Slates Letters and parameter values for sequence models

Sequence Penalties Sequence penalty index 2's effect on fitness is outlined in Equation 9.

$$fitness = fitness - (1 * (0.5 * amplitude_{sequence}) + (0.8 * fulfillment_{early})) \quad (9)$$

where $amplitude_{sequence}$ is defined as the length of the sequence divided by the total possible length, averaged across the parameters in the rule, and $fulfillment_{early}$ is defined as the percentage backward the earliest bound occurs in the total sequence limit. Sequence penalty 3's effect on fitness is displayed in Equation 10.

$$fitness = fitness - (1 * (0.5 * amplitude_{sequence}) + (0.8 * fulfillment_{late})) \quad (10)$$

where $amplitude_{sequence}$ is defined as the length of the sequence divided by the total possible length, averaged across the parameters in the rule, and $fulfillment_{late}$ is defined as the percentage backward the latest bound occurs in the total sequence limit.

Range Penalties Range penalty index 0's effect on fitness is displayed in Equation 11.

$$fitness = fitness - 1 * (0.1 * amplitude_{bound}) \quad (11)$$

where $amplitude_{bound}$ refers to the average bound range out of the total bound possible bound range across all parameters in a rule.

Range penalty index 1's effect on fitness is displayed in Equation 12.

$$fitness = fitness - 0.2 * (0.1 * amplitude_{bound}) \tag{12}$$

Fitness Functions Fitness function index 1's calculation is given in Equation 13.

$$fitness = (2 * support * (num\_whole\_rule/num\_consequent)) * confidence \tag{13}$$

For fitness function 2, the calculation is given in Equation 14.

$$fitness = (5 * support + 0.5 * confidence) \tag{14}$$

For fitness function 5, Equation 15 gives the calculation.

$$fitness = (5 * support + 0.5 * confidence + 0.1 * lift) \tag{15}$$

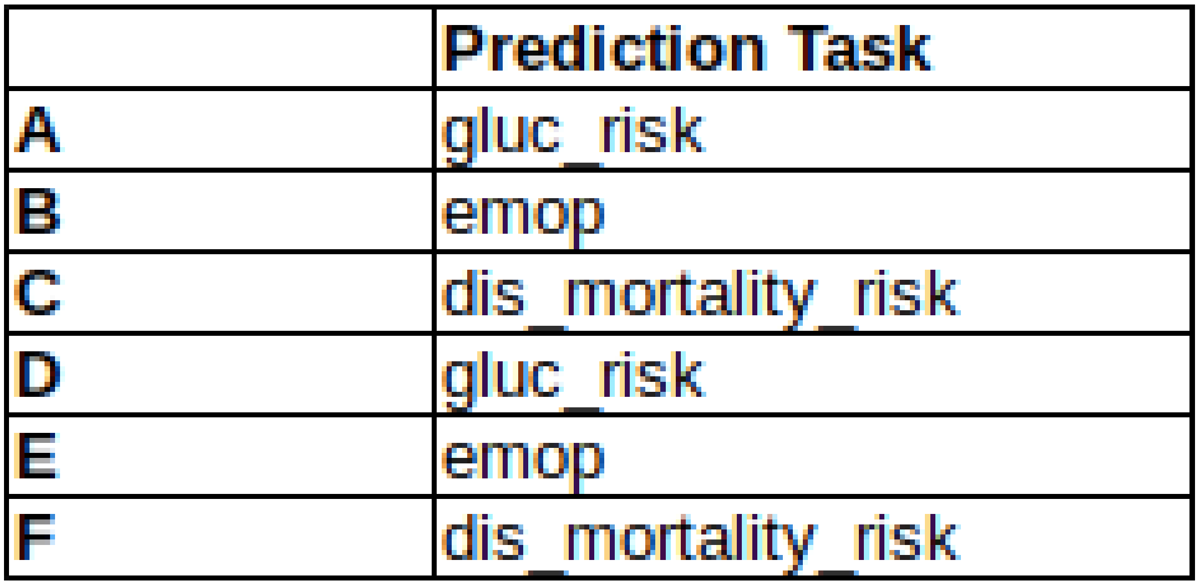
5.4 VitalDB - QARM GA Models

Data Formatting

For the QARM GA, the data formatting was similar to the deep learning models, as with Jornada. Again, the data wasn't normalized, and prediction columns were shifted upwards. N/A data had been forward and backward filled for prediction tasks as before with VitalDB. One slight difference in these models, however, is that the entire dataset was resampled to 1/30 of its size. Therefore, for sequences, 1 sequence timestep actually represents 30 total seconds.

Models to Run

As with the Jornada QARM GA models, experiments were run according to a slate letter, with 4 sets of parameters, for 3 runs each. Since VitalDB data is not separable by location (case), the models run all cases at once, for a total of 12 models per slate letter. Each slate letter also had a sequence and non-sequence version. Different slates letters had different prediction tasks, Slates A and D predicted the glucose risk, Slates B and E predicted an emergency operation, and Slates C and F predicted the discharge mortality risk. Figure 30 breaks down the slates by prediction tasks.



	Prediction Task
A	gluc_risk
B	emop
C	dis_mortality_risk
D	gluc_risk
E	emop
F	dis_mortality_risk

Figure 30: Slate letters and prediction tasks.

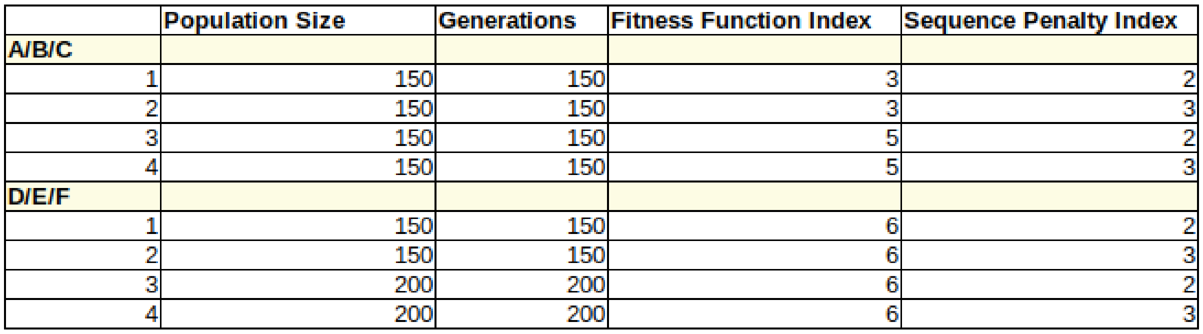
Experiment Parameters

All slates in these experiments used a mutation rate and amount of 20, no range restriction, range penalty 0 from Equation 11, and an initial rule limit of 2. As with Jornada, there was a 30% chance of adding or subtracting a parameter, and a 70% chance of changing a parameter bound when it came to mutation. Only 5 mutation attempts were allowed. The number of top rules to keep for the models was set at 10. The tournament size for selection was limited to 4 individuals. For sequence models, the sequence limit was 12 timesteps backward. Killing dominated parameters was set to true, as was diversifying the top rules. Reseeding the population from the best rules was always false.

Slates of Experiments

For the VitalDB slates, the number of population members, number of generations, and the fitness function index were varied across model letters and parameter sets. For sequence models, the sequence penalty index was also experimented with. Figure 31 gives the model letters and parameters for sequence and non-sequence models.

For non-sequence models, the sequence penalty index column is irrelevant and for a few letters more than one parameter set is the same due to this.



	Population Size	Generations	Fitness Function Index	Sequence Penalty Index
A/B/C				
1	150	150	3	2
2	150	150	3	3
3	150	150	5	2
4	150	150	5	3
D/E/F				
1	150	150	6	2
2	150	150	6	3
3	200	200	6	2
4	200	200	6	3

Figure 31: Slate letters and associated parameters.

For this set of models, the sequence penalties 2 and 3 are the same as in Equations 9 and 10. Fitness function 2 is the same as Equation 14 and Fitness function 5 is the same as Equation 15. Fitness functions 3 and 6 are outlined in Equations 16 and 17 respectively.

$$fitness = (2 * support * (num\_whole\_rule/num\_consequent)) * confidence * lift \quad (16)$$

$$fitness = (2 * support * 2 * confidence * 5 * (1 - lift)) \quad (17)$$

Data Reduction Error

Due to an error in comma separated value creation, the QARM GA Models for the VitalDB dataset ran with 4 less cases than the deep learning models: cases 4481, 55, 2332, and 6227. Only two of these cases had any positive prediction factors, both for the discharge mortality risk. Since this risk was found 23 of the original 38 cases, having 21 of the 34 cases with the discharge mortality risk had roughly the same balance as the deep learning set.

5.5 Experiment Analysis Techniques and Assumptions

For both the Jornada Basin and VitalDB dataset, the MSE metric was chosen as the evaluation for model performance for regression tasks and AEs. For prediction tasks, the

F1 metric was chosen, as the datasets tended to be unbalanced and the F1 statistic in this case is a better representation of model capability. The F1 and MSE metrics were based on the test set evaluation for a given model (20% of the dataset for each model was reserved for testing). This section describes the results of slates and models within slates for each dataset.

In the results and analysis below, slates break up the models to compare only those which predict the same inputs and outputs. Combinations of models are then used to find overall comparisons between input and output groups, when applicable.

Wilcoxon Signed-Rank Test

In addition to comparing mean, minimum, and maximum metrics between models in the group, it is useful to compare the statistical significance between the differences to gain a more accurate picture as to the performance of one versus the other. Each model had multiple variations, for both input/output day combinations (Jornada Basin only), model structures, and (for those models which broke up input by datastream or location) the particular index of the input. It was important to look at the differences between individual networks with exactly the same model parameters across the letters (i.e., same number of input days, same number of output days, same network structure, same input and output). For this reason, some kind of paired samples t-test was needed. In this case, the model letter was one nominal variable, and the type of individual neural networks within the model letter was the second nominal variable. The measurement variable was the overall test MSE of the network.

However, the difference distributions of two given model letters are highly non-normal. Because of this, a paired samples t-test could not be used (which has a null hypothesis of the mean differences between distributions as 0.) Instead, the Wilcoxon signed-rank test was used, which does not rely on the assumption of normality. This test uses a null hypothesis of median differences between distributions as 0. Information on the use of the Wilcoxon signed rank test was found in [166]. To perform the Wilcoxon signed rank test, the scipy.stats module was used.

Comparing Multiple Separate Models to One Combination Model

Some of the model letters in each slate only predict a subset of features (one datastream's worth), and in the Jornada Basin case, only for one location. This dissertation investigates any benefit or detriment to maintaining multiple separate models for specific datastreams or locations versus maintaining a larger model which predicts more features. In order to do this, the best separate models for the feature subset they operate on are chosen and analyzed for the same error metric as the larger combination model.

All separate model metrics had to be weighted. The outputs were multiplied by the number of features they predict, divided by the number of features the combination models predict before being added together. That allowed a comparison of the separated and combination models. Figure 32 illustrates this with a sample case for the MSE metric.

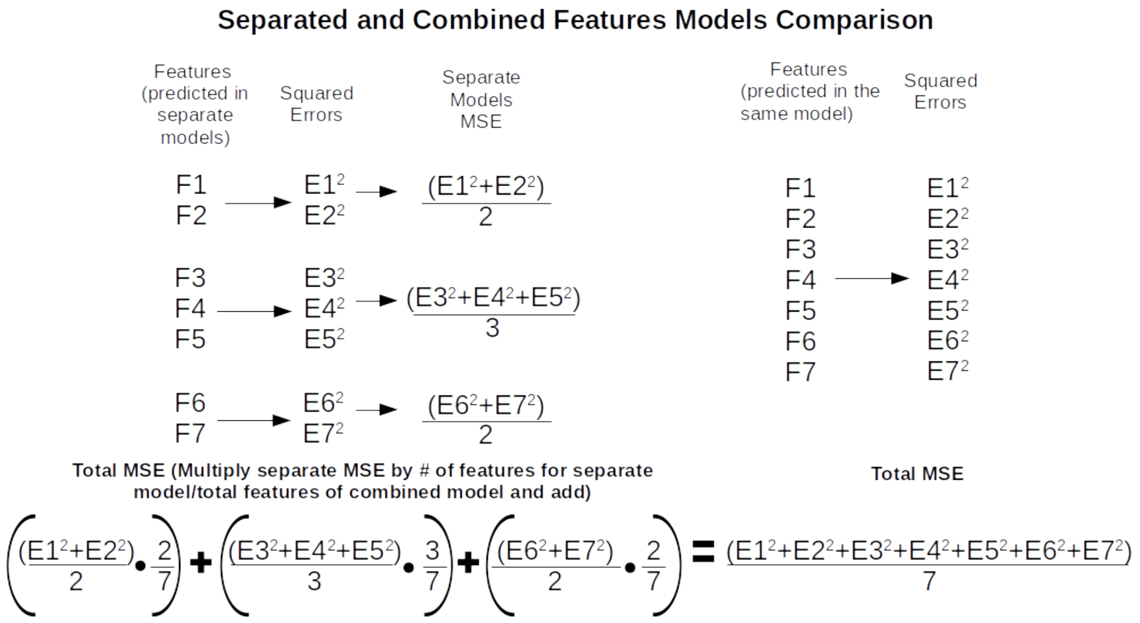

Separated and Combined Features Models Comparison
Features (predicted in separate models)
Squared Errors
Separate Models MSE
Features (predicted in the same model)
Squared Errors
F1 F2 F3 F4 F5 F6 F7
E1² E2² E3² E4² E5² E6² E7²
(E1²+E2²)/2
(E3²+E4²+E5²)/3
(E6²+E7²)/2
Total MSE (Multiply separate MSE by # of features for separate model/total features of combined model and add)
Total MSE
((E1²+E2²)/2 • 2/7) + ((E3²+E4²+E5²)/3 • 3/7) + ((E6²+E7²)/2 • 2/7) = (E1²+E2²+E3²+E4²+E5²+E6²+E7²)/7

Figure 32: Multiplying separate model MSE by the number of features they predict divided by the total features predicted by the combination model before adding them together creates the comparable MSE of the separate versus combination model.

5.6 Experimental Setup Summary

This section described the experimental setup and parameters for both focus areas of the dissertation with regard to both chosen datasets. The deep learning models and genetic algorithms are both grouped into slates, where model letters within slates look at particular network and separation scheme combinations. Slates had different hyperparameter sets on the high-level models. For genetic algorithms, model letters within slates looked at particular tasks and/or combinations of data to use, while slate sets had different hyperparameters. This section also discussed some of the underlying analysis assumptions and some evaluation techniques (including the model comparison and statistical significance analysis) to be used discussing experimental results. The next section gives the in-depth results of the experiments described here.

6 Results and Discussion

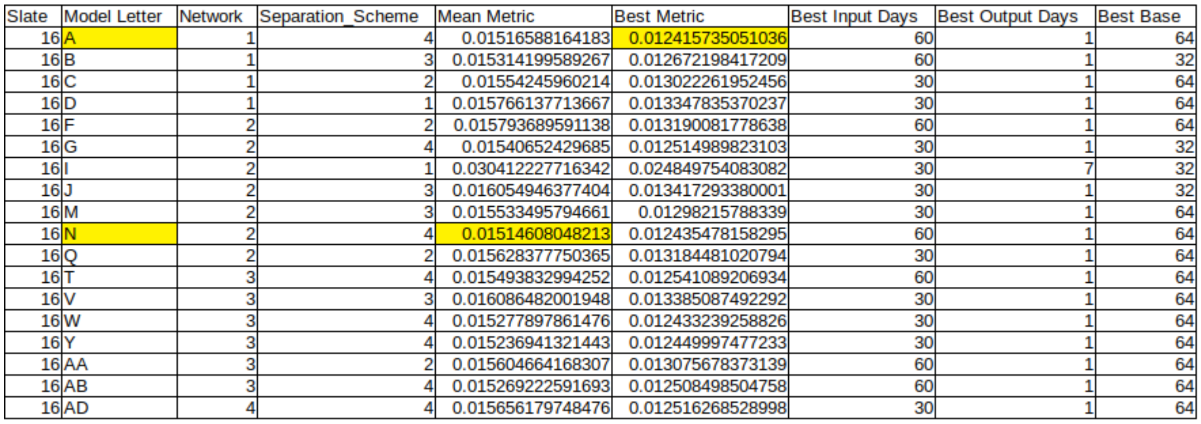
This section of the dissertation presents the results of the experiments described in the previous section, broken up by focus area (deep learning networks vs. QARM GA models) and by dataset (Jornada Basin and VitalDB). This section particularly looks at the experimental results and model performance with regard to the research questions posed in the dissertation.

6.1 Jornada Basin Dataset - Deep Learning Models

Slate Model Performances

For the Jornada Basin Dataset, the performances per-slate will first be discussed, followed by the relevance of the experiments to the research questions.

Slate 16 Slate 16 was a regression scheme (every datastream was predicted for every site) that used the normal shallow LSTM or normal AE for all model letters. Figure 33 looks at the performance for each LSTM model letter in slate 16. For models that predict datastreams or locations separately, the MSE presented is based on the weighted models for all datastreams and locations together.

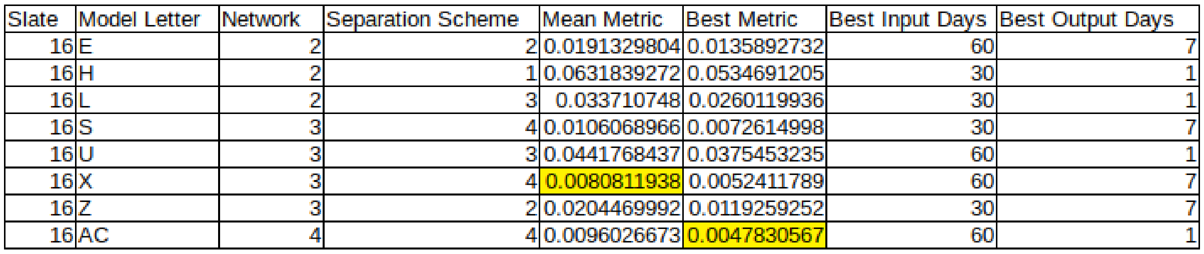


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
16	A	1	4	0.01516588164183	0.012415735051036	60	1	64
16	B	1	3	0.015314199589267	0.012672198417209	60	1	32
16	C	1	2	0.01554245960214	0.013022261952456	30	1	64
16	D	1	1	0.015766137713667	0.013347835370237	30	1	64
16	F	2	2	0.015793689591138	0.013190081778638	60	1	64
16	G	2	4	0.01540652429685	0.012514989823103	30	1	32
16	I	2	1	0.030412227716342	0.024849754083082	30	7	32
16	J	2	3	0.016054946377404	0.013417293380001	30	1	32
16	M	2	3	0.015533495794661	0.01298215788339	30	1	64
16	N	2	4	0.01514608048213	0.012435478158295	60	1	64
16	Q	2	2	0.015628377750365	0.013184481020794	30	1	64
16	T	3	4	0.015493832994252	0.012541089206934	60	1	64
16	V	3	3	0.016086482001948	0.013385087492292	30	1	64
16	W	3	4	0.015277897861476	0.012433239258826	30	1	64
16	Y	3	4	0.015236941321443	0.012449997477233	30	1	64
16	AA	3	2	0.015604664168307	0.013075678373139	60	1	64
16	AB	3	4	0.015269222591693	0.012508498504758	60	1	64
16	AD	4	4	0.015656179748476	0.012516268528998	30	1	64

Figure 33: Slate 16 LSTM models and their performance.

The best mean MSE metric across the variety of hyperparameters was found for model letter N, a Network 2 letter which used the autoencoder model L. L processed all datastreams for one location at a time. N took that latent space output for each location and fused it to come up with predictions for all datastreams and all locations. The model letter which contained the overall best performing model was model letter A, which was a Network 1 (monolithic) model. The best model for A used a base 64 neural network.

The AE performance for the slate is displayed in Figure 34. The minimum mean AE model was X, a Network 2 model while previous input from model L, while the lowest overall MSE was found in model AC, a Network 4 model which used autoencoders with all datastreams and location together. Models X and AC had considerably lower MSE compared to the other AE letters.

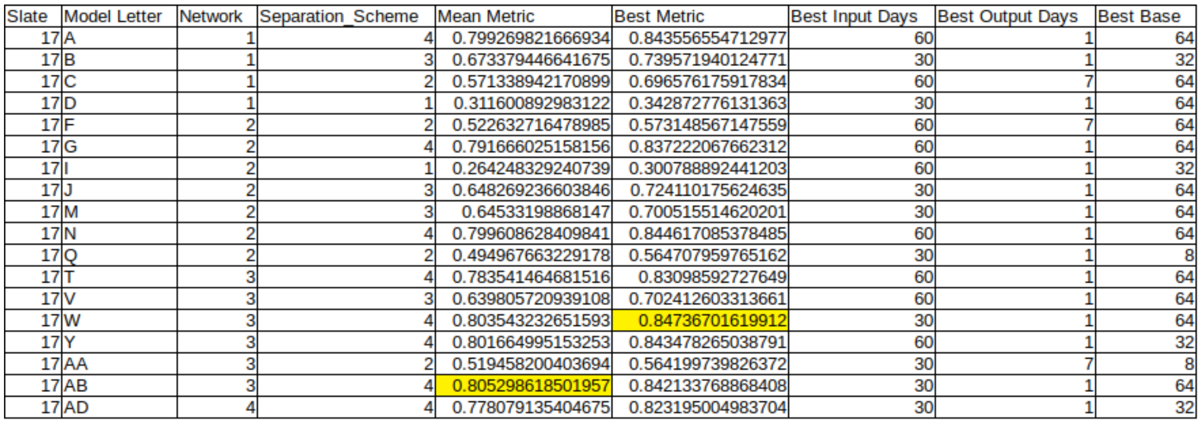


Slate	Model Letter	Network	Separation Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days
16	E	2	2	0.0191329804	0.0135892732	60	7
16	H	2	1	0.0631839272	0.0534691205	30	1
16	L	2	3	0.033710748	0.0260119936	30	1
16	S	3	4	0.0106068966	0.0072614998	30	7
16	U	3	3	0.0441768437	0.0375453235	60	1
16	X	3	4	0.0080811938	0.0052411789	60	7
16	Z	3	2	0.0204469992	0.0119259252	30	7
16	AC	4	4	0.0096026673	0.0047830567	60	1

Figure 34: Slate 16 autoencoder models and their performance.

Slate 17 Slate 17 was a prediction scheme that reused the same autoencoders from Slate 16. The normal shallow LSTM models were used, but this time for prediction tasks. Frost was predicted for all LSTM letters, and the combination models used the same weighting scheme depending on datastream or location as the regression task, but with regard to the F1 statistic rather than the MSE. Figure 35 displays the parameters and best and mean model performance for the F1 statistic for slate 17.

The model with the best mean F1 statistic was model AB, which is a Network 3 model which gets input from AE Z. Z is Network 3 AE which preprocesses input for all locations on one datastream after getting input from AE H. The model with the best overall F1 statistic was model W, which used the AE U as input. Model letter U is a Network 3 AE that processes input for all datastreams on one location after fusing input from AE H, which outputs one datastream per location.



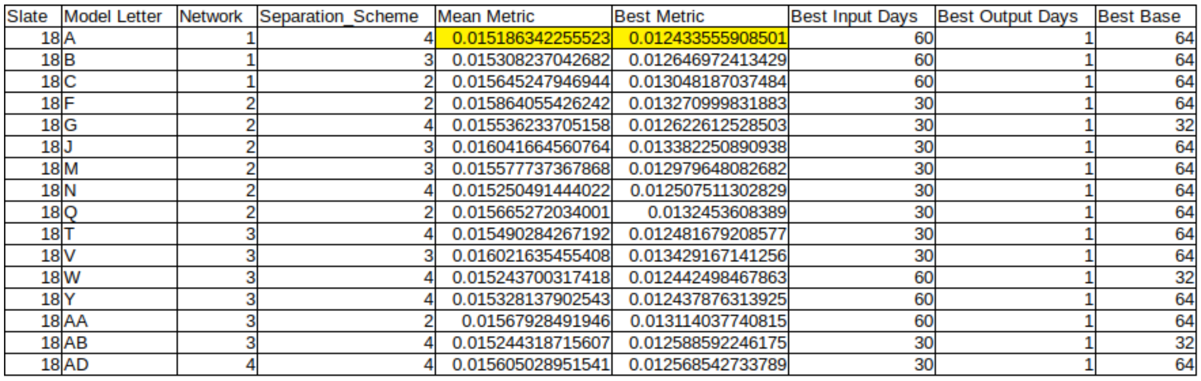
Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
17	A	1	4	0.799269821666934	0.843556554712977	60	1	64
17	B	1	3	0.673379446641675	0.739571940124771	30	1	32
17	C	1	2	0.571338942170899	0.696576175917834	60	7	64
17	D	1	1	0.311600892983122	0.342872776131363	30	1	64
17	F	2	2	0.522632716478985	0.573148567147559	60	7	64
17	G	2	4	0.791666025158156	0.837222067662312	60	1	64
17	I	2	1	0.264248329240739	0.300788892441203	60	1	32
17	J	2	3	0.648269236603846	0.724110175624635	30	1	64
17	M	2	3	0.64533198868147	0.700515514620201	30	1	64
17	N	2	4	0.799608628409841	0.844617085378485	60	1	64
17	Q	2	2	0.494967663229178	0.564707959765162	30	1	8
17	T	3	4	0.783541464681516	0.83098592727649	60	1	64
17	V	3	3	0.639805720939108	0.702412603313661	60	1	64
17	W	3	4	0.803543232651593	0.84736701619912	30	1	64
17	Y	3	4	0.801664995153253	0.843478265038791	60	1	32
17	AA	3	2	0.519458200403694	0.564199739826372	30	7	8
17	AB	3	4	0.805298618501957	0.842133768868408	30	1	64
17	AD	4	4	0.778079135404675	0.823195004983704	30	1	32

Figure 35: Slate 17 LSTM models and their performance.

Slates 16 and 17 preliminary observations From Slates 16 and 17, it was evident that the separation scheme which maintained a separate model for each datastream and location (model letters D and I) was not performant compared to all other separation schemes. This was unsurprising, as these models do not have opportunity to learn interactions from other features and/or locations. For subsequent slates, model letters D and I were dropped. However, the AE model letter H was kept (despite processing datastreams and locations separately) to account for the possibility that other model processing on top of H's outputs might result in useful models.

Slate 18 Slate 18 (a regression model) used the same parameters as slate 16, with the exception of the depth of the LSTM network structure depth. Slate 18 used deep LSTMs for all of the base 8, 32, and 64 model variants. Slate 18 re-used the autoencoders from Slate 16, since LSTM model depth was the only network structure change. Figure 36 displays the model letter results from Slate 18.

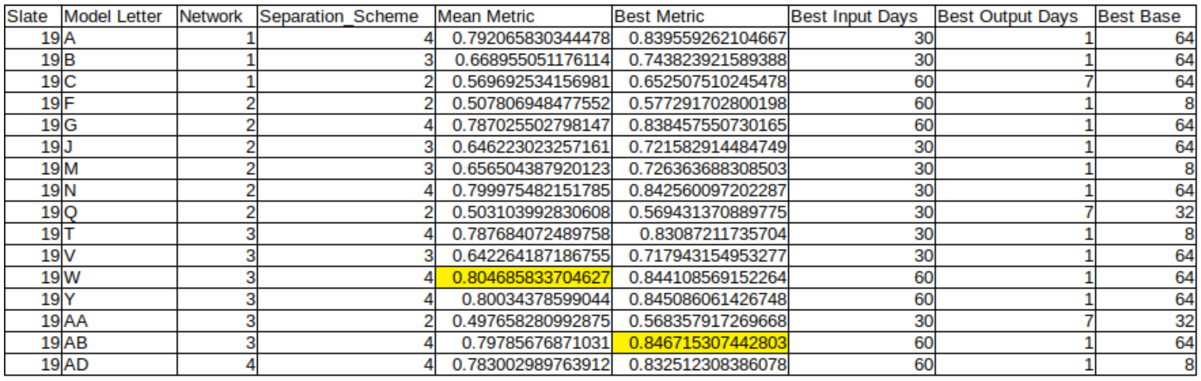
For Slate 18, model letter monolithic model letter A had both the overall minimum MSE as well as the lowest average MSE. This is not entirely surprising, as depth is generally believed to be related to the model capacity, and more depth presumably allows the model more ability to understand the interactions between different features. Figure 37 displays the results from Slate 19.



Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
18	A	1	4	0.015186342255523	0.012433555908501	60	1	64
18	B	1	3	0.015308237042682	0.012646972413429	60	1	64
18	C	1	2	0.015645247946944	0.013048187037484	60	1	64
18	F	2	2	0.015864055426242	0.013270999831883	30	1	64
18	G	2	4	0.015536233705158	0.012622612528503	30	1	32
18	J	2	3	0.016041664560764	0.013382250890938	30	1	64
18	M	2	3	0.015577737367868	0.012979648082682	30	1	64
18	N	2	4	0.015250491444022	0.012507511302829	30	1	64
18	Q	2	2	0.015665272034001	0.0132453608389	30	1	64
18	T	3	4	0.015490284267192	0.012481679208577	30	1	64
18	V	3	3	0.016021635455408	0.013429167141256	30	1	64
18	W	3	4	0.015243700317418	0.012442498467863	60	1	32
18	Y	3	4	0.015328137902543	0.012437876313925	60	1	64
18	AA	3	2	0.01567928491946	0.013114037740815	60	1	64
18	AB	3	4	0.015244318715607	0.012588592246175	30	1	32
18	AD	4	4	0.015605028951541	0.012568542733789	30	1	64

Figure 36: Slate 18 LSTM models and their performance.

Slate 19 Slate 19 (a prediction model) was similar to Slate 17 in terms of parameters, except for using deep LSTM network structures as in Slate 18.

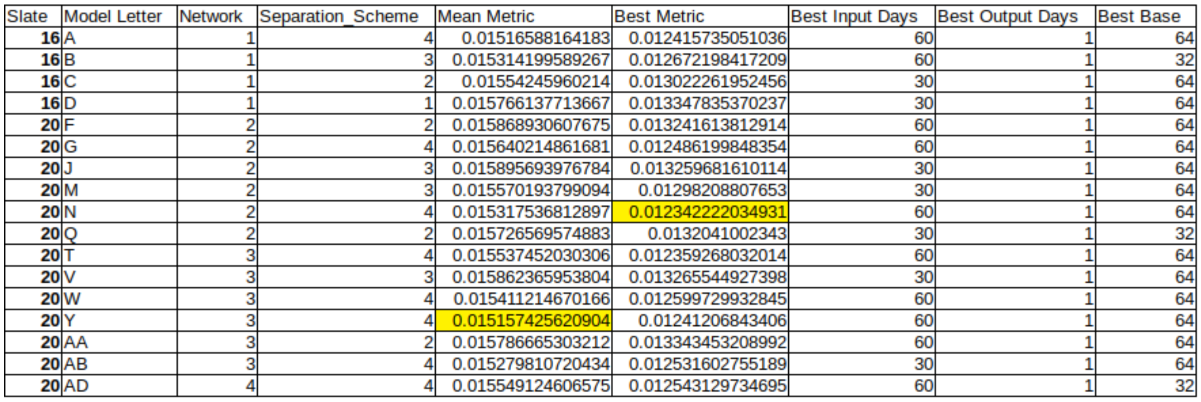


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
19	A	1	4	0.792065830344478	0.839559262104667	30	1	64
19	B	1	3	0.668955051176114	0.743823921589388	30	1	64
19	C	1	2	0.569692534156981	0.652507510245478	60	7	64
19	F	2	2	0.507806948477552	0.577291702800198	60	1	8
19	G	2	4	0.787025502798147	0.838457550730165	60	1	64
19	J	2	3	0.646223023257161	0.721582914484749	30	1	64
19	M	2	3	0.656504387920123	0.726363688308503	30	1	8
19	N	2	4	0.799975482151785	0.842560097202287	30	1	64
19	Q	2	2	0.503103992830608	0.569431370889775	30	7	32
19	T	3	4	0.787684072489758	0.83087211735704	30	1	8
19	V	3	3	0.642264187186755	0.717943154953277	30	1	64
19	W	3	4	0.804685833704627	0.844108569152264	60	1	64
19	Y	3	4	0.80034378599044	0.845086061426748	60	1	64
19	AA	3	2	0.497658280992875	0.568357917269668	30	7	32
19	AB	3	4	0.79785676871031	0.846715307442803	60	1	64
19	AD	4	4	0.783002989763912	0.832512308386078	60	1	8

Figure 37: Slate 19 LSTM models and their performance.

For slate 19, model letter W had the highest mean F1 statistic, while model letter AB had the overall highest F1 - model letters that flipped ”best” positions from Slate 17. AB and W are both network 3 models but took slightly different processing paths as noted in the paragraph in Slate 17.

Slate 20 Slate 20 trained regression models similar to Slate 16, but with deep AE preprocessing models rather than the normal shallow ones. Since model letters A-D do not use any type of AE model, they were left out of this slate. For comparison, the results for A-D for slate 16 are provided in Figure 38.

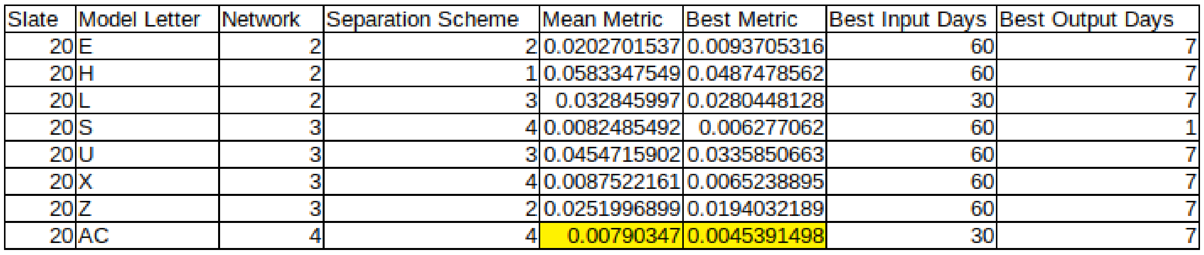


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
16	A	1	4	0.01516588164183	0.012415735051036	60	1	64
16	B	1	3	0.015314199589267	0.012672198417209	60	1	32
16	C	1	2	0.01554245960214	0.013022261952456	30	1	64
16	D	1	1	0.015766137713667	0.013347835370237	30	1	64
20	F	2	2	0.015868930607675	0.013241613812914	60	1	64
20	G	2	4	0.015640214861681	0.012486199848354	60	1	64
20	J	2	3	0.015895693976784	0.013259681610114	30	1	64
20	M	2	3	0.015570193799094	0.01298208807653	30	1	64
20	N	2	4	0.015317536812897	0.012342222034931	60	1	64
20	Q	2	2	0.015726569574883	0.0132041002343	30	1	32
20	T	3	4	0.015537452030306	0.012359268032014	60	1	64
20	V	3	3	0.015862365953804	0.013265544927398	30	1	64
20	W	3	4	0.015411214670166	0.012599729932845	60	1	64
20	Y	3	4	0.015157425620904	0.01241206843406	60	1	64
20	AA	3	2	0.015786665303212	0.013343453208992	60	1	64
20	AB	3	4	0.015279810720434	0.012531602755189	30	1	64
20	AD	4	4	0.015549124606575	0.012543129734695	60	1	32

Figure 38: Slate 20 LSTM models and their performance.

Model letter Y boasted the best overall mean metric. Y is a network 3 model which takes input from AE X, which in turn gets input from model letter L. Model N had the overall lowest MSE, with input from AE L.

Since slate 20 used deep AE models, it was naturally necessary to retrain them, which took slightly longer than before due to the added depth. Figure 39 gives the performance of the trained AEs for Slate 20.

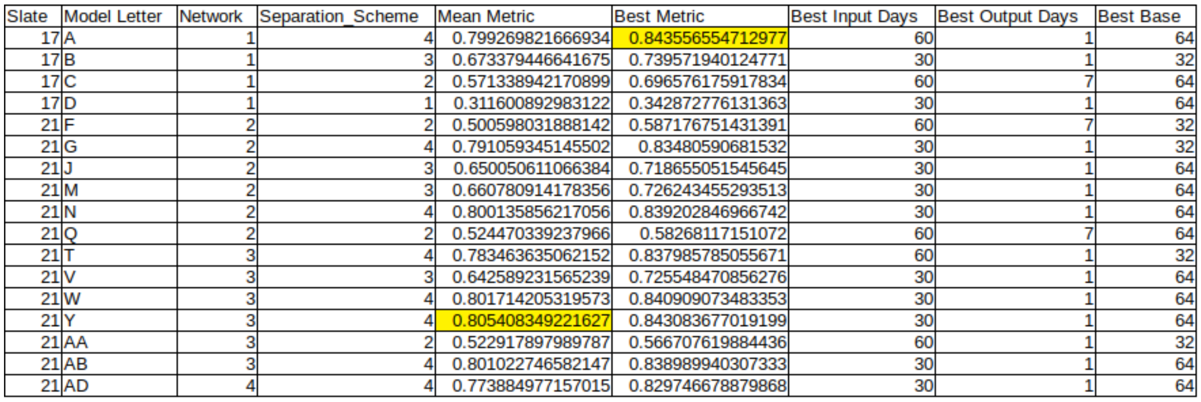


Slate	Model Letter	Network	Separation Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days
20	E	2	2	0.0202701537	0.0093705316	60	7
20	H	2	1	0.0583347549	0.0487478562	60	7
20	L	2	3	0.032845997	0.0280448128	30	7
20	S	3	4	0.0082485492	0.006277062	60	1
20	U	3	3	0.0454715902	0.0335850663	60	7
20	X	3	4	0.0087522161	0.0065238895	60	7
20	Z	3	2	0.0251996899	0.0194032189	60	7
20	AC	4	4	0.00790347	0.0045391498	30	7

Figure 39: Slate 20 autoencoder models and their performance.

Model letter AC had the overall lowest MSE as well as the lowest mean MSE. However, it can be seen that model letters S and X had similarly good performance compared to the over model letters.

Slate 21 Slate 21 was a prediction slate with the same parameters as slate 17, except for the use of the deep AE models from slate 20. As with slate 20, models A-D were not run for this slate due to their lack of AE, but were provided for reference in Figure 40.

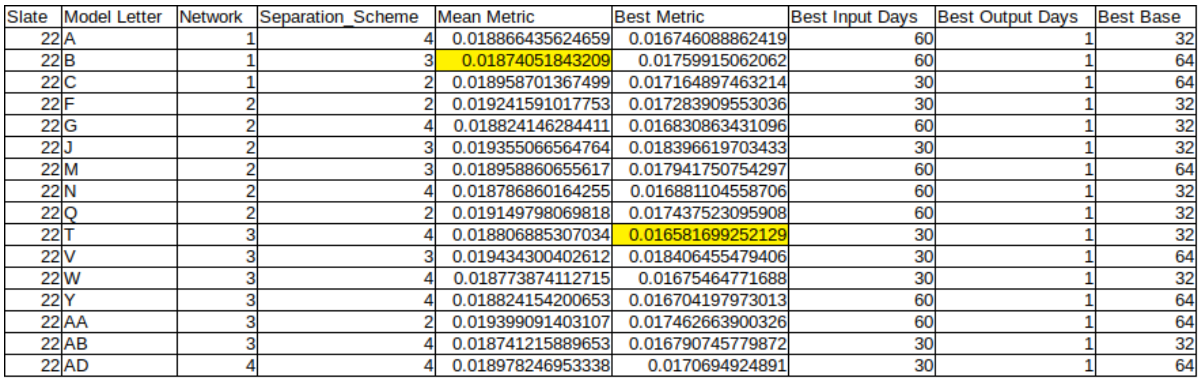


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
17	A	1	4	0.799269821666934	0.843556554712977	60	1	64
17	B	1	3	0.673379446641675	0.739571940124771	30	1	32
17	C	1	2	0.571338942170899	0.696576175917834	60	7	64
17	D	1	1	0.311600892983122	0.342872776131363	30	1	64
21	F	2	2	0.500598031888142	0.587176751431391	60	7	32
21	G	2	4	0.791059345145502	0.83480590681532	30	1	32
21	J	2	3	0.650050611066384	0.718655051545645	30	1	64
21	M	2	3	0.660780914178356	0.726243455293513	30	1	64
21	N	2	4	0.800135856217056	0.839202846966742	30	1	64
21	Q	2	2	0.524470339237966	0.58268117151072	60	7	64
21	T	3	4	0.783463635062152	0.837985785055671	60	1	32
21	V	3	3	0.642589231565239	0.725548470856276	30	1	64
21	W	3	4	0.801714205319573	0.840909073483353	30	1	64
21	Y	3	4	0.805408349221627	0.843083677019199	30	1	64
21	AA	3	2	0.522917897989787	0.566707619884436	60	1	32
21	AB	3	4	0.801022746582147	0.838989940307333	30	1	64
21	AD	4	4	0.773884977157015	0.829746678879868	30	1	64

Figure 40: Slate 21 LSTM models and their performance.

For Slate 21, model letter Y had the best overall mean performance, while model letter A had the overall highest F1 statistic. As mentioned previously, Y takes input from AE X, which takes input from AE L.

Slate 22 Slate 22 was a regression slate with a shallow AE and LSTM network. However, Slate 22 was geared to provide a starting point for examining the impact of adding a datastream. Slate 22 therefore used all of the same model letters as slate 16, but with the temperature/humidity datastream missing. Slate 22 therefore had only 3 datastreams as opposed to 4. Because of the missing datastream, it was also necessary to train new AEs for the starting point of Slate 22. Figure 41 displays the results from this slate.



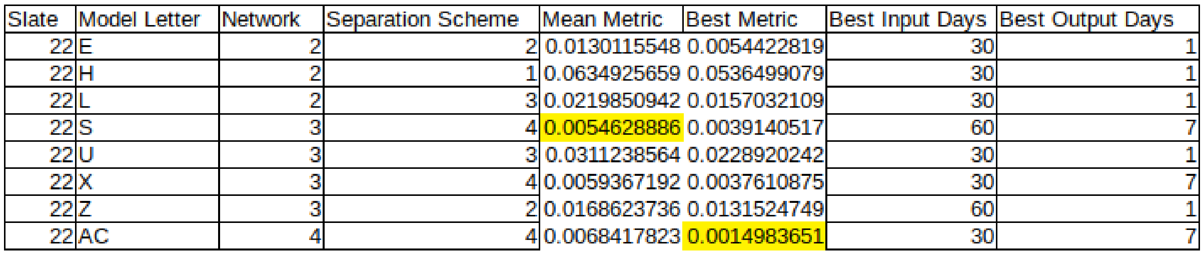
Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
22	A	1	4	0.018866435624659	0.016746088862419	60	1	32
22	B	1	3	0.01874051843209	0.01759915062062	60	1	64
22	C	1	2	0.018958701367499	0.017164897463214	30	1	64
22	F	2	2	0.019241591017753	0.017283909553036	30	1	32
22	G	2	4	0.018824146284411	0.016830863431096	60	1	32
22	J	2	3	0.019355066564764	0.018396619703433	30	1	32
22	M	2	3	0.018958860655617	0.017941750754297	60	1	64
22	N	2	4	0.018786860164255	0.016881104558706	60	1	32
22	Q	2	2	0.019149798069818	0.017437523095908	60	1	32
22	T	3	4	0.018806885307034	0.016581699252129	30	1	32
22	V	3	3	0.019434300402612	0.018406455479406	30	1	64
22	W	3	4	0.018773874112715	0.01675464771688	30	1	32
22	Y	3	4	0.018824154200653	0.016704197973013	60	1	64
22	AA	3	2	0.019399091403107	0.017462663900326	60	1	64
22	AB	3	4	0.018741215889653	0.016790745779872	30	1	32
22	AD	4	4	0.018978246953338	0.0170694924891	30	1	64

Figure 41: Slate 22 LSTM models and their performance.

The overall lowest mean from this slate came from model letter B, a monolithic

Network 1 letter that uses site-specific models (15 total). Model letter T had the best overall MSE for a model. Model T is a Network 3 model which uses AE S, which uses AE E. AE E is a datastream-specific model.

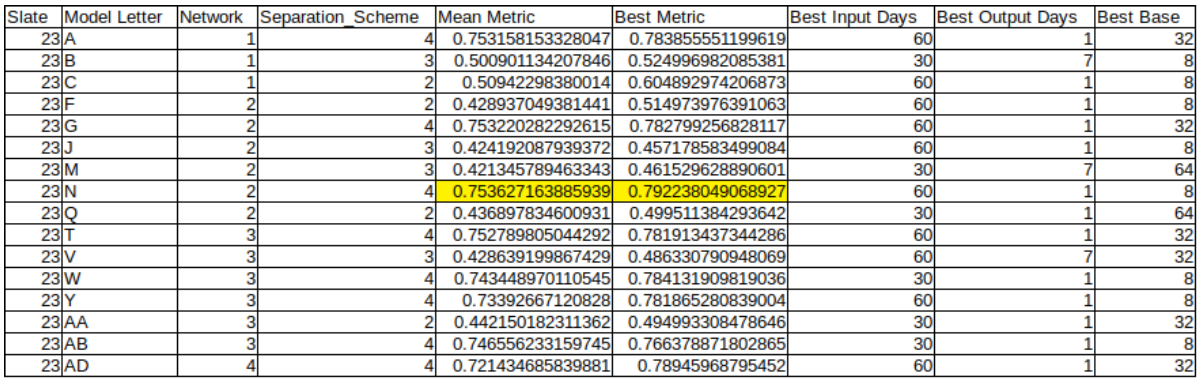
Slate 22's AE model letters were retrained to accommodate one less datastream. Figure 42 displays the performance of these AEs. The overall best mean MSE was found in AE model S, while the overall lowest was noted in model letter AC. S, X, and AC had significantly lower MSE compared to the other models.



Slate	Model Letter	Network	Separation Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days
22	E	2	2	0.0130115548	0.0054422819	30	1
22	H	2	1	0.0634925659	0.0536499079	30	1
22	L	2	3	0.0219850942	0.0157032109	30	1
22	S	3	4	0.0054628886	0.0039140517	60	7
22	U	3	3	0.0311238564	0.0228920242	30	1
22	X	3	4	0.0059367192	0.0037610875	30	7
22	Z	3	2	0.0168623736	0.0131524749	60	1
22	AC	4	4	0.0068417823	0.0014983651	30	7

Figure 42: Slate 22 autoencoder models and their performance.

Slate 23 Slate 23 was similar to Slate 22 in that it excluded a datastream but Slate 23 used the AE trained in Slate 22 and ran the frost prediction task. Like Slate 22, Slate 23 used shallow LSTM network structures. Figure 43 displays the results from Slate 23.



Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
23	A	1	4	0.753158153328047	0.783855551199619	60	1	32
23	B	1	3	0.500901134207846	0.524996982085381	30	7	8
23	C	1	2	0.50942298380014	0.604892974206873	60	1	8
23	F	2	2	0.428937049381441	0.514973976391063	60	1	8
23	G	2	4	0.753220282292615	0.782799256828117	60	1	32
23	J	2	3	0.424192087939372	0.457178583499084	60	1	8
23	M	2	3	0.421345789463343	0.461529628890601	30	7	64
23	N	2	4	0.753627163885939	0.792238049068927	60	1	8
23	Q	2	2	0.436897834600931	0.499511384293642	30	1	64
23	T	3	4	0.752789805044292	0.781913437344286	60	1	32
23	V	3	3	0.428639199867429	0.486330790948069	60	7	32
23	W	3	4	0.743448970110545	0.784131909819036	30	1	8
23	Y	3	4	0.73392667120828	0.781865280839004	60	1	8
23	AA	3	2	0.442150182311362	0.494993308478646	30	1	32
23	AB	3	4	0.746556233159745	0.766378871802865	30	1	8
23	AD	4	4	0.721434685839881	0.78945968795452	60	1	32

Figure 43: Slate 23 LSTM models and their performance.

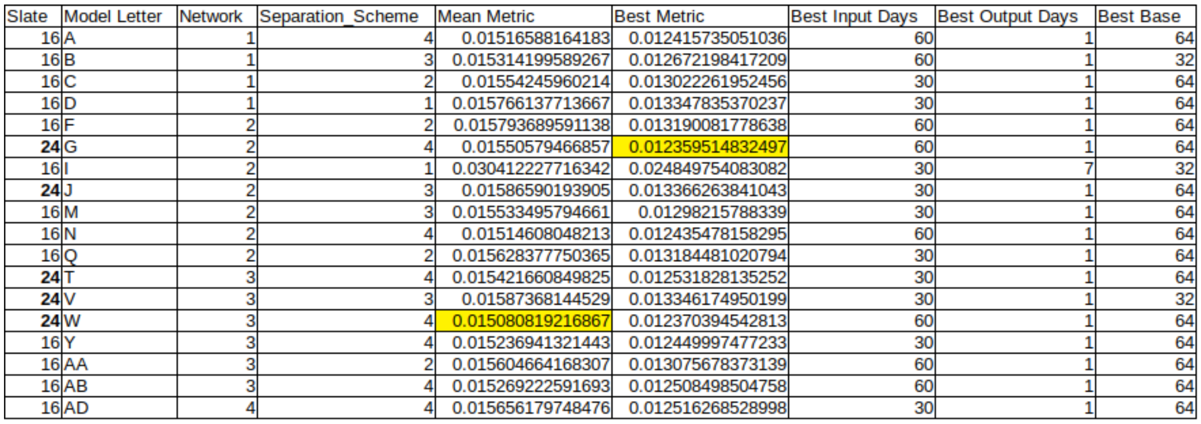
The highest mean F1 statistic as well as the overall highest F1 statistic was found in model letter N, a Network 2 model which uses AE model letter L.

Note on Slates 22 and 23 It is plain from the data that Slates 22 and 23 did not perform quite as well as the slates the came before them. This is unsurprising given that the exclusion the temp/humidity datastream would eliminate very useful information in a weather system, particularly for the prediction task. The presence or absence of frost was derived from one of the features in the temp/humidity datastream, so it expected to see lower performance. For a general prediction task, it would not necessarily be obvious which data streams are most related to performance.

Slate 24 Slate 24 is meant to mimic Slate 16 exactly in terms of total output but uses Slate 22 as a starting point. Since the temp/humidity datastream is added back in, Slate 24 retrains autoencoders E, H, S, and U, which separate by datastream or use input from another AE which does. For E and H, which separate by datastream on an individual letter, only the AE which processes the new datastream must be trained, which takes down the overall training time for these models considerably (they are able to use input from Slate 22 on the previously trained datastreams for these model letters). Since many model letters require end-to-end training, many of the models would exactly mimic Slate 16's process. These models were also not retrained, but their performance in Slate 16 is provided for comparison. Ultimately, model letters E (in part), G, H (in part), J, S, T, U, V, and W were retrained for Slate 24. Figure 44 gives the performance for Slate 24 models (bolded in the Figure) compared in conjunction with the analogous Slate 16 models.

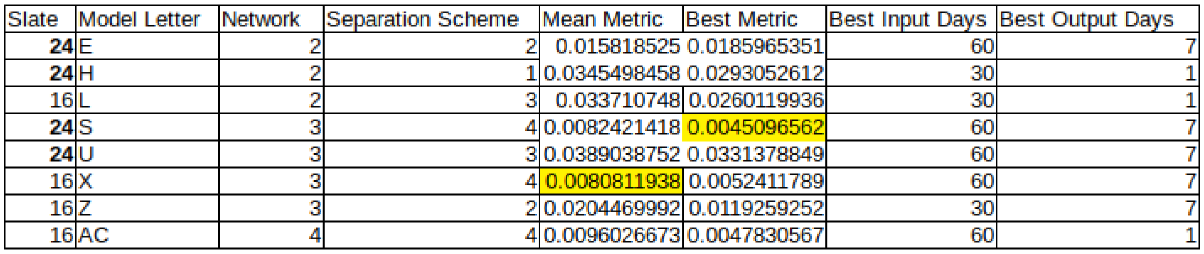
For the simulated full retrain, model letter W had the best overall mean performance, while model letter G had the best minimum performance. As previously mentioned, model W uses input from AEs U and H, while model letter G uses input from AE letter E.

Figure 45 displays the AE performance for Slate 24 as well as the comparison models from slate 16 for the full simulated retrain case. It should be noted that for E and H, it is giving only the performance for the retrained models, which doesn't completely encapsulate every AE model used for these letters, as slate 23 models were also incorporated.



Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
16	A	1	4	0.01516588164183	0.012415735051036	60	1	64
16	B	1	3	0.015314199589267	0.012672198417209	60	1	32
16	C	1	2	0.01554245960214	0.013022261952456	30	1	64
16	D	1	1	0.015766137713667	0.013347835370237	30	1	64
16	F	2	2	0.015793689591138	0.013190081778638	60	1	64
24	G	2	4	0.01550579466857	0.012359514832497	60	1	64
16	I	2	1	0.030412227716342	0.024849754083082	30	7	32
24	J	2	3	0.01586590193905	0.013366263841043	30	1	64
16	M	2	3	0.015533495794661	0.01298215788339	30	1	64
16	N	2	4	0.01514608048213	0.012435478158295	60	1	64
16	Q	2	2	0.015628377750365	0.013184481020794	30	1	64
24	T	3	4	0.015421660849825	0.012531828135252	30	1	64
24	V	3	3	0.01587368144529	0.013346174950199	30	1	32
24	W	3	4	0.015080819216867	0.012370394542813	60	1	64
16	Y	3	4	0.015236941321443	0.012449997477233	30	1	64
16	AA	3	2	0.015604664168307	0.013075678373139	60	1	64
16	AB	3	4	0.015269222591693	0.012508498504758	60	1	64
16	AD	4	4	0.015656179748476	0.012516268528998	30	1	64

Figure 44: Slate 24 LSTM models and their performance.

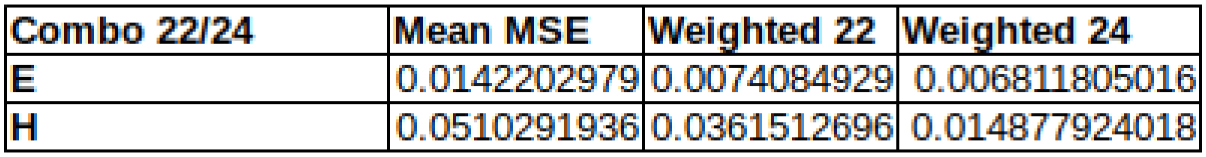


Slate	Model Letter	Network	Separation Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days
24	E	2	2	0.015818525	0.0185965351	60	7
24	H	2	1	0.0345498458	0.0293052612	30	1
16	L	2	3	0.033710748	0.0260119936	30	1
24	S	3	4	0.0082421418	0.0045096562	60	7
24	U	3	3	0.0389038752	0.0331378849	60	7
16	X	3	4	0.0080811938	0.0052411789	60	7
16	Z	3	2	0.0204469992	0.0119259252	30	7
16	AC	4	4	0.0096026673	0.0047830567	60	1

Figure 45: Slate 24 autoencoder models and their performance.

Model letter X from Slate 16 had the overall lowest mean MSE for this simulated combination, while model letter S from slate 24 had the overall lowest MSE. Model letter AC from Slate 16 had similar performance to S and X as well.

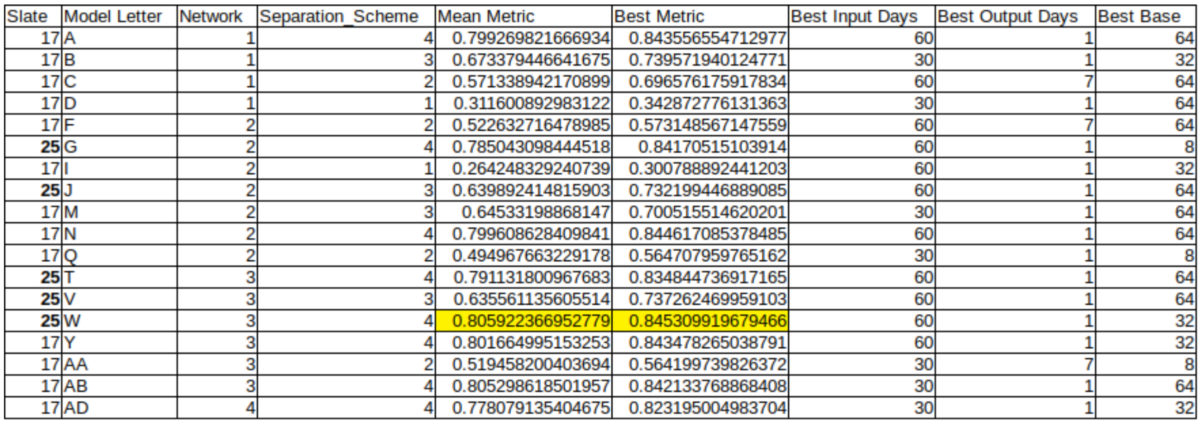
The mean performance of the AE model letter E and H used by the Slate 24 models upstream is presented in Figure 46.



Combo 22/24	Mean MSE	Weighted 22	Weighted 24
E	0.0142202979	0.0074084929	0.006811805016
H	0.0510291936	0.0361512696	0.014877924018

Figure 46: The weighted MSE for model letters E and H from Slates 22 and 24.

Slate 25 Slate 25 simulated adding the temp/humidity datastream, similar to Slate 24, but for the frost prediction task. When applicable, Slate 25 reused the AEs from Slate 24, but for end-to-end models that would need full retraining for the added datastream, Slate 17 models are provided for comparison. Figure 47 displays the results from this Slate in conjunction with the Slate 17 comparison models.

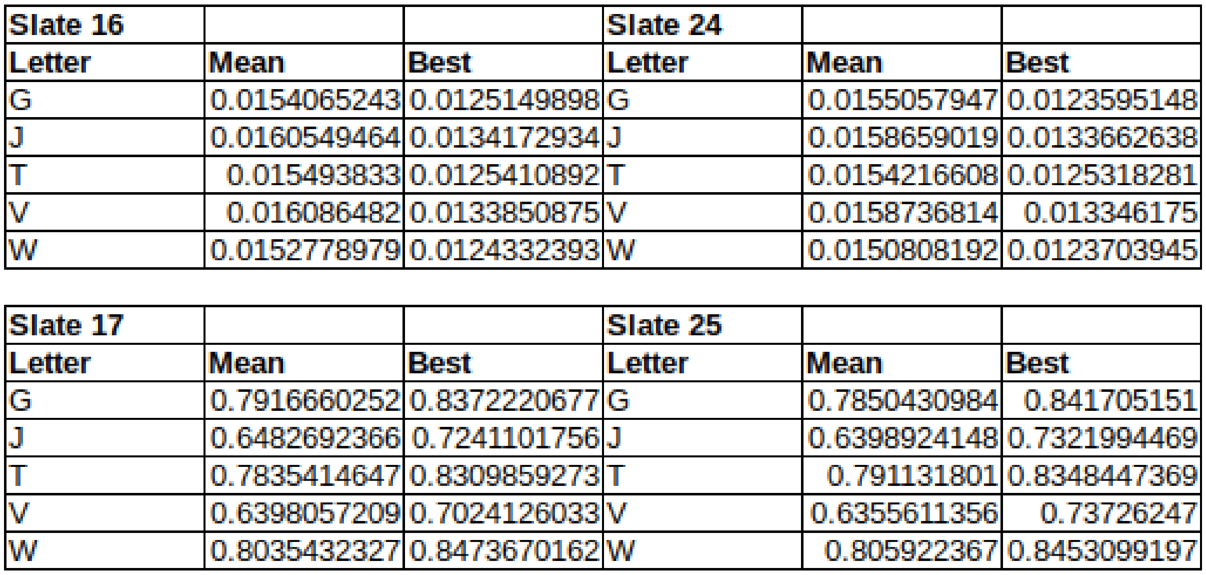


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
17	A	1	4	0.799269821666934	0.843556554712977	60	1	64
17	B	1	3	0.673379446641675	0.739571940124771	30	1	32
17	C	1	2	0.571338942170899	0.696576175917834	60	7	64
17	D	1	1	0.311600892983122	0.342872776131363	30	1	64
17	F	2	2	0.522632716478985	0.573148567147559	60	7	64
25	G	2	4	0.785043098444518	0.84170515103914	60	1	8
17	I	2	1	0.264248329240739	0.300788892441203	60	1	32
25	J	2	3	0.639892414815903	0.732199446889085	60	1	64
17	M	2	3	0.64533198868147	0.700515514620201	30	1	64
17	N	2	4	0.799608628409841	0.844617085378485	60	1	64
17	Q	2	2	0.494967663229178	0.564707959765162	30	1	8
25	T	3	4	0.791131800967683	0.834844736917165	60	1	64
25	V	3	3	0.635561135605514	0.737262469959103	60	1	64
25	W	3	4	0.805922366952779	0.845309919679466	60	1	32
17	Y	3	4	0.801664995153253	0.843478265038791	60	1	32
17	AA	3	2	0.519458200403694	0.564199739826372	30	7	8
17	AB	3	4	0.805298618501957	0.842133768868408	30	1	64
17	AD	4	4	0.778079135404675	0.823195004983704	30	1	32

Figure 47: Slate 25 LSTM models and their performance.

For this simulated retraining slate, model letter W had the best mean F1 statistic as well as the highest overall performance. As before, model letter W uses input from AE U which takes input from AE H.

Sanity Checking Slates 23 and 24 With only a few exceptions, most models would require complete end-to-end training when adding a new datastream. For this reason, Slates 23 and 24 retrained the models that had enough modularity to avoid complete end-to-end training and then compared performance to analogous models in Slates 16 and 17. However, as a sanity check, the LSTM model performances from the fully trained and the retrained models should be compared. Since they have exactly the same inputs and outputs and use models with exactly the same parameters (though perhaps trained at different times, in the case of 23 and 24), the performance would not be expected to differ. Figure 48 displays the mean and best metrics for Slates 16 and 24 and Slates 17 and 25 on the models letters in question.



Slate 16			**Slate 24**		
Letter	**Mean**	**Best**	**Letter**	**Mean**	**Best**
G	0.0154065243	0.0125149898	G	0.0155057947	0.0123595148
J	0.0160549464	0.0134172934	J	0.0158659019	0.0133662638
T	0.015493833	0.0125410892	T	0.0154216608	0.0125318281
V	0.016086482	0.0133850875	V	0.0158736814	0.013346175
W	0.0152778979	0.0124332393	W	0.0150808192	0.0123703945

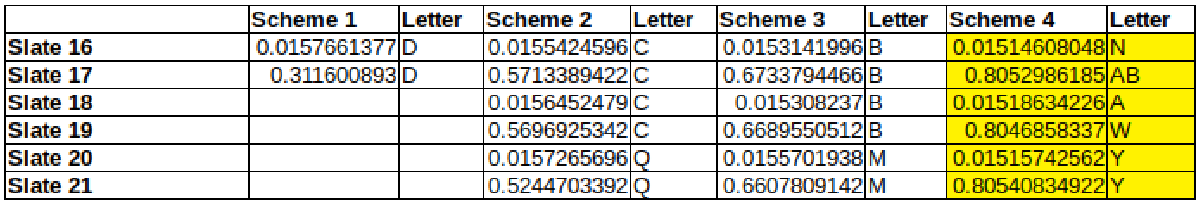
Slate 17			**Slate 25**		
Letter	**Mean**	**Best**	**Letter**	**Mean**	**Best**
G	0.7916660252	0.8372220677	G	0.7850430984	0.841705151
J	0.6482692366	0.7241101756	J	0.6398924148	0.7321994469
T	0.7835414647	0.8309859273	T	0.791131801	0.8348447369
V	0.6398057209	0.7024126033	V	0.6355611356	0.73726247
W	0.8035432327	0.8473670162	W	0.805922367	0.8453099197

Figure 48: Mean and Best Overall Model Metrics for Slates 16 and 24 and Slates 17 and 25.

As expected, there is little difference between Slates 16 and 24 and between Slates 17 and 25 in terms of model letter performance.

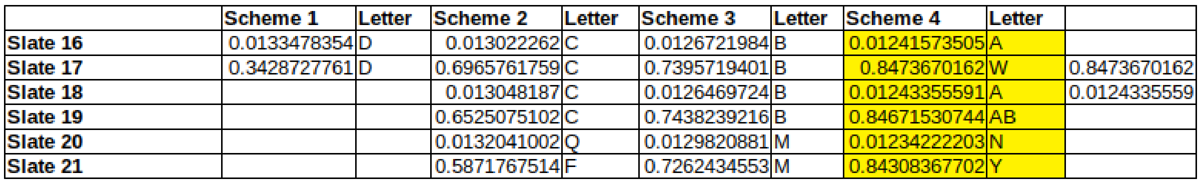
Jornada Combination Models and Separation Schemes

For the Jornada Basin dataset, there were several ways to separate out datastreams and locations, which were explored by the different separation schemes. For regression tasks, the ultimate goal was to predict every datastream value for every site, while for prediction tasks, the goal was to predict the presence or absence of frost for each site. However, the experiments run in this dissertation accomplished this both by using large combination models and by using combinations of smaller models which predicted subsets of the data. One of the investigations in this dissertation for this dataset was determining which of the separation schemes functioned best. Figure 49 displays the best mean of each separation scheme for each slate from 16-21 and Figure 50 displays the best overall minimum of each separation scheme for the same slates. Slates 22 and 23 were not included in these Figure since they were missing a datastream.



	Scheme 1	Letter	Scheme 2	Letter	Scheme 3	Letter	Scheme 4	Letter
Slate 16	0.0157661377	D	0.0155424596	C	0.0153141996	B	0.01514608048	N
Slate 17	0.311600893	D	0.5713389422	C	0.6733794466	B	0.8052986185	AB
Slate 18			0.0156452479	C	0.015308237	B	0.01518634226	A
Slate 19			0.5696925342	C	0.6689550512	B	0.8046858337	W
Slate 20			0.0157265696	Q	0.0155701938	M	0.01515742562	Y
Slate 21			0.5244703392	Q	0.6607809142	M	0.80540834922	Y

Figure 49: Best mean performance of separation schemes for Jornada dataset.



	Scheme 1	Letter	Scheme 2	Letter	Scheme 3	Letter	Scheme 4	Letter	
Slate 16	0.0133478354	D	0.013022262	C	0.0126721984	B	0.01241573505	A	
Slate 17	0.3428727761	D	0.6965761759	C	0.7395719401	B	0.8473670162	W	0.8473670162
Slate 18			0.013048187	C	0.0126469724	B	0.01243355591	A	0.0124335559
Slate 19			0.6525075102	C	0.7438239216	B	0.84671530744	AB	
Slate 20			0.0132041002	Q	0.0129820881	M	0.01234222203	N	
Slate 21			0.5871767514	F	0.7262434553	M	0.84308367702	Y	

Figure 50: Best overall performance of separation schemes for Jornada dataset.

It can be seen from these figures that predicting all datastreams for all sites together or predicting frost for all sites together performed better than any other separation scheme. Separation scheme 1, which used a separate model for each datastream and each location, performed sufficiently poorly to cease to run for Slates 18 on. In second place was separation scheme 3, which predicted datastreams or frost for all locations separately (in other words, models that took in site-specific data only). While all models predicted specific features for specific sites, allowing (in some scheme) the models to take in data from all sites was generally a net gain in performance.

Slate 22's mean model performance was the only exception. Without the temp/humidity datastream, this slate found site-specific model data (with model letter B) to have the lowest mean for the regression task. Slate 22 also had overall comparatively higher model error (unsurprising, given the missing datastream). Without this datastream, Slate 22 had better luck predicting only on a site-by-site basis than with a combination model. This did not hold for Slate 23, interestingly, for the frost prediction task with the similarly missing datastream.

While the combination models of separation scheme 4 in general fared much better than their separated counterparts, it should be noted that this does not eliminate the possibility of more modular data management, as many of the best performing models incorporated more modular autoencoder breakdowns to arrive at the final predictions. These model organizations are discussed in more detail in the following section.

Jornada Network Organizations

For these experiments, 4 network types were proposed, incorporating various degrees of modularity and autoencoder preprocessing. Some AEs and LSTMs combined separation schemes at various points in the process.

The results for the AE performance in Slates 16 and 20 (normal and deep AE) had significantly lower errors for model letters S, X, and AC. Model S and X are Network 3 models, whereas model AC is a Network 4 model. S, X, and AC were all separation scheme 4 models, which means their latent space represented features for all datastreams and all locations. However, model letter X takes input from AE L, which outputs site-specific latent space, where S takes input from AE E, which outputs datastream-specific latent space.

It is interesting that the Network 3 architecture output performant AEs for both the datastream and location separation schemes: in both cases, AEs X and S were able to learn representative features allowing them to recreate the input with less errors than the prior AEs. This was the same for both shallow and deep AEs, which does not lend itself to concluding that depth alone was responsible for the increased performance. The multileveling itself (sub-tasks in earlier AEs before the combination task) may be beneficial to the increased performance.

Autoencoder AC generally also had higher performance than the other models. AC took in all datastreams for all locations and output all datastreams for all locations. S, X, and AC demonstrate the usefulness of having more data features processed together in order for the AE to learn representative latent space features.

However, better AE performance does not necessarily translate into better LSTM model performance. Autoencoders can sometimes function as regularizers, which prevent noisy data from impacting model training. Higher capacity to correctly recreate data may translate into more noise reaching the final model.

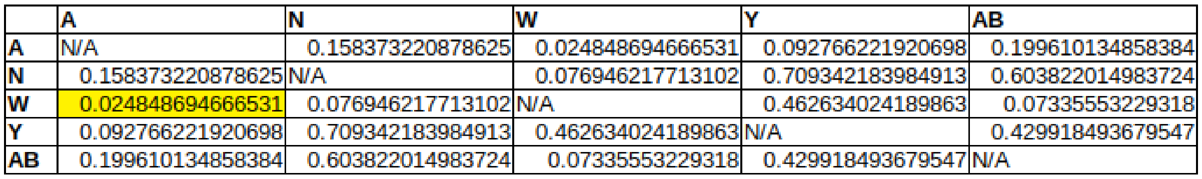
Model letters A, N, W, Y, and AB all found themselves in mean or overall best model for a Slate at least twice. Models G and T only scored once each (in either a retraining or missing datastream slate), while model AD never topped the charts. Model G is a Network 2 model which took input from AE E, which separated information by datastream (one datastream, all locations) which is synthesize for each datastream to arrive at the final output. Model letter T is a Network 3 model which took input from AE S, which had synthesized the info from AE E. Model letter AD only took input from AE AC.

Model letter A was a monolithic network which did not use any AE preprocessing. Model letter N took input from AE L (a per-location AE), while model letter W took input from model letter U, which output per-location data based on synthesized information taken from AE model H. Model letter Y took input from X, which synthesized per-location information from AE model L. Model letter AB took synthesized information from AE letter Z, which output per-datastream information synthesized from model letter H.

For Slates 16-21, the best mean regression model was found in Slate 16, with model letter N. The best mean prediction model was found in Slate 21, with model letter Y. The overall minimum regression model was found in Slate 20, model letter N, while the overall maximum prediction model was found in Slate 17 with model letter W.

The top performers overall were therefore either in the normal shallow models or in the deep AE models, and all top performers incorporated location-specific data at some point in the process. Model letter N was a better performer for regression tasks, while Y and W fared better for prediction tasks. Models N, Y, and W all incorporated some form of site-specific separation.

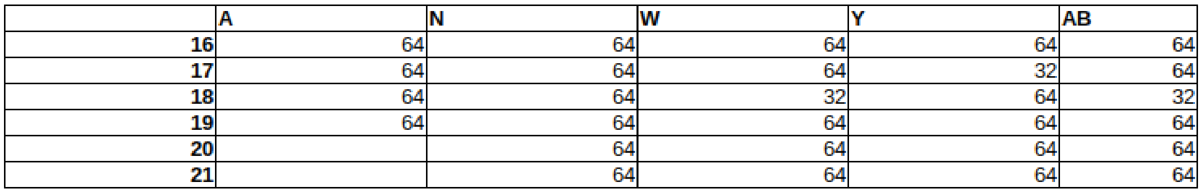
A Wilcoxon signed rank test was performed on the model distributions for A, N, W, Y, and AB for Slates 16,17,18,19,20,21, 22, and 23. (For slates 20 and 21, the distributions for A were repeated, since these slates looked at deep AEs and A does not incorporate an AE). Figure 51 demonstrates a table of p-values of the test between model letters. At the $p < 0.05$ level, only W demonstrated a statistically significant difference from A's distribution. None of the other autoencoder models had a statistically significant difference in distributions between themselves.



	A	N	W	Y	AB
A	N/A	0.158373220878625	0.024848694666531	0.092766221920698	0.199610134858384
N	0.158373220878625	N/A	0.076946217713102	0.709342183984913	0.603822014983724
W	0.024848694666531	0.076946217713102	N/A	0.462634024189863	0.07335553229318
Y	0.092766221920698	0.709342183984913	0.462634024189863	N/A	0.429918493679547
AB	0.199610134858384	0.603822014983724	0.07335553229318	0.429918493679547	N/A

Figure 51: P-values of Wilcoxon Signed Rank Test between distributions of model letters.

For this dissertation, network depth was another important factor in evaluating performance. The depth (base 8, 32, or 64) at which model letters A, N, W, Y, and AB achieved their best performance is presented in Figure 52.



	A	N	W	Y	AB
16	64	64	64	64	64
17	64	64	64	32	64
18	64	64	32	64	32
19	64	64	64	64	64
20		64	64	64	64
21		64	64	64	64

Figure 52: Base network where model letters achieved best performance for each Slate.

Figure 52 demonstrates that, with few exceptions, most models had their best performance with the deep network structure, base 64.

Though AE models had slightly better overall performance, it would then be tempting to conclude that from a depth standpoint, a small organization would be better off sticking with monolithic models than with AE models, as incorporating AEs is overall more models and thus more processing and development time. For the pure regression task, this mostly proved true. However, for the prediction task, an interesting pattern emerges for the high performing AE nodes with regard to mean performance for depth. Figure 53 displays the mean performance of model letters for the prediction task per base network structure.

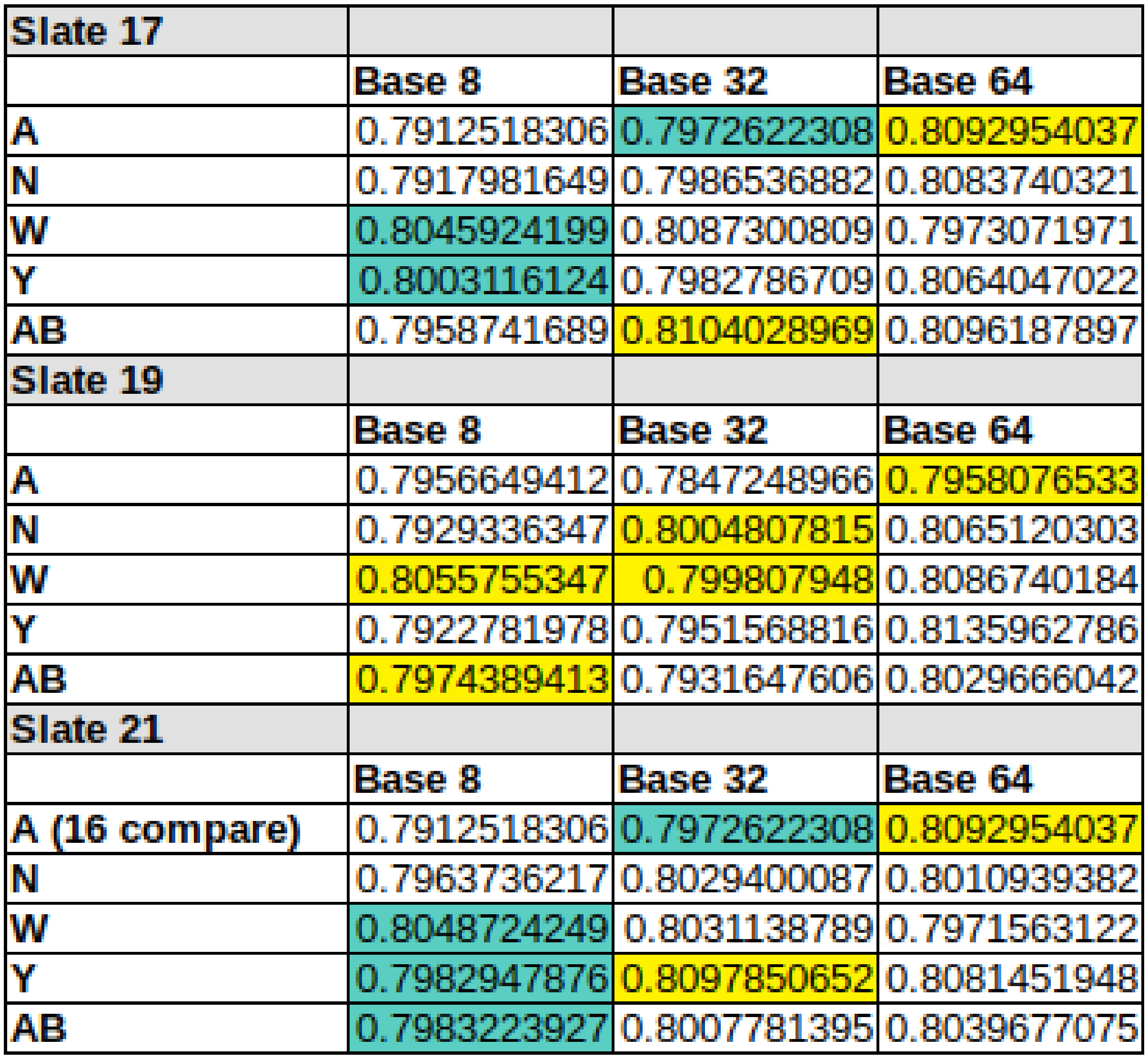
Figure 53 shows that models W and Y (and occasionally N and AB) often beat (or came very close to) monolithic letter A's performance **at a lower network depth**. Y and W are reusing the feature learning from training in the prior regression slate, allowing for shallower models to achieve the same or better performance on the subsequent prediction task.

For Slate 17, model letter W and Y did better than model letter A at 8 nodes than A did at 32 nodes. W and Y performed best at 8 nodes. For Slate 19, N, W, W, and AB all outperformed model letter A at 8 or 32 nodes from A's 64 nodes. For Slate 21, models W, Y, and AB at 8 nodes outperformed model letter A at 32 nodes, and model Y at 32 nodes also outperformed model letter A at 64 nodes.

Discussion on Jornada Basin Dataset Deep Learning Models with Regard to Research Questions

How modular can we make a multivariate system? While initially this dissertation assumed modularity would be most impactful in terms of types of data added and subtracted into the system, the experiments found generally better performance when separating data out by location of collection rather than datastream.

For AE training, both datastream or location separation schemes performed well, as AE performance was more affected by the ability to process all datastreams together at some point than by pure separation scheme. This was borne out by the fact the models S (separation by datastream before combination processing), X (separation by location



Slate 17			
	Base 8	Base 32	Base 64
A	0.7912518306	0.7972622308	0.8092954037
N	0.7917981649	0.7986536882	0.8083740321
W	0.8045924199	0.8087300809	0.7973071971
Y	0.8003116124	0.7982786709	0.8064047022
AB	0.7958741689	0.8104028969	0.8096187897
Slate 19			
	Base 8	Base 32	Base 64
A	0.7956649412	0.7847248966	0.7958076533
N	0.7929336347	0.8004807815	0.8065120303
W	0.8055755347	0.799807948	0.8086740184
Y	0.7922781978	0.7951568816	0.8135962786
AB	0.7974389413	0.7931647606	0.8029666042
Slate 21			
	Base 8	Base 32	Base 64
A (16 compare)	0.7912518306	0.7972622308	0.8092954037
N	0.7963736217	0.8029400087	0.8010939382
W	0.8048724249	0.8031138789	0.7971563122
Y	0.7982947876	0.8097850652	0.8081451948
AB	0.7983223927	0.8007781395	0.8039677075

Figure 53: Mean performance of model letter per base network structure on prediction task.

before combination processing) and AC (only combination processing) had significantly lower recreation error compared to the other models.

The performant AE models were N, W, Y, and AB. Of these, W, Y, and N all incorporate location specific processing at some point in the process. The path of the data for each model is outlined below:

N: All data for each location is processed separately by an AE model (L). The latent space output for each location is concatenated and provided as input to model N, an LSTM.

W: Each datastream for each location is processed separately by an AE model (H). The latent space model for each datastream is concatenated per location and each location's total datastream latent spaces are processed by an AE model (U). The latent space per location is provided as input to model letter W, an LSTM.

Y: All data for each location is processed separately by an AE model (L). The latent space output for each location is concatenated and provided as input to an AE model (X). The latent space AE model is provided as input to model Y, an LSTM.

AB: Each datastream for each location is processed separately by an AE model (H). The latent space output for each datastream (every location) is concatenated and provided as input to an AE model (Z). The latent space output per datastream is concatenated and provided as input to model AB, an LSTM.

In these experiments, modularity provided a boost to model performance, especially when it came to location-specific processing.

However, for these experiments, complete separation of datastreams and/or locations did not lead to high performing models. The best performing models necessitates having all data together at some point in the process. No other separation scheme consistently achieved as high results as separation scheme 4, which used one model to output all data.

What is the best method of incorporating new data into an existing neural network design? For Jornada Basin dataset, the best models generally used AE processing. For each of model letters N, W, Y, and AB, the retraining process would be different if adding a new datastream or a new location. The top level LSTM models would need to be retrained for all.

For model N, only the new location would need an additional AE in L to be trained. N would be fully retrained. For a new datastream at all locations, both the AE and LSTM would need to be fully retrained.

For model W, for new location or new datastream, only the corresponding new highly specific AEs in H would need to be retrained. AE model U and W would be fully retrained.

For model Y, only the new location would need to be trained for AE model L processing. AE X and LSTM Y would need to be fully retrained. For a new datastream, L, X, and Y would all need to be fully retrained.

As seen in the results from models 24 and 25, the retraining process does not hurt model performance. If using a monolithic network, the monolithic network alone would need to be retrained.

Can we delegate multivariate (multiple types of input data) decision tasks to exist on top of an existing feature set? All autoencoder models created latent space feature sets that were eventually used as input to LSTM models to perform tasks. In these experiments, the frost prediction tasks were trained on top of the latent feature output of these autoencoder models in comparison to the monolithic models. In general, the model trained on top of the AE feature sets outperformed the monolithic models. Additionally, the prediction tasks utilizing an existing feature set were often able to get better results at a lower network depth than the monolithic model.

Jornada Basin Dataset Deep Learning Final Thoughts

The Jornada Basin Dataset explored several types of AE and LSTM networks in many combinations of datastreams and locations to gain insight into structuring of highly dynamic data environments for farms or organizations that may not enjoy unlimited computing power or ability to collect and label data. There are advantages and disadvantages of different approaches when it comes to computing power, manageability, and expected rate of change for model inputs and outputs.

Model letter N is fairly easy to implement if a farm expects changes purely on a location basis, since it can add a per-location AE input to the feature set before retraining top-level tasks. Models W, Y, and AB require more AE retraining but potentially smaller top-level networks.

If a farmer or business owner needs a lot of high-level tasks for the same data inputs (for instance, one model per specific pest or disease in a highly susceptible environment), reusing an existing feature set could be very useful. Each high-level task can be potentially made of a smaller network that takes input from the feature latent space and trained more quickly. The existing feature sets also have the advantage that they were created with unsupervised learning and do not need labeled data to train the initial model. In this case, the feature set can potentially be created with much more data than exists in the high-level task. For example, if a farmer has 8 years of weather data but only 2 years of pest data, the AEs can be trained with 8 years and the pest set with 2.

However, the difficulty of maintenance of that feature set likely depends on the degree of variability of the data environment. If new datastreams **and** new locations are constantly being added, there will be potentially less retraining when working with highly specific AE models. All performant models in these experiments needed some form of combination processing at some point, which limits the usefulness of modularity to some degree.

Manageability is another aspect to consider in a data environment. If new locations are constantly added, it might be less work for a farmer to maintain site-specific models only (without input from other sites) at the cost of some accuracy.

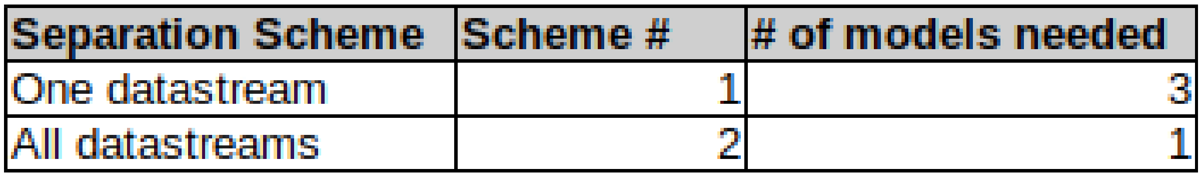
Jornada Basin Deep Learning Results Summary

In this set of experiments:

- Models that processed all datastreams together fared better than models that did not;
- However, the best faring models generally incorporated some version of site specific datastream processing;
- More modular networks had no statistically significant difference in distribution from monolithic models; and
- However, the site modular networks often obtained the same performance as a monolithic model at a lower depth.

6.2 VitalDB - Deep Learning Models

For the VitalDb dataset, it did not make sense to separate the data by "location" which in this case corresponded to the patient in question. Unlike the Jornada Basin dataset, the surgeries for the patients in the VitalDB dataset would have taken place at different times, often for different lengths of time. Because of this, there are only two separations schemes for this dataset: separated by datastream (each datastream predicts the metric separately) or not separated at all. There were three datastreams for the vital models, corresponding to the Orchestra infusion pump "orch", the SNUADC ECG "snu", and the Solar8000 ventilation monitor, "solar". For regression LSTMs, clinical information was as given as input but not predicted (as it is the same across all time sequences for a given case). For prediction tasks (and autoencoders), this information was not given as input, as it was too easy for the networks to memorize clinical information associated with a particular case. Figure 54 displays the separation schemes for the VitalDB dataset.



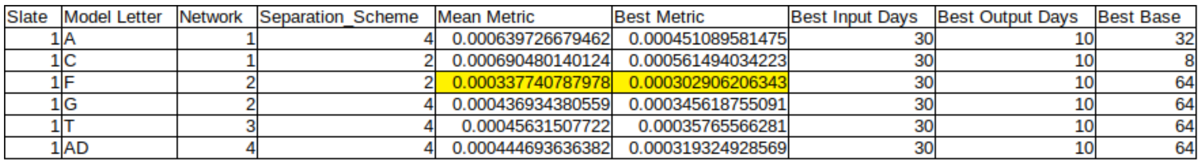
Separation Scheme	Scheme #	# of models needed
One datastream	1	3
All datastreams	2	1

Figure 54: Separation Schemes for the VitalDB dataset.

Slate Model Performances

As with Jornada, VitalDB individual slate metrics will be discussed first, followed by a discussion of experiment performance in relation to the research questions.

Slate 1 Slate 1 was composed of regression models for which took 30 seconds of input data and outputted the next 10 seconds of vital signs. For the LSTM regression models, clinical data was also provided as input (not so for autoencoders or prediciton models). For models that predicted datastreams separately, the metrics for the relevant model combinations which predict all datastreams is presented. Figure 1 55 displays the MSE results for Slate 1 LSTM models.



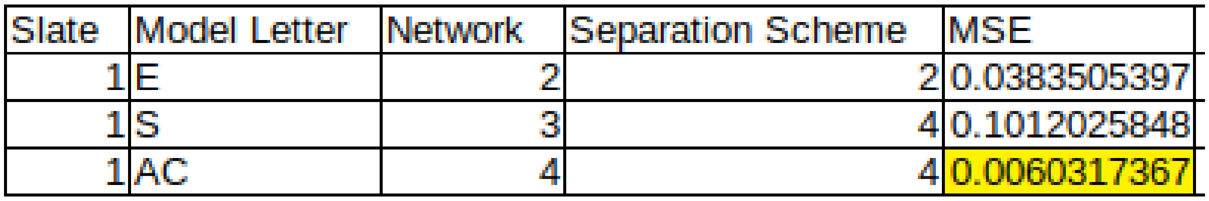
Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
1	A	1	4	0.000639726679462	0.000451089581475	30	10	32
1	C	1	2	0.000690480140124	0.000561494034223	30	10	8
1	F	2	2	0.000337740787978	0.000302906206343	30	10	64
1	G	2	4	0.000436934380559	0.000345618755091	30	10	64
1	T	3	4	0.00045631507722	0.00035765566281	30	10	64
1	AD	4	4	0.000444693636382	0.000319324928569	30	10	64

Figure 55: VitalDB Slate 1 Metrics for LSTM models.

Model letter F had the best performance for both the mean MSE and the lowest overall MSE. Model F is Network 2 model with separation scheme 2, meaning it predicts datastreams separately. Model F is actually 3 models (one per each datastream) with the metric in Figure 55 reporting the average MSE across the 3 models. Model F takes input from AE letter E, which processes each datastream separately.

The AE model performances are displayed in Figure 56.

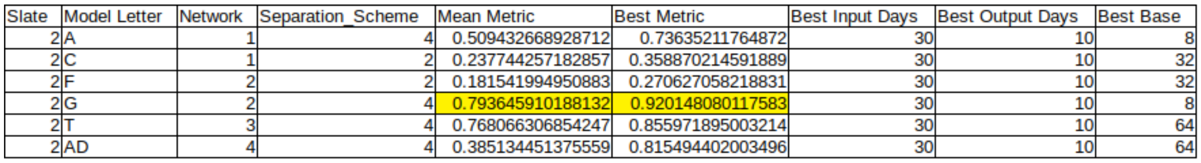
The best performing AE for this slate was model letter AC. Model AC processes all datastreams together, outputting a latent space with features from all datastreams. Model letter S for this slate did considerably more poorly than model letter E or AC.



Slate	Model Letter	Network	Separation Scheme	MSE
1	E	2	2	0.0383505397
1	S	3	4	0.1012025848
1	AC	4	4	0.0060317367

Figure 56: VitalDB Slate 1 Metrics for AE models.

Slate 2 Slate 2 was a prediction slate to determine whether or not the inputs represented an emergency operation. Slate 2 reused the AEs from Slate 1 and used the same inputs and outputs. Figure 57 displays the results for Slate 2.

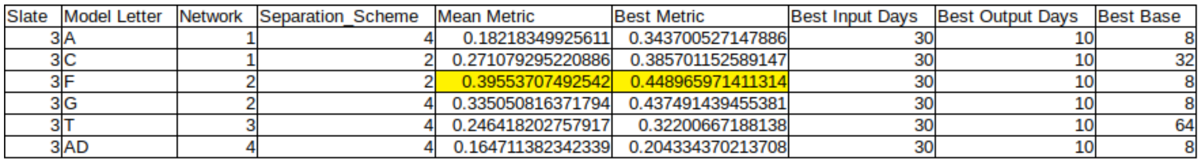


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
2	A	1	4	0.509432668928712	0.73635211764872	30	10	8
2	C	1	2	0.237744257182857	0.358870214591889	30	10	32
2	F	2	2	0.181541994950883	0.270627058218831	30	10	32
2	G	2	4	0.793645910188132	0.920148080117583	30	10	8
2	T	3	4	0.768066306854247	0.855971895003214	30	10	64
2	AD	4	4	0.385134451375559	0.815494402003496	30	10	64

Figure 57: VitalDB Slate 2 Metrics for LSTM models.

For Slate 2, model letter G had the highest mean F1 statistic and highest overall mean statistic. Model letter G is a network 2, separation scheme 4 model which takes input from model letter E per datastream and concatenates the latest space before processing it.

Slate 3 Slate 3 is the same as Slate 2, except that it predicts the presence or absence of a discharge mortality risk. Figure 58 displays the results for Slate 3.

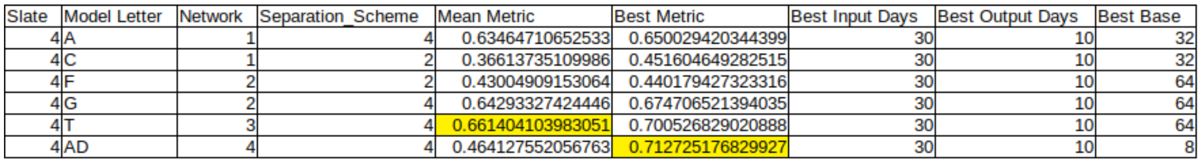


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
3	A	1	4	0.18218349925611	0.343700527147886	30	10	8
3	C	1	2	0.271079295220886	0.385701152589147	30	10	32
3	F	2	2	0.39553707492542	0.448965971411314	30	10	8
3	G	2	4	0.335050816371794	0.437491439455381	30	10	8
3	T	3	4	0.246418202757917	0.32200667188138	30	10	64
3	AD	4	4	0.164711382342339	0.204334370213708	30	10	8

Figure 58: VitalDB Slate 3 Metrics for LSTM models.

Model letter F had the highest mean and overall F1 statistic. However, it should be noted that the models across the board performed quite terribly for this task. Model F's highest F1 statistic was approximately 0.449, which was much lower than for Slate 2's prediction task.

Slate 4 Slate 4 is the same as Slates 2 and 3, but it predicts the presence or absence of the glucose risk factor. Figure 59 gives the results of Slate 4.



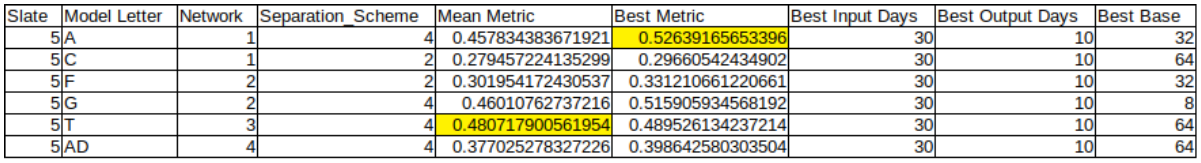
Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
4	A	1	4	0.63464710652533	0.650029420344399	30	10	32
4	C	1	2	0.36613735109986	0.451604649282515	30	10	32
4	F	2	2	0.43004909153064	0.440179427323316	30	10	64
4	G	2	4	0.64293327424446	0.674706521394035	30	10	64
4	T	3	4	0.661404103983051	0.700526829020888	30	10	64
4	AD	4	4	0.464127552056763	0.712725176829927	30	10	8

Figure 59: VitalDB Slate 4 Metrics for LSTM models.

For Slate 4, model letter T had the highest mean F1 statistic while model letter AD had the overall highest F1 statistic. Model letter T is a Network 3, separation scheme 4 model which outputs all datastreams together. T takes input from model AE letter S, which synthesizes individual datastream latent space outputs from model letter E. AD is a Network 4, separation scheme model which takes input only from model letter AC, with all datastreams processed together for both models.

It should be noted that model letter AD's mean performance was considerably lower than model letter T, G, and A's average performance for this task. T's best model was fairly close to model letter AD. Model letters A and G performance was close behind T.

Slate 5 Slate 5 combined slates 2-4 to try and predict the presence or absence of an emergency operation, discharge mortality risk, or glucose risk together. Figure 60 displays the results for Slate 5.



Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
5	A	1	4	0.457834383671921	0.52639165653396	30	10	32
5	C	1	2	0.279457224135299	0.29660542434902	30	10	64
5	F	2	2	0.301954172430537	0.331210661220661	30	10	32
5	G	2	4	0.46010762737216	0.515905934568192	30	10	8
5	T	3	4	0.480717900561954	0.489526134237214	30	10	64
5	AD	4	4	0.377025278327226	0.398642580303504	30	10	64

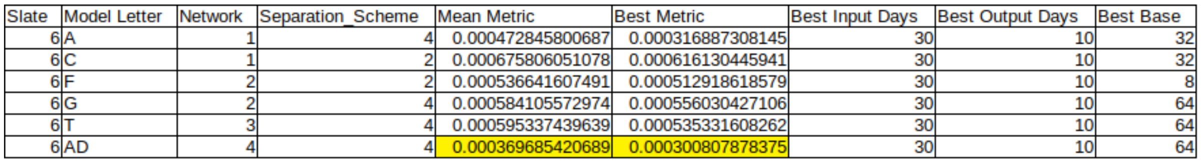
Figure 60: VitalDB Slate 5 Metrics for LSTM models.

Model letter T had the highest mean F1, while model letter A had the highest overall

F1. However, Slate 5 (similar to Slate 3) had generally bad performance for all models. The highest overall F1 statistic, from model letter A, was only about 0.526.

Given that this slate incorporates the same prediction task as the underperforming slate 3, this is not an altogether surprising result.

Slate 6 Slate 6 is a regression slate, essentially the same as Slate 1 except the "orch" (Orchestra infusion pump) datastream is excluded from the models. The models in Slate 6 were trained with only the ECG the Respirator information, as well as clinical parameters (regression models only). Slate 6 sets up future retraining in slate 8. Because of the missing datastream, Slate 6 retrains the needed autoencoders. Figure 61 displays the results from Slate 6.

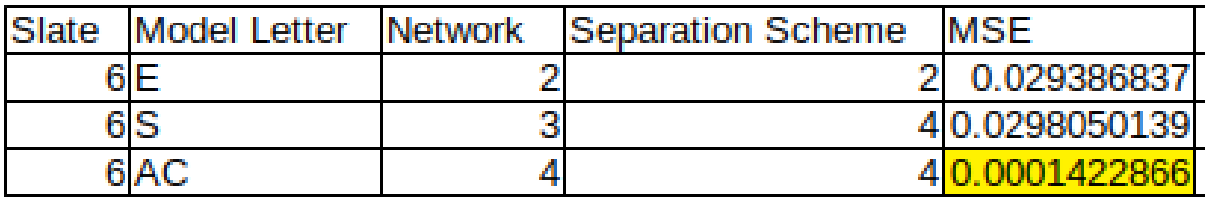


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
6	A	1	4	0.000472845800687	0.000316887308145	30	10	32
6	C	1	2	0.000675806051078	0.000616130445941	30	10	32
6	F	2	2	0.000536641607491	0.000512918618579	30	10	8
6	G	2	4	0.000584105572974	0.000556030427106	30	10	64
6	T	3	4	0.000595337439639	0.000535331608262	30	10	64
6	AD	4	4	0.000369685420689	0.000300807878375	30	10	64

Figure 61: VitalDB Slate 6 Metrics for LSTM models.

Model AD had the lowest mean and overall MSE for this slate. Model A was the only other model with similar performance to model AD.

Figure 62 gives the AE metrics for Slate 6.

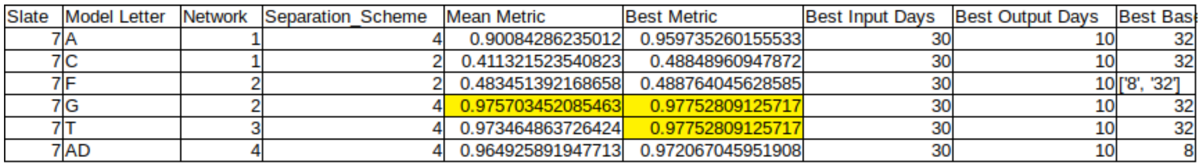


Slate	Model Letter	Network	Separation Scheme	MSE
6	E	2	2	0.029386837
6	S	3	4	0.0298050139
6	AC	4	4	0.0001422866

Figure 62: VitalDB Slate 6 Metrics for AE models.

For this slate, model letter AE had the lowest MSE, while model letters E and S had much higher MSE. Models E and S's errors did not vary much from one another.

Slate 7 Slate 7 was essentially the same as slate 6, but with the prediction task for emergency operations. Slate 7 also did not include the "orch" datastream. Since this slate was a prediction task, it did not use any clinical data as input. Figure 63 displays the results for Slate 7.

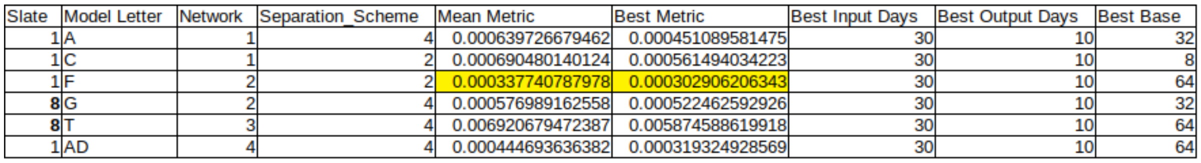


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Bas
7	A	1	4	0.90084286235012	0.959735260155533	30	10	32
7	C	1	2	0.411321523540823	0.48848960947872	30	10	32
7	F	2	2	0.483451392168658	0.488764045628585	30	10	['8', '32']
7	G	2	4	0.975703452085463	0.97752809125717	30	10	32
7	T	3	4	0.973464863726424	0.97752809125717	30	10	32
7	AD	4	4	0.964925891947713	0.972067045951908	30	10	8

Figure 63: VitalDB Slate 7 Metrics for LSTM models.

For this slate, model G had the highest mean F1 statistic, while models G and T tied for the overall highest F1 statistic. Models G, T, and AD all did well for this task, with model A not far behind. Models C and F performed comparatively poorly on this task.

Slate 8 Slate 8 retrained the models from Slate 6 with the ”orch” datastream added back in. As in the Jornada Experiments, some models would need to be retrained end-to-end, making them equivalent models to slate 1. In Slates 8 and 9, only LSTM model letters G and T needed to be investigated at since they took input in combination from the retrained AE models. Slate 8 is a regression slate. The results are presented in Figure 64 with the missing models added from Slate 1 for comparison.



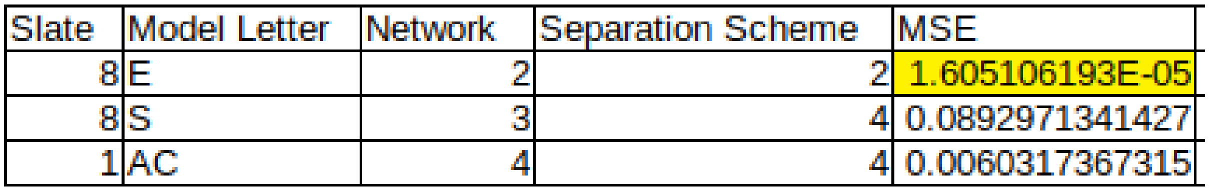
Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
1	A	1	4	0.000639726679462	0.000451089581475	30	10	32
1	C	1	2	0.000690480140124	0.000561494034223	30	10	8
1	F	2	2	0.000337740787978	0.000302906206343	30	10	64
8	G	2	4	0.000576989162558	0.000522462592926	30	10	32
8	T	3	4	0.006920679472387	0.005874588619918	30	10	64
1	AD	4	4	0.000444693636382	0.000319324928569	30	10	64

Figure 64: VitalDB Slate 8 Metrics for LSTM models.

In Figure 64 it can be seen that in the combination Slate, model letter F (retrained) still would have fared better than the retrained models G or T, as in Slate 1.

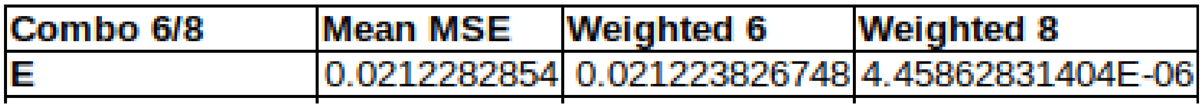
The AE metrics for Slate 8 are presented in Figure 65. For model letter E, only the model for the new datastream was trained, which is why it has a much lower MSE.

The overall performance of model letter E from Slate 8 in combination with Slate 6 for all datastreams is given in Figure 66.



Slate	Model Letter	Network	Separation Scheme	MSE
8	E	2	2	1.605106193E-05
8	S	3	4	0.0892971341427
1	AC	4	4	0.0060317367315

Figure 65: VitalDB Slate 8 Metrics for AE models.

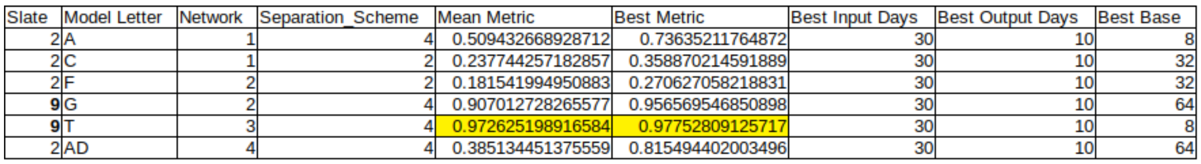


Combo 6/8	Mean MSE	Weighted 6	Weighted 8
E	0.0212282854	0.021223826748	4.45862831404E-06

Figure 66: VitalDB Slate 6 and 8 combination performance for model letter E.

Ultimately, models S and AC used for this slate and its comparison outperformed the newly retrained combination of AE model letters for E.

Slate 9 Slate 9 is the same as Slate 7, with the "orch" datastream added back in. As with Slate 8, only models G and T are relevant for this task, using the combination of retrained AEs from slates 6 and 8. Slate 9 is a prediction task for the emergency operation factor. Figure 67 displays the results for Slate 9.



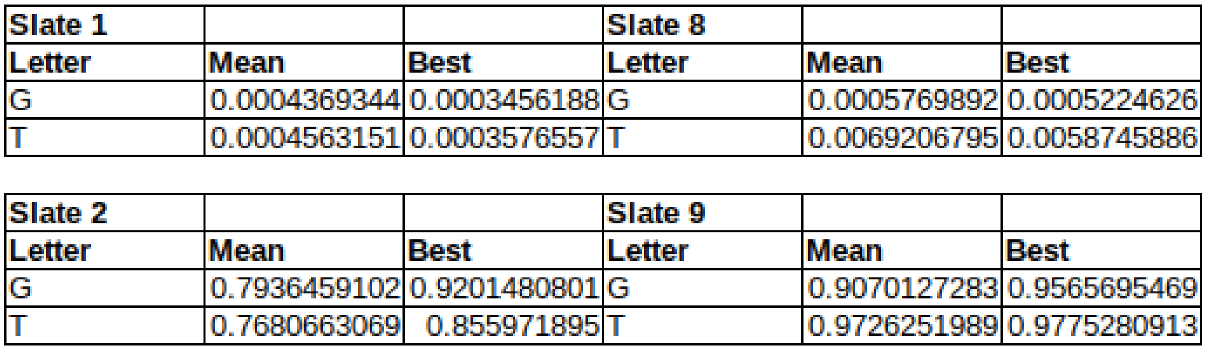
Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
2	A	1	4	0.509432668928712	0.73635211764872	30	10	8
2	C	1	2	0.237744257182857	0.358870214591889	30	10	32
2	F	2	2	0.181541994950883	0.270627058218831	30	10	32
9	G	2	4	0.907012728265577	0.956569546850898	30	10	64
9	T	3	4	0.972625198916584	0.97752809125717	30	10	8
2	AD	4	4	0.385134451375559	0.815494402003496	30	10	64

Figure 67: VitalDB Slate 9 Metrics for LSTM models

Models G and T, retrained from Slate 9, fared best in this slate, with model T having the highest overall and mean F1 statistic.

Sanity Checking Slates 8 and 9 Slates 8 and 9 were mostly run as a sanity check to verify that the performance of retrained models aligned with expectations set in Slates 1 and 2. Ideally, there should be little difference in performance with the retrained models as the initial models.

However, overall less VitalDB models were run per slate than Jornada, and the model behavior is less consistent between runs. Figure 68 displays the comparison of Slates 1 and 8, and Slates 2 and 9.



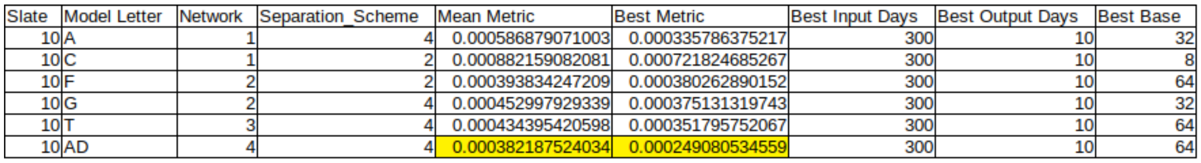
Slate 1			Slate 8		
Letter	Mean	Best	Letter	Mean	Best
G	0.0004369344	0.0003456188	G	0.0005769892	0.0005224626
T	0.0004563151	0.0003576557	T	0.0069206795	0.0058745886

Slate 2			Slate 9		
Letter	Mean	Best	Letter	Mean	Best
G	0.7936459102	0.9201480801	G	0.9070127283	0.9565695469
T	0.7680663069	0.855971895	T	0.9726251989	0.9775280913

Figure 68: VitalDB Slates 8 and 9 Mean Metric Comparison.

It can be seen that model performances varied more from each other than in Jornada, but not dramatically. The one exception was model letter T in Slate 8, which performed quite a bit worse than in Slate 1 (though in Slate 9, it performed better than Slate 2). Less experimental runs might explain the more variable nature of the mean performance for these slates. At the same time, the emergency operation may just be a difficult prediction task for which good performance is more difficult for a model to converge to.

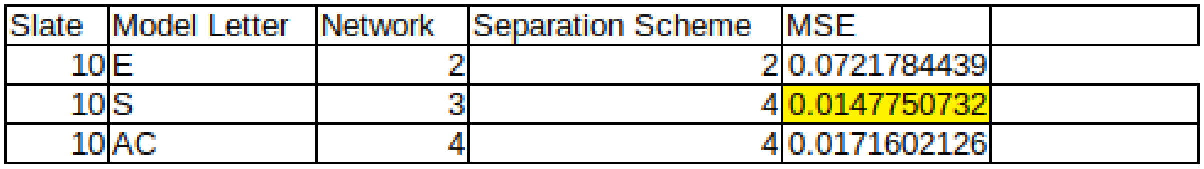
Slate 10 Slate 10 is a regression slate, essentially the same as Slate 1 except for using 300 seconds (5 minutes) of input rather than 30 seconds. Otherwise, it uses the same parameters as slate 1.



Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
10	A	1	4	0.000586879071003	0.000335786375217	300	10	32
10	C	1	2	0.000882159082081	0.000721824685267	300	10	8
10	F	2	2	0.000393834247209	0.000380262890152	300	10	64
10	G	2	4	0.000452997929339	0.000375131319743	300	10	32
10	T	3	4	0.000434395420598	0.000351795752067	300	10	64
10	AD	4	4	0.000382187524034	0.000249080534559	300	10	64

Figure 69: VitalDB Slate 10 Metrics for LSTM models.

In Slate 10, the lowest mean and overall MSE were found in model letter AD. Model letter F was slightly behind AD.



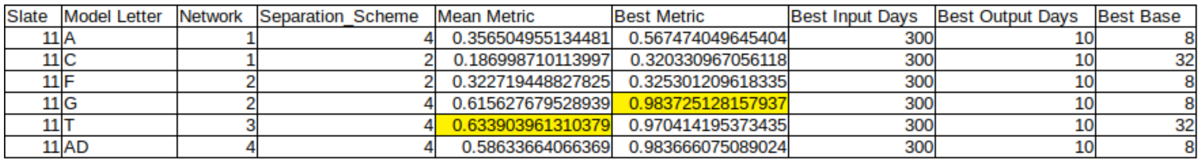
Slate	Model Letter	Network	Separation Scheme	MSE	
10	E	2	2	0.0721784439	
10	S	3	4	0.0147750732	
10	AC	4	4	0.0171602126	

Figure 70: VitalDB Slate 10 Metrics for AE models.

Figure 70 displays the AE results for slate 10.

In Slate 10, model letter S had the lowest MSE, with model letter AC not far behind.

Slate 11 Slate 11 was the same as Slate 10, but a prediction task for the presence or absence of an emergency operation. Like slate 10, Slate 11 used 300 seconds of input rather than 30. Figure 71 displays the results from Slate 11.



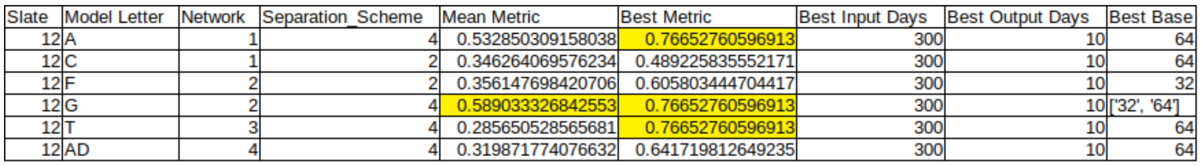
Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
11	A	1	4	0.356504955134481	0.567474049645404	300	10	8
11	C	1	2	0.186998710113997	0.320330967056118	300	10	32
11	F	2	2	0.322719448827825	0.325301209618335	300	10	8
11	G	2	4	0.615627679528939	0.983725128157937	300	10	8
11	T	3	4	0.633903961310379	0.970414195373435	300	10	32
11	AD	4	4	0.58633664066369	0.983666075089024	300	10	8

Figure 71: VitalDB Slate 11 Metrics for LSTM models.

In this slate, model letter T had the highest overall mean F1 statistic while model letter G had the overall highest F1 statistic. However, models G,T, and AD were all relatively close to one another in terms of performance, especially regarding their overall best models. Model G synthesizes input from AE letter E, while model T takes input from AE letter S, which synthesizes input from AE letter E. Model letter AD takes input from AE letter AC, which processes all datastreams together.

Slate 12 Slate 12 was the identical to Slate 11 except that it predicted the discharge mortality risk. Figure 72 displays the results of slate 12.

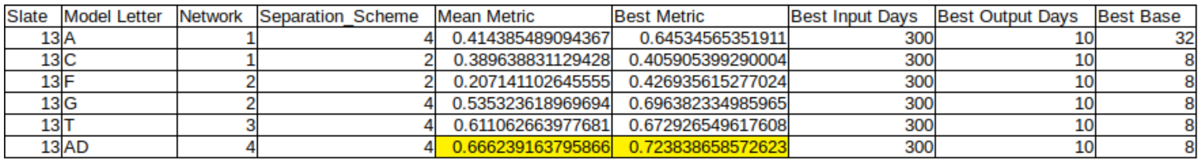
The highest mean F1 statistic for this slate belonged to model letter G, while the overall highest F1 statistic belonged to A, G and T. For average performance, none of the models did very well. In terms of mean F1, none of the models performed very well, while the overall highest F1 models were acceptable compared to the analogous Slate 3.



Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
12	A	1	4	0.532850309158038	0.76652760596913	300	10	64
12	C	1	2	0.346264069576234	0.489225835552171	300	10	64
12	F	2	2	0.356147698420706	0.605803444704417	300	10	32
12	G	2	4	0.589033326842553	0.76652760596913	300	10	['32', '64']
12	T	3	4	0.285650528565681	0.76652760596913	300	10	64
12	AD	4	4	0.319871774076632	0.641719812649235	300	10	64

Figure 72: VitalDB Slate 12 Metrics for LSTM models.

Slate 13 Slate 13 is identical to Slates 11 and 12 except that it predicted glucose risk instead of emergency operations or discharge mortality risk.

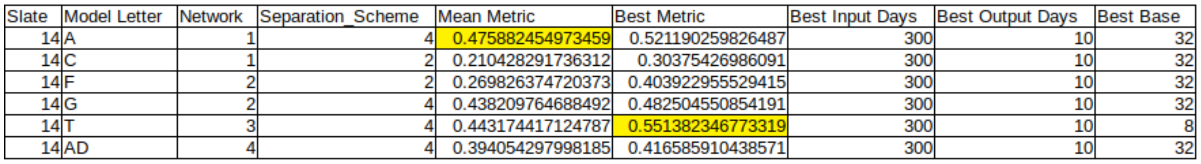


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
13	A	1	4	0.414385489094367	0.64534565351911	300	10	32
13	C	1	2	0.389638831129428	0.405905399290004	300	10	8
13	F	2	2	0.207141102645555	0.426935615277024	300	10	8
13	G	2	4	0.535323618969694	0.696382334985965	300	10	8
13	T	3	4	0.611062663977681	0.672926549617608	300	10	8
13	AD	4	4	0.666239163795866	0.723838658572623	300	10	8

Figure 73: VitalDB Slate 13 Metrics for LSTM models.

The mean highest and overall highest F1 statistics for this slate occurred with model letter AD. The performance. Model letter T was just behind model letter AD in terms of mean and overall performance, while model letter G's overall performance was also close.

Slate 14 Slate 14 was the combination of Slates 11, 12, and 13, which attempted to predict the presence or absence of an emergency operation, discharge mortality risk, and glucose risk altogether. Figure 74 gives the results of Slate 14.

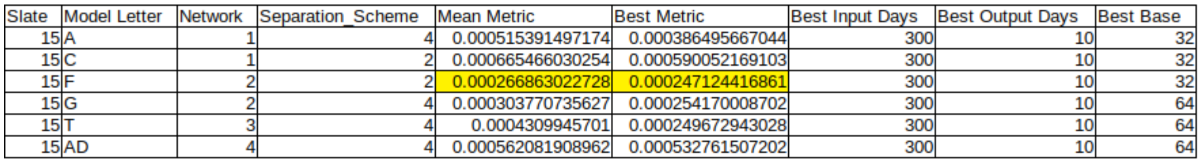


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
14	A	1	4	0.475882454973459	0.521190259826487	300	10	32
14	C	1	2	0.210428291736312	0.30375426986091	300	10	32
14	F	2	2	0.269826374720373	0.403922955529415	300	10	32
14	G	2	4	0.438209764688492	0.482504550854191	300	10	32
14	T	3	4	0.443174417124787	0.551382346773319	300	10	8
14	AD	4	4	0.394054297998185	0.416585910438571	300	10	32

Figure 74: VitalDB Slate 14 Metrics for LSTM models.

In this slate, no model performed very well. The mean highest F1 belonged to model letter A, while the highest overall F1 belonged to model letter T. A and T were close to each other in both measures. However, the overall highest F1 statistic was only about 0.551, which is not good performance.

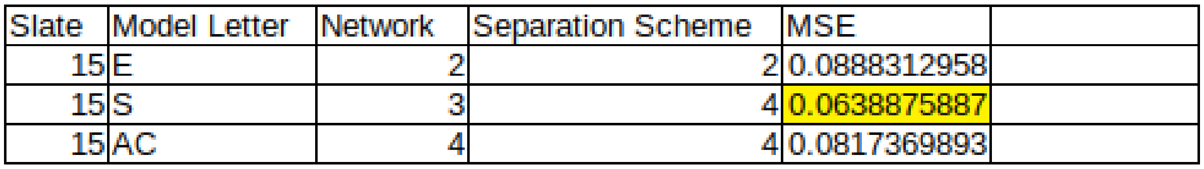
Slate 15 Slate 15 was identical to slate 10 except that it removed the "orch" datastream. It's analogous slate is slate 6, except that slate 15 uses 300 input seconds instead of 30. Figure 75 displays the results of Slate 15.



Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
15	A	1	4	0.000515391497174	0.000386495667044	300	10	32
15	C	1	2	0.000665466030254	0.000590052169103	300	10	32
15	F	2	2	0.000266863022728	0.000247124416861	300	10	32
15	G	2	4	0.000303770735627	0.000254170008702	300	10	64
15	T	3	4	0.0004309945701	0.000249672943028	300	10	64
15	AD	4	4	0.000562081908962	0.000532761507202	300	10	64

Figure 75: VitalDB Slate 15 Metrics for LSTM models.

Slate 15 had the lowest mean and overall MSE with model letter F, with G close behind. F is a network 2 separation scheme 2 model (datastream-specific). Figure 76

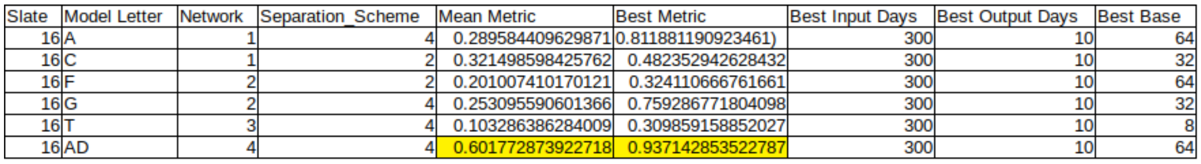


Slate	Model Letter	Network	Separation Scheme	MSE	
15	E	2	2	0.0888312958	
15	S	3	4	0.0638875887	
15	AC	4	4	0.0817369893	

Figure 76: VitalDB Slate 15 Metrics for AE models.

Model letter S had the lowest overall MSE for AEs in Slate 15. Model letters E and AC had very similar performance.

Slate 16 Slate 16 is identical to Slate 15 in the removal of the "orch" datastream but is a prediction task for the emergency operation factor. Figure 77 shows the results of Slate 16.

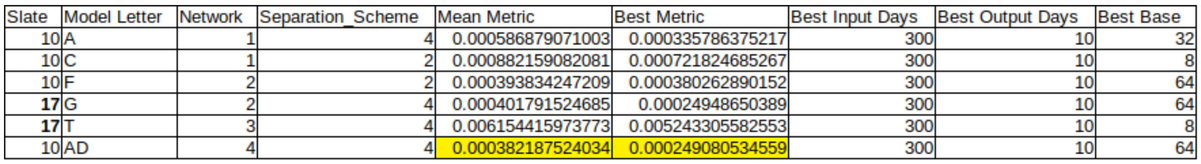


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
16	A	1	4	0.289584409629871	0.811881190923461)	300	10	64
16	C	1	2	0.321498598425762	0.482352942628432	300	10	32
16	F	2	2	0.201007410170121	0.324110666761661	300	10	64
16	G	2	4	0.253095590601366	0.759286771804098	300	10	32
16	T	3	4	0.103286386284009	0.309859158852027	300	10	8
16	AD	4	4	0.601772873922718	0.937142853522787	300	10	64

Figure 77: VitalDB Slate 16 Metrics for LSTM models.

For Slate 16, model letter AD had the mean highest F1 statistic and overall highest F1 statistic. The overall highest F1 statistic for model letter A was close behind model letter AD, but its mean performance was significantly lower.

Slate 17 Slate 17 retrained AEs and associated LSTM models from Slate 15 with the "orch" datastream added back in, analogous to Slate 8 with Slate 6. The comparison models from Slate 10 are provided for this simulated combination. Slate 17 is a regression task. Figure 78 displays the results.

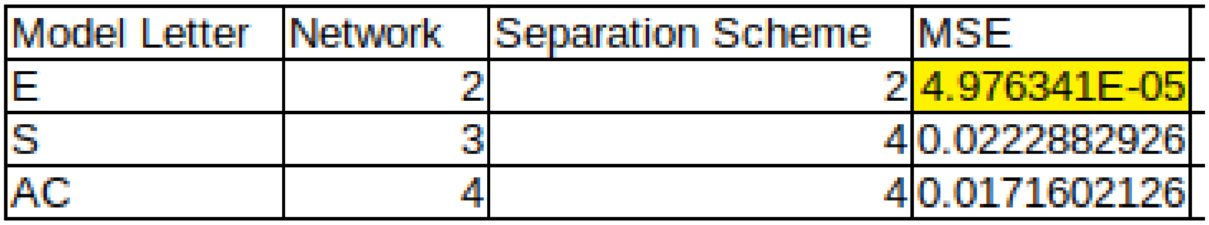


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
10	A	1	4	0.000586879071003	0.000335786375217	300	10	32
10	C	1	2	0.000882159082081	0.000721824685267	300	10	8
10	F	2	2	0.000393834247209	0.000380262890152	300	10	64
17	G	2	4	0.000401791524685	0.00024948650389	300	10	64
17	T	3	4	0.006154415973773	0.005243305582553	300	10	8
10	AD	4	4	0.000382187524034	0.000249080534559	300	10	64

Figure 78: VitalDB Slate 17 Metrics for LSTM models.

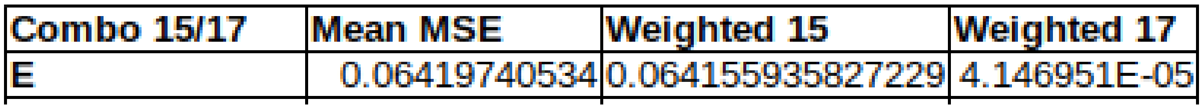
The comparison model letter AD from Slate 10 still had lowest mean and overall MSE.

The retrained AEs and comparison are displayed in Figure 79. Note that the AE model letter E in this slate only represents the one trained model for the missing datastream, which is why it is comparatively lower. Figure 80 gives the value of the overall performance of model letter E representing all datastreams from Slates 15 and 17.



Model Letter	Network	Separation Scheme	MSE
E	2	2	4.976341E-05
S	3	4	0.0222882926
AC	4	4	0.0171602126

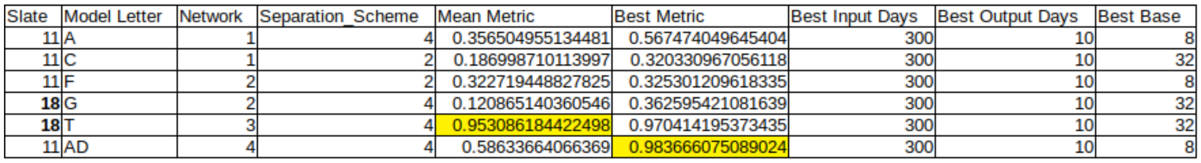
Figure 79: VitalDB Slate 17 Metrics for AE models.



Combo 15/17	**Mean MSE**	**Weighted 15**	**Weighted 17**
E	0.06419740534	0.064155935827229	4.146951E-05

Figure 80: VitalDB Slate 15 and 17 combination performance for model letter E.

Slate 18 Slate 18 was the same as 17, except adding back in the "orch "datastream for the emergency operation prediction task. Figure 81 displays the results from Slate 18, with missing models provided by comparison Slate 11.

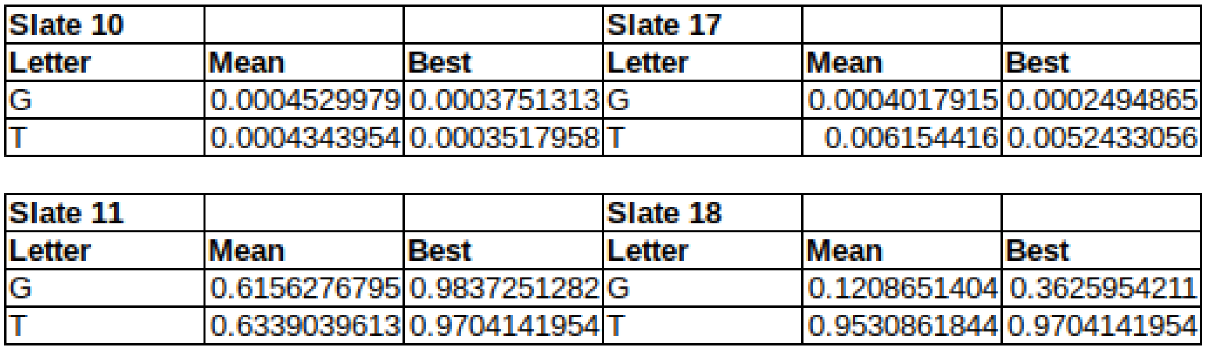


Slate	Model Letter	Network	Separation_Scheme	Mean Metric	Best Metric	Best Input Days	Best Output Days	Best Base
11	A	1	4	0.356504955134481	0.567474049645404	300	10	8
11	C	1	2	0.186998710113997	0.320330967056118	300	10	32
11	F	2	2	0.322719448827825	0.325301209618335	300	10	8
18	G	2	4	0.120865140360546	0.362595421081639	300	10	32
18	T	3	4	0.953086184422498	0.970414195373435	300	10	32
11	AD	4	4	0.58633664066369	0.983666075089024	300	10	8

Figure 81: VitalDB Slate 18 Metrics for LSTM models.

In this simulated comparison slate, model letter T from slate 18 had the highest mean F1 statistic, while model letter AD from comparison Slate 11 had the overall highest F1 statistic. Model T's mean and overall highest F1 statistic were much closer to one another compared to the other model models letters.

Sanity Checking Slates 17 and 18 As with Slates 8 and 9, Slates 17 and 18 were sanity-checked to ensure model performance when retrained with consistent with expectations. Figure 82 displays the mean and best metric comparison between Slates 10 and 17 and Slates 11 and 18.



Slate 10			**Slate 17**		
Letter	**Mean**	**Best**	**Letter**	**Mean**	**Best**
G	0.0004529979	0.0003751313	G	0.0004017915	0.0002494865
T	0.0004343954	0.0003517958	T	0.006154416	0.0052433056

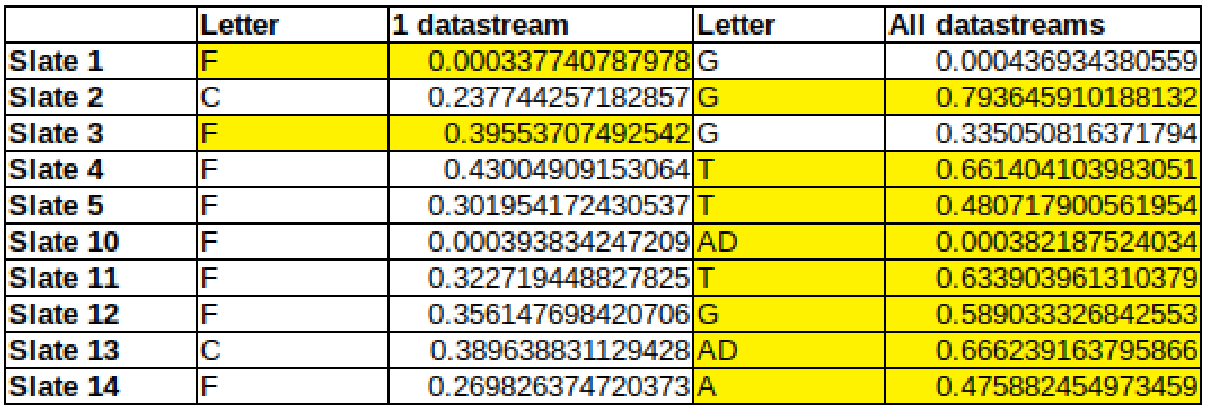
Slate 11			**Slate 18**		
Letter	**Mean**	**Best**	**Letter**	**Mean**	**Best**
G	0.6156276795	0.9837251282	G	0.1208651404	0.3625954211
T	0.6339039613	0.9704141954	T	0.9530861844	0.9704141954

Figure 82: VitalDB Slates 17 and 18 Mean and Best Metric Comparison.

While mostly the models did not see huge differences in performance, model letter G in Slate 18 had much worse performance than model letter G in Slate 11. As with Slates 8 and 9, there was less consistency in between runs for the Vital models. In conjunction with Slates 8 and 9, it is reasonable to conclude that while there is a fair amount of variation in the model predictions from run to run, retraining does not appear to have a consistent negative impact on model outputs, as expected.

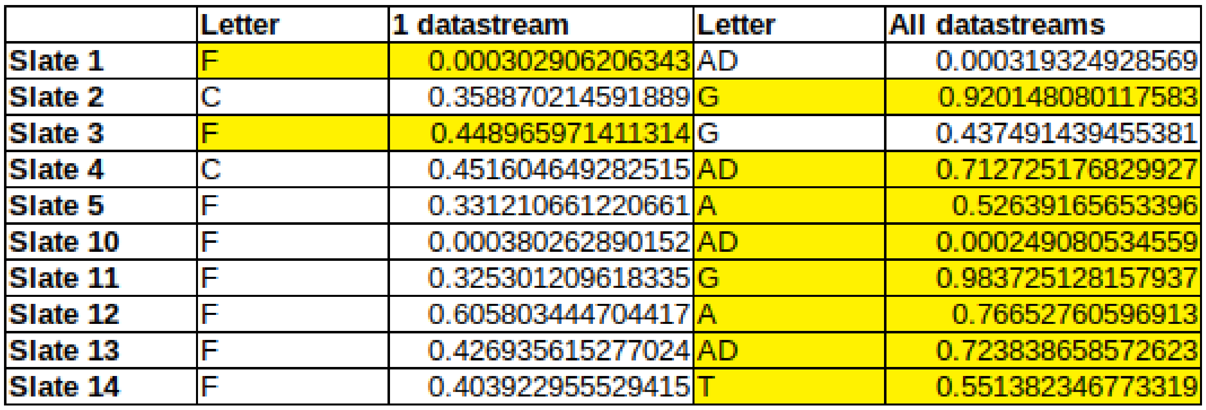
VitalDB Combination Models and Separation Schemes

As with the Jornada Basin Models, one aspect of this dissertation was to determine if one combination model or several datastream-specific models performed best on a given task. Vital DB only had separation schemes 2 and 4 rather than 1-4, since data represented individual patients who were not undergoing simultaneously, rather than weather collection sites which collected data at the same time. Therefore, separation by individual patient case was not feasible. Figure 83 gives the mean best performers for Slates 1-5 and 10-14, while Figure 84 gives the overall best performers for the same slates.



	Letter	1 datastream	Letter	All datastreams
Slate 1	F	0.000337740787978	G	0.000436934380559
Slate 2	C	0.237744257182857	G	0.793645910188132
Slate 3	F	0.39553707492542	G	0.335050816371794
Slate 4	F	0.43004909153064	T	0.661404103983051
Slate 5	F	0.301954172430537	T	0.480717900561954
Slate 10	F	0.000393834247209	AD	0.000382187524034
Slate 11	F	0.322719448827825	T	0.633903961310379
Slate 12	F	0.356147698420706	G	0.589033326842553
Slate 13	C	0.389638831129428	AD	0.666239163795866
Slate 14	F	0.269826374720373	A	0.475882454973459

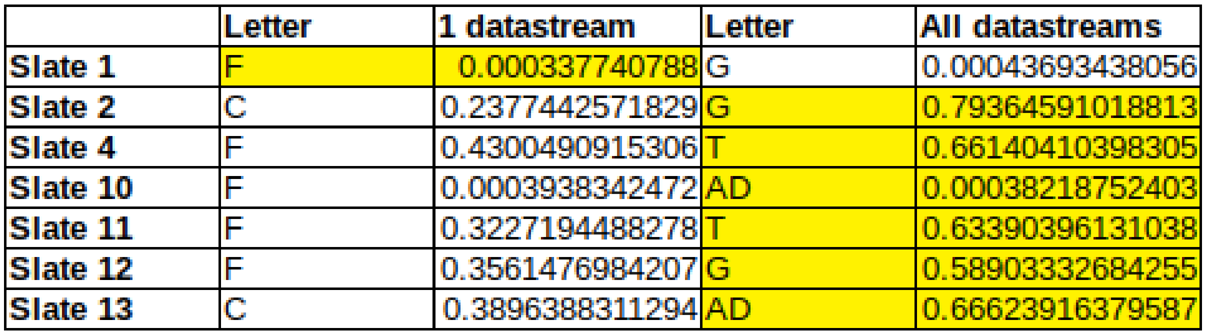
Figure 83: Separation schemes and top performers for metric mean by slate.



	Letter	1 datastream	Letter	All datastreams
Slate 1	F	0.000302906206343	AD	0.000319324928569
Slate 2	C	0.358870214591889	G	0.920148080117583
Slate 3	F	0.448965971411314	G	0.437491439455381
Slate 4	C	0.451604649282515	AD	0.712725176829927
Slate 5	F	0.331210661220661	A	0.52639165653396
Slate 10	F	0.000380262890152	AD	0.000249080534559
Slate 11	F	0.325301209618335	G	0.983725128157937
Slate 12	F	0.605803444704417	A	0.76652760596913
Slate 13	F	0.426935615277024	AD	0.723838658572623
Slate 14	F	0.403922955529415	T	0.551382346773319

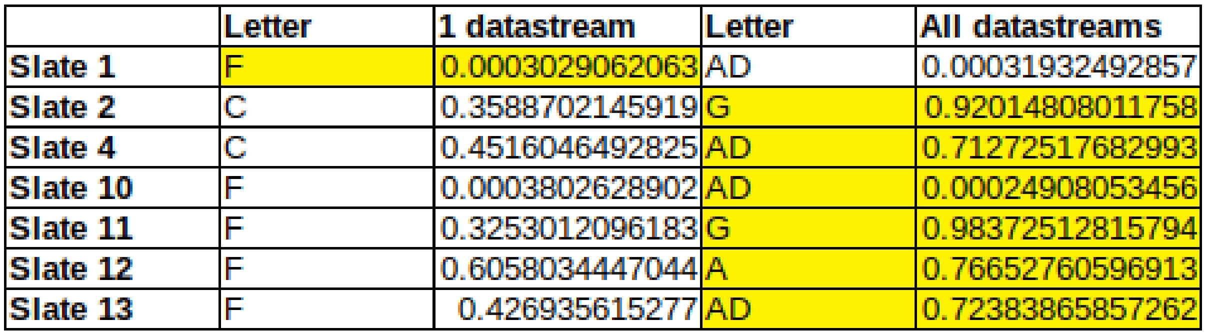
Figure 84: Separation schemes and top performers for overall best metric by slate.

In these Slates 1-5 and 10-14, however, some of the prediction tasks did not succeed. To be able to analyze actual performant models, any prediction model that did not achieve an F1 statistic of at least 0.7 in some parameter combination was removed from consideration. This removed the combinations of prediction models (Slates 5 and 14) and one of the discharge mortality risk slates (Slate 3). Figure 85 shows the mean performance by slate for the remaining models, while Figure 86 gives the overall best performances for these same models.



	Letter	1 datastream	Letter	All datastreams
Slate 1	F	0.000337740788	G	0.00043693438056
Slate 2	C	0.2377442571829	G	0.79364591018813
Slate 4	F	0.4300490915306	T	0.66140410398305
Slate 10	F	0.0003938342472	AD	0.00038218752403
Slate 11	F	0.3227194488278	T	0.63390396131038
Slate 12	F	0.3561476984207	G	0.58903332684255
Slate 13	C	0.3896388311294	AD	0.66623916379587

Figure 85: Separation schemes and top performers for metric mean by slate, limited to "good" models.



	Letter	1 datastream	Letter	All datastreams
Slate 1	F	0.0003029062063	AD	0.00031932492857
Slate 2	C	0.3588702145919	G	0.92014808011758
Slate 4	C	0.4516046492825	AD	0.71272517682993
Slate 10	F	0.0003802628902	AD	0.00024908053456
Slate 11	F	0.3253012096183	G	0.98372512815794
Slate 12	F	0.6058034447044	A	0.76652760596913
Slate 13	F	0.426935615277	AD	0.72383865857262

Figure 86: Separation schemes and top performers for overall metric by slate, limited to "good" models.

In the majority of the performant models, separation scheme 4 worked much better than separation scheme 2. For regression tasks, the MSE between separation schemes 2 and 4 were very close, but for prediction tasks, separation scheme 4 was far and away a better option. Slate 1 was the only slate that saw slightly better performance when separating out by datastream.

VitalDB saw very similar results to the Jornada Basin Dataset with regard to separation scheme: LSTM models worked much better when able to process all datastreams together at some point in the process. The data in the VitalDB set is highly interrelated, as each individual sample is coming from a single person undergoing liver transplant surgery. It is logical that interactions between the different datastreams would be important for the model's performance.

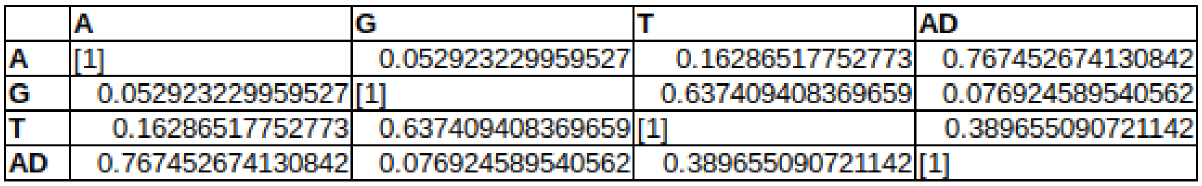
VitalDB Network Organizations

For the performant slates, model letter G had the top mean 3 times, model T and AD 2 times each, and model F one time. For overall best performer, model AD had the top spot 4 times, model G had it 2 times, and model A once. The overall minimum mean MSE for a regression task was found in Slate 10 for model AD and for a prediction task, in slate 2 for model G (emergency operation prediction task). The overall best MSE for a regression model was found in Slate 10 for model letter AD and the overall maximum F1 statistic for a prediction model was found in Slate 11 for model letter G (emergency operation prediction task).

In all cases except for the maximum F1 statistic in Slate 12 (discharge mortality prediction task), an LSTM model utilizing an AE outperformed the monolithic model for models that processed datastreams together.

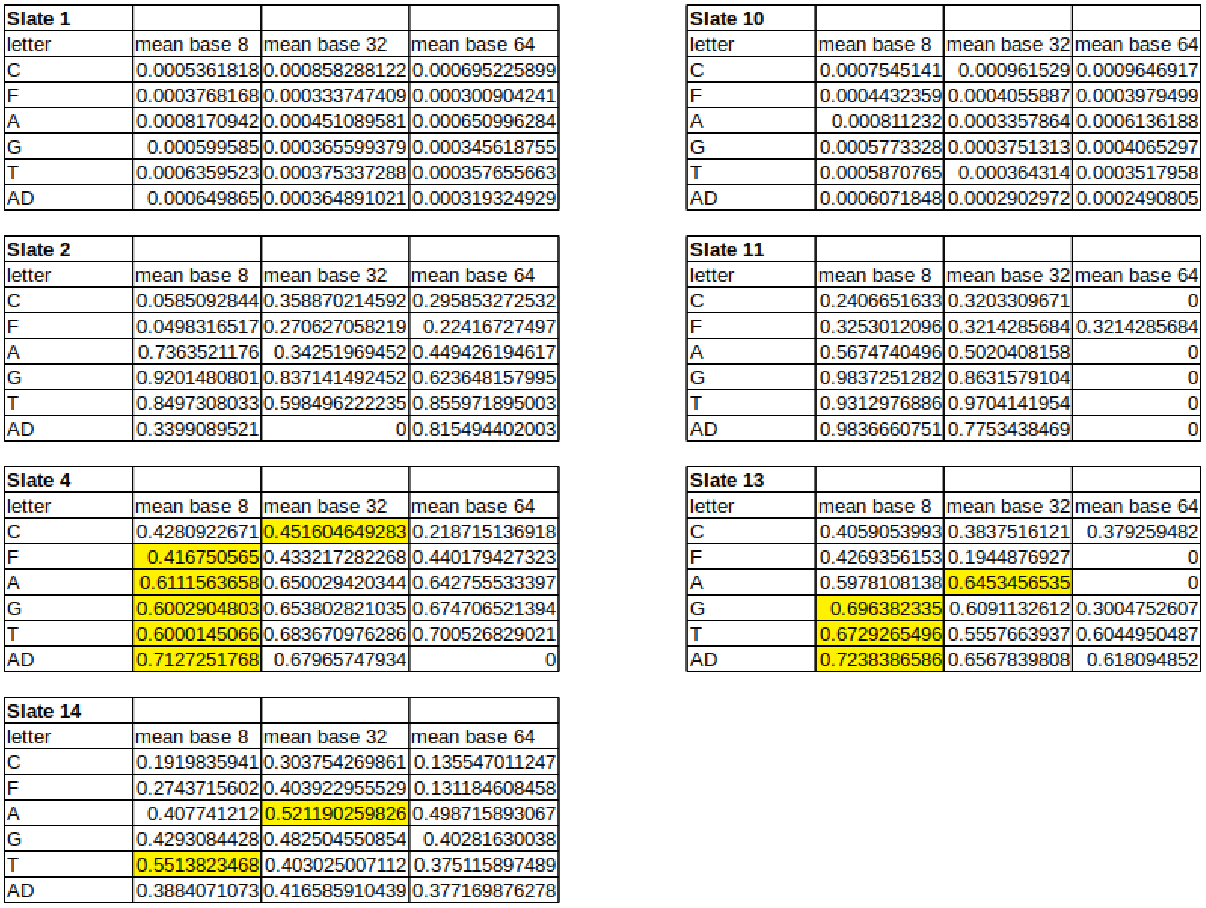
While it might make the researcher (and the researcher's dissertation committee) happy to conclude that AE models are better suited to VitalDB prediction tasks, this conclusion is likely erroneous. Figure 87 displays the p-values of Wilcoxon signed rank test between model letters A, G, T, and AD for Slates 1-7 and 10-16. None of the models demonstrated a statistically significant difference in distributions from any other.

Of course, this is not necessarily a death knell for the usefulness of the AE models in this context! But Figure 88 is.



	A	G	T	AD
A	[1]	0.052923229959527	0.16286517752773	0.767452674130842
G	0.052923229959527	[1]	0.637409408369659	0.076924589540562
T	0.16286517752773	0.637409408369659	[1]	0.389655090721142
AD	0.767452674130842	0.076924589540562	0.389655090721142	[1]

Figure 87: Wilcoxon signed rank test between model letters A, G, T, and AD.



Slate 1			
letter	mean base 8	mean base 32	mean base 64
C	0.0005361818	0.000858288122	0.000695225899
F	0.0003768168	0.000333747409	0.000300904241
A	0.0008170942	0.000451089581	0.000650996284
G	0.000599585	0.000365599379	0.000345618755
T	0.0006359523	0.000375337288	0.000357655663
AD	0.000649865	0.000364891021	0.000319324929

Slate 10			
letter	mean base 8	mean base 32	mean base 64
C	0.0007545141	0.000961529	0.0009646917
F	0.0004432359	0.0004055887	0.0003979499
A	0.000811232	0.0003357864	0.0006136188
G	0.0005773328	0.0003751313	0.0004065297
T	0.0005870765	0.000364314	0.0003517958
AD	0.0006071848	0.0002902972	0.0002490805

Slate 2			
letter	mean base 8	mean base 32	mean base 64
C	0.0585092844	0.358870214592	0.295853272532
F	0.0498316517	0.270627058219	0.22416727497
A	0.7363521176	0.34251969452	0.449426194617
G	0.9201480801	0.837141492452	0.623648157995
T	0.8497308033	0.598496222235	0.855971895003
AD	0.3399089521	0	0.815494402003

Slate 11			
letter	mean base 8	mean base 32	mean base 64
C	0.2406651633	0.3203309671	0
F	0.3253012096	0.3214285684	0.3214285684
A	0.5674740496	0.5020408158	0
G	0.9837251282	0.8631579104	0
T	0.9312976886	0.9704141954	0
AD	0.9836660751	0.7753438469	0

Slate 4			
letter	mean base 8	mean base 32	mean base 64
C	0.4280922671	0.451604649283	0.218715136918
F	0.416750565	0.433217282268	0.440179427323
A	0.6111563658	0.650029420344	0.642755533397
G	0.6002904803	0.653802821035	0.674706521394
T	0.6000145066	0.683670976286	0.700526829021
AD	0.7127251768	0.67965747934	0

Slate 13			
letter	mean base 8	mean base 32	mean base 64
C	0.4059053993	0.3837516121	0.379259482
F	0.4269356153	0.1944876927	0
A	0.5978108138	0.6453456535	0
G	0.696382335	0.6091132612	0.3004752607
T	0.6729265496	0.5557663937	0.6044950487
AD	0.7238386586	0.6567839808	0.618094852

Slate 14			
letter	mean base 8	mean base 32	mean base 64
C	0.1919835941	0.303754269861	0.135547011247
F	0.2743715602	0.403922955529	0.131184608458
A	0.407741212	0.521190259826	0.498715893067
G	0.4293084428	0.482504550854	0.40281630038
T	0.5513823468	0.403025007112	0.375115897489
AD	0.3884071073	0.416585910439	0.377169876278

Figure 88: Performance by base network structure for model letters in Slates 1, 2, 4, 10, 11, 12, and 13.

Figure 88 gives the mean performance by network structure depth for model letters in the performant slate. Highlights indicate where, if anywhere, an AE model beat monolithic model letter A's performance at an earlier depth. This occurred only in 3 of the 7 slates pictured. Additionally, the performance by depth demonstrates extremely high sensitivity of the models to network depth as regards overall performance on the prediction tasks.

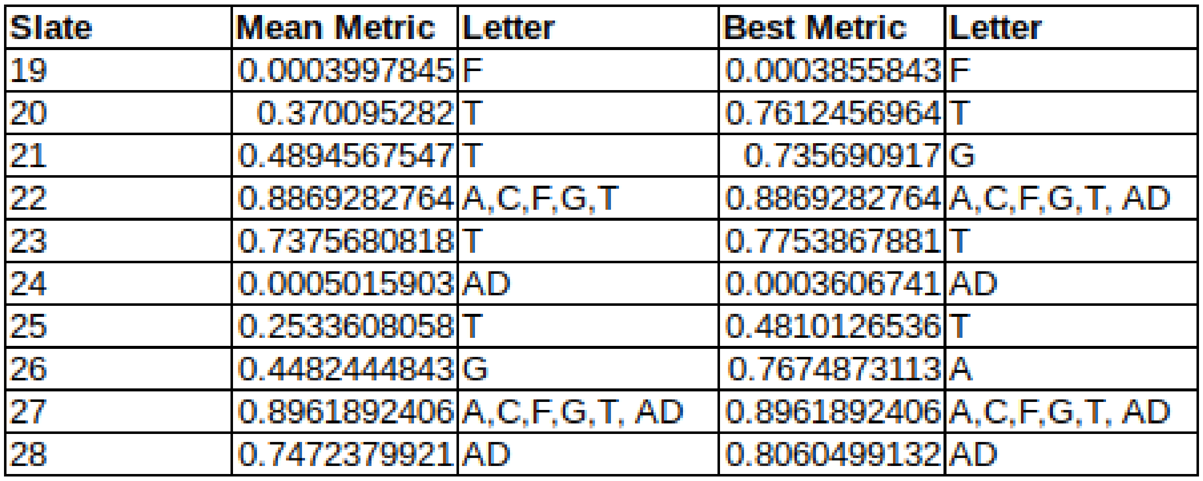
It is entirely possible that the models are generally too deep (even at their shallowest) to capture information about model depth versus performance. For these experiments, however, there does not appear to be benefits in using AE models over monolithic models.

Without statistically significant differences in model performance, there could be advantages to using AE models if the top-level prediction tasks could use smaller models than monolithic ones if reprocessed this way. Essentially, the prediction tasks would be sharing features discovered by unsupervised learning in one phase of processing, with ideally less processing afterward. However, since the top-level tasks did not consistently need shallower models, then a model using an AE is likely going to be slightly larger than a similar monolithic network.

The only potential advantage of an AE model from this set of experiments could be found in the potential of unsupervised learning. It could be the case that the top-level task data is available for less time or samples than the input data, in which case it would be worth investigating if an LSTM using an AE that trained on unlabeled data for a longer period of time would outperform a monolithic LSTM that had access to the labeled data for a shorter period.

To test that, 10 additional slates of experiments were run. Slates 19-23 ran only the regression and 4 prediction LSTM tasks on top of Slate 1's autoencoder models with 30 seconds of input, while Slates 24-28 ran the same on top of Slate 10's autoencoder models with 300 seconds of input. For these models, the training set was limited to only 30 percent of its size (the autoencoders used, of course, had the entire training set). The results of this small run are displayed in Figure 89.

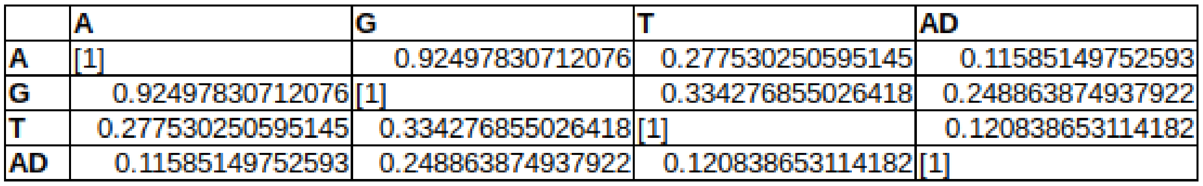
The slate results were very similar to the initial model results. In general, AE models outperformed based models. A Wilcoxon signed rank test was performed on results from



Slate	Mean Metric	Letter	Best Metric	Letter
19	0.0003997845	F	0.0003855843	F
20	0.370095282	T	0.7612456964	T
21	0.4894567547	T	0.735690917	G
22	0.8869282764	A,C,F,G,T	0.8869282764	A,C,F,G,T, AD
23	0.7375680818	T	0.7753867881	T
24	0.0005015903	AD	0.0003606741	AD
25	0.2533608058	T	0.4810126536	T
26	0.4482444843	G	0.7674873113	A
27	0.8961892406	A,C,F,G,T, AD	0.8961892406	A,C,F,G,T, AD
28	0.7472379921	AD	0.8060499132	AD

Figure 89: Results of rerunning LSTM tasks on smaller training dataset, with same autoencoders models from Slates 1 and 10.

Slates 18-21 and Slates 24-26 (The slates the predicted glucose risk were removed. It can be seen that several models arrived at the same metric for this task. By eliminating two thirds of the dataset, the dataset had mostly positive glucose risk cases, which led to models that mostly only predicted the glucose risk). This is displayed in Figure 90.



	A	G	T	AD
A	[1]	0.92497830712076	0.277530250595145	0.11585149752593
G	0.92497830712076	[1]	0.334276855026418	0.248863874937922
T	0.277530250595145	0.334276855026418	[1]	0.120838653114182
AD	0.11585149752593	0.248863874937922	0.120838653114182	[1]

Figure 90: Results of rerunning LSTM tasks on smaller training dataset, with same autoencoders models from Slates 1 and 10.

There were no statistically significant differences found between the base Network 1 model and any of the AE models for this experimental re-run. It is unlikely in this case that an AE model poses an advantage over a base model for neural networks of this type and form.

Discussion on VitalDB Dataset Deep Learning Models with Regard to Research Questions

In this section, the specific research questions posed for this dissertation are related back to the broader research questions being explored.

How modular can we make a multivariate system? For VitalDB, modularity did not prove to be a huge advantage. While modular models G and T had generally good performance, they did not have statistically significant differences from the monolithic models. They also did not perform similar at shallower depths, meaning performant modular models are likely to be as large or larger than the monolithic models. Retraining for added datastreams or tasks would likely take just as long or longer as using an end-to-end neural network model.

What is the best method of incorporating new data into an existing neural network design? In these VitalDB experiments, retraining the modular models for new datastreams did consistently hurt their performance. However, ultimately these experiments did not find a more performant or faster way of incorporating new data into a neural network system than training an entirely new end-to-end model.

Can we delegate multivariate (multiple types of input data) decision tasks to exist on top of an existing feature set? This research question proved to be true for VitalDB models as well as the Jornada models. The prediction tasks for VitalDB models fared no worse (and usually fared better) than the monolithic models, although all non-monolithic prediction tasks shared a feature set with the other prediction tasks. However, not all prediction tasks worked especially well. For Slates 1-5, the emergency operation and glucose risk predictions tasks worked well, while for Slates 10-14, all prediction tasks except for the combination task found acceptable models.

It is interesting that the combination task fared so poorly. This supports the idea of small top-level and task-specific models instead of combination tasks, as the one task that combined multiple others never received an F1 statistic above 0.7. However, in this case, the AE shared feature set did not pose an advantage over task-specific monolithic neural networks.

VitalDB Deep Learning Final Thoughts

In the Jornada Basin Dataset models, models that shared location-specific feature sets in some combination generally provided a slight advantage in terms of shared depth than monolithic models. VitalDB models in these experiments had only the option of sharing datastream-specific features sets and did not see that same advantage.

Jornada Basin did not see advantages in datastream-specific shared feature sets and so perhaps it is unsurprising that VitalDB did not have this property either. However, it should be noted that VitalDB had several differences from Jornada: it's time-sequenced data happened on the order of seconds rather than days, the environment for VitalDB is highly dynamic, and the domain is completely different. Due to time constraints, less models were also run on VitalDB with different hyperparameters. It's possible that the hyperparameter choice might heavily impact the model performance.

However, VitalDB models did perform – achieving acceptable F1 scores for the best prediction models, although any given model run had quite a bit of variation compared to Jornada Basin. More research should be conducted to note advantages of site-specificity compared to datastream-specificity to determine if one is generally better than or another, or if performance between the two is domain-dependent.

VitalDB Deep Learning Models Results Summary

In this set of results,

- Prediction tasks performance was highly variable across experimental runs;
- Processing all data together performed better than processing datastreams separately;
- In general, modular networks performed slighly better than non-modular networks, but;

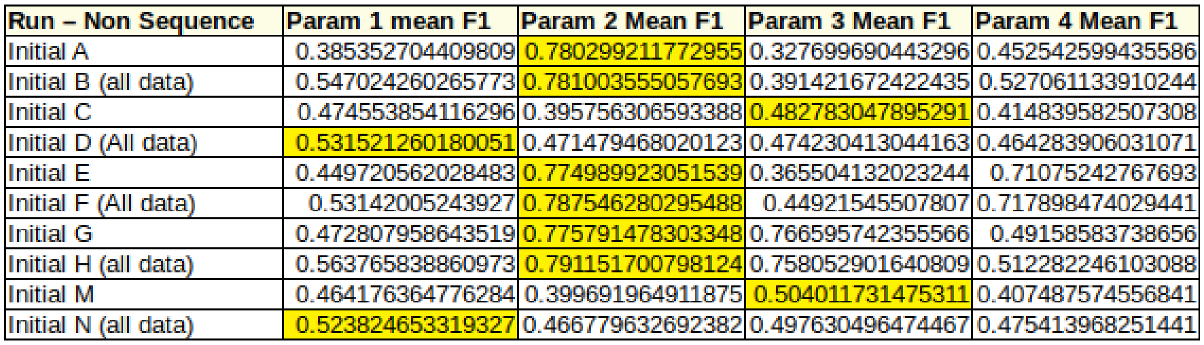
- No statistically significant difference in network distributions suggests no advantage in using a modular network over a monolithic one for this dataset.

6.3 Jornada - QARM-GA

This subsection presents and discusses the results of the Quantitative Association Rules with Genetic Algorithms.

Slate Metrics

The average F1 score across the best F1 score per run for all the parameters in the non-sequence models is presented in Figure 91 and these same metrics for the sequence models are presented in Figure 92.



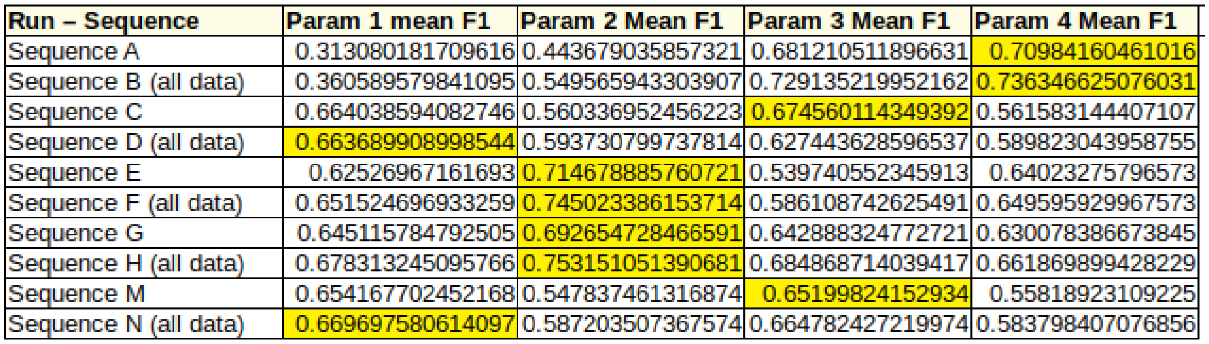
Run – Non Sequence	Param 1 mean F1	Param 2 Mean F1	Param 3 Mean F1	Param 4 Mean F1
Initial A	0.385352704409809	0.780299211772955	0.327699690443296	0.452542599435586
Initial B (all data)	0.547024260265773	0.781003555057693	0.391421672422435	0.527061133910244
Initial C	0.474553854116296	0.395756306593388	0.482783047895291	0.414839582507308
Initial D (All data)	0.531521260180051	0.471479468020123	0.474230413044163	0.464283906031071
Initial E	0.449720562028483	0.774989923051539	0.365504132023244	0.71075242767693
Initial F (All data)	0.53142005243927	0.787546280295488	0.44921545507807	0.717898474029441
Initial G	0.472807958643519	0.775791478303348	0.766595742355566	0.49158583738656
Initial H (all data)	0.563765838860973	0.791151700798124	0.758052901640809	0.512282246103088
Initial M	0.464176364776284	0.399691964911875	0.504011731475311	0.407487574556841
Initial N (all data)	0.523824653319327	0.466779632692382	0.497630496474467	0.475413968251441

Figure 91: Average F1 score across all sites for average of best rules per run of the parameters for non-sequence models.

The overall mean best rules across runs was found in the non-sequence models for Model H in parameter set 3. The sequence models, this was also found in Model H for parameter set 2.

If the best rule out of each run for a given parameter was taken instead of averaged, the best F1 for all sites for non-sequence models is given in Figure 93 for non-sequence models and Figure 94 for sequence models.

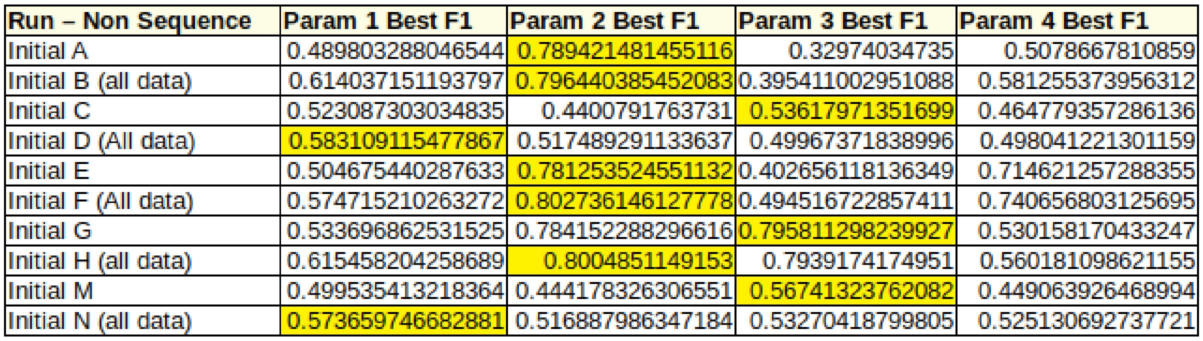
The best F1 score across the sites for a given parameter was found in Slate letter F in parameter 2 in non-sequence models, and for the sequence models, it occurred in



Run – Sequence	Param 1 mean F1	Param 2 Mean F1	Param 3 Mean F1	Param 4 Mean F1
Sequence A	0.313080181709616	0.443679035857321	0.681210511896631	0.70984160461016
Sequence B (all data)	0.360589579841095	0.549565943303907	0.729135219952162	0.736346625076031
Sequence C	0.664038594082746	0.560336952456223	0.674560114349392	0.561583144407107
Sequence D (all data)	0.663689908998544	0.593730799737814	0.627443628596537	0.589823043958755
Sequence E	0.62526967161693	0.714678885760721	0.539740552345913	0.64023275796573
Sequence F (all data)	0.651524696933259	0.745023386153714	0.586108742625491	0.649595929967573
Sequence G	0.645115784792505	0.692654728466591	0.642888324772721	0.630078386673845
Sequence H (all data)	0.678313245095766	0.753151051390681	0.684868714039417	0.661869899428229
Sequence M	0.654167702452168	0.547837461316874	0.65199824152934	0.55818923109225
Sequence N (all data)	0.669697580614097	0.587203507367574	0.664782427219974	0.583798407076856

Figure 92: Average F1 score across all sites for average of best rules per run of the parameters for sequence models.

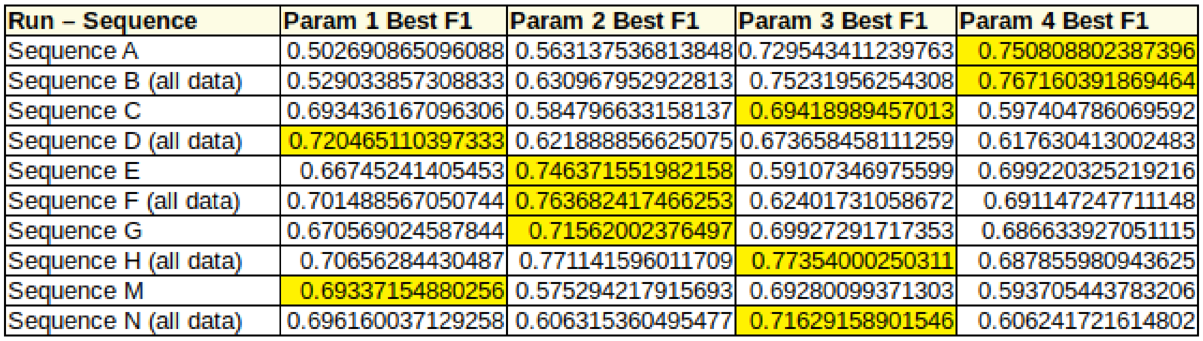
.



Run – Non Sequence	Param 1 Best F1	Param 2 Best F1	Param 3 Best F1	Param 4 Best F1
Initial A	0.489803288046544	0.789421481455116	0.32974034735	0.5078667810859
Initial B (all data)	0.614037151193797	0.796440385452083	0.395411002951088	0.581255373956312
Initial C	0.523087303034835	0.4400791763731	0.53617971351699	0.464779357286136
Initial D (All data)	0.583109115477867	0.517489291133637	0.49967371838996	0.498041221301159
Initial E	0.504675440287633	0.781253524551132	0.402656118136349	0.714621257288355
Initial F (All data)	0.574715210263272	0.802736146127778	0.494516722857411	0.740656803125695
Initial G	0.533696862531525	0.784152288296616	0.795811298239927	0.530158170433247
Initial H (all data)	0.615458204258689	0.8004851149153	0.7939174174951	0.560181098621155
Initial M	0.499535413218364	0.444178326306551	0.56741323762082	0.449063926468994
Initial N (all data)	0.573659746682881	0.516887986347184	0.53270418799805	0.525130692737721

Figure 93: Average F1 score across all sites for best rule per run of the parameters for non-sequence models.

.



Run – Sequence	Param 1 Best F1	Param 2 Best F1	Param 3 Best F1	Param 4 Best F1
Sequence A	0.502690865096088	0.563137536813848	0.729543411239763	0.750808802387396
Sequence B (all data)	0.529033857308833	0.630967952922813	0.75231956254308	0.767160391869464
Sequence C	0.693436167096306	0.584796633158137	0.69418989457013	0.597404786069592
Sequence D (all data)	0.720465110397333	0.621888856625075	0.673658458111259	0.617630413002483
Sequence E	0.66745241405453	0.746371551982158	0.59107346975599	0.699220325219216
Sequence F (all data)	0.701488567050744	0.763682417466253	0.62401731058672	0.691147247711148
Sequence G	0.670569024587844	0.71562002376497	0.69927291717353	0.686633927051115
Sequence H (all data)	0.70656284430487	0.771141596011709	0.77354000250311	0.687855980943625
Sequence M	0.69337154880256	0.575294217915693	0.69280099371303	0.593705443783206
Sequence N (all data)	0.696160037129258	0.606315360495477	0.71629158901546	0.606241721614802

Figure 94: Average F1 score across all sites for best rule per run of the parameters for sequence models.

slate letter H in parameter 3. In general, models worked best with fitness function 2 and reseeding from the best rules.

Interestingly, for both the sequence and non-sequence models, having other site data incorporated into the site specific data was helpful and generally had better results than using only site specific data. For both models, most of the mined rules for a given site incorporated weather from a different site.

Samples rules for some of the sites are given in Figures 95 and 96 for non-sequence and sequence runs, respectively.

In Figure 95, the NPP c_grav average air temperature was the best frost predictor for the NPP c_cali site: if the average air temperature was between roughly -2 and 7, on a given day, there was likely to be frost the next day. For Figure 96, the NPP t_west average air temperature was the best predictor for the NPP p_smal site. If the average air temp had been between 4 and 12 degrees anytime in the previous 9 days, a frost event could be predicted.

Discussion on Jornada Basin Dataset QARM-GA Models with Regard to Research Questions

This section looks at the performance of the QARM GA predictors specifically in relation to the research questions.

What is the performance of a QARM GA predictor compared to a neural network predictor? Overall, the QARM GA predictor was not as powerful as a neural network predictor, but it wasn't too far behind. The best neural network F1 score was 0.847, while the best for a QARM GA was 0.803. Considering the QARM GA predictor is a small series of value comparisons instead of high-parameter function evaluation, this is surprisingly close. In addition, the results for the QARM GA models are highly interpretable and potentially more insightful, as the best rules captured statistically interesting interactions between unrelated sites when it came to predicting frost.

```
"npp_c_cali": [
{
    "parameters": {
        "npp_c_gravAir_TempC_Avg": {
            "lower_bound": -2.943737408868765,
            "upper_bound": 6.914454394126818
    }
},
    "support": 0.06873879258816497,
    "confidence": 0.7516339869281046,
    "lift": 7.310951512387901,
    "fitness": 0.7134103174657996
}
],
"npp_c_grav": [
{
    "parameters": {
        "npp_m_rabbAir_TempC_Avg": {
            "lower_bound": -2.6047473295618273,
            "upper_bound": 6.341436730109321
    }
},
    "support": 0.09802749551703527,
    "confidence": 0.7735849056603774,
    "lift": 5.701354833347187,
    "fitness": 0.8709397005574128
}
]
```

Figure 95: Sample site rules for non-sequence runs for Slate letter H.

```
"npp_m_well": [
{
    "parameters": {
        "npp_p_smalAir_TempC_Min": {
            "lower_bound": -9.227515495969113,
            "upper_bound": -5.425585853230013,
            "seq_lower_bound": 0,
            "seq_upper_bound": 9
}},
    "support": 0.1972504482964734,
    "confidence": 0.7051282051282052,
    "lift": 3.080103099685345,
    "fitness": 0.9585987087181179
}],
"npp_p_coll": [
{
    "parameters": {
        "npp_m_nortAir_TempC_Avg": {
            "lower_bound": -2.677282997019196,
            "upper_bound": 13.56464798653699,
            "seq_lower_bound": 0,
            "seq_upper_bound": 1
}},
    "support": 0.2355050806933652,
    "confidence": 0.6512396694214876,
    "lift": 2.6638727798096546,
    "fitness": 1.4516494496692345
}]
```

Figure 96: Sample site rules for sequence runs for Slate letter H.

How modular can we make a QARM GA? As with the Jornada models, models ran best when they had all site data rather than a subset. Site-specific models ran more quickly and were not far behind in performance, however, and may be easier to implement. It was interesting that almost all the top rules in models that used all data relied on features from a different site than the one being predicted. This could be simply because it was difficult to mine the exact same site parameter and using a different one regularized the algorithm's analysis somewhat. However, it may also be the case that the Jornada sites are sufficiently spread out that one site gets weather events slightly earlier than the next – or one site's past weather events affects the next site's more than its own.

For time-variable datasets, is there an advantage to using a sequence extension on QARM? For these slates of experiments, the non-sequence models had slightly higher performance for a given high-performing set of parameters than sequence models did. However, in averaging the best parameter runs across all the slates, the sequence models had a slightly higher average performance than the non-sequence models. While it makes sense that the day prior would be most important in predicting frost, the sequence models seemed slightly more robust to algorithm runs with less optimal parameters – essentially, the sequence models tended to still be able to find performant rules when parameters did perform as well overall compared to the non-sequence models. This makes some sense – the sequence models were able to look back at longer and more dynamic time intervals (the best rules for model runs F and H used 0 to 11 days in the past, depending on the rule), which is probably more stable than relying on a single past day.

Jornada Basin QARM-GA Results Summary

For the Jornada QARM-GA results:

- Using all data together generally performed better than using site-specific models.
- However, site-specific models were not far behind.

- QARM-GA models did not perform quite as well as neural network models.
- However, QARM-GA models are highly explainable and very easy to implement in real-time.
- Non-sequence models performed slightly better than sequence models.
- However, sequence models appeared to be more robust to sub-optimal parameters.

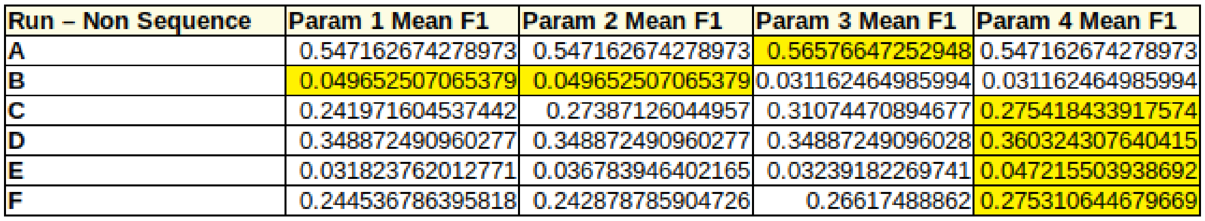
6.4 VitalDB - QARM-GA

This subsection presents the results of the QARM GA models for the VitalDB dataset on the glucose risk, emergency operation, and discharge mortality risk prediction tasks.

Slate Metrics

This subsection presents and discusses the results of the QARM GA models with regard to the VitalDB dataset.

The best rule for each parameter was taken from each run and then averaged together to give the average best results of the F1 statistic for non-sequence models in Figure 97: and for sequence models in Figure 98.

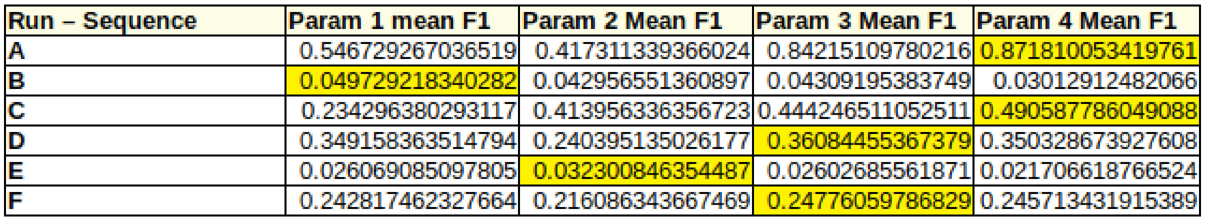


Run – Non Sequence	Param 1 Mean F1	Param 2 Mean F1	Param 3 Mean F1	Param 4 Mean F1
A	0.547162674278973	0.547162674278973	0.56576647252948	0.547162674278973
B	0.049652507065379	0.049652507065379	0.031162464985994	0.031162464985994
C	0.241971604537442	0.27387126044957	0.31074470894677	0.275418433917574
D	0.348872490960277	0.348872490960277	0.34887249096028	0.360324307640415
E	0.031823762012771	0.036783946402165	0.03239182269741	0.047215503938692
F	0.244536786395818	0.242878785904726	0.26617488862	0.275310644679669

Figure 97: Average best F1 score across runs for each parameter set for the vital non-sequence models.

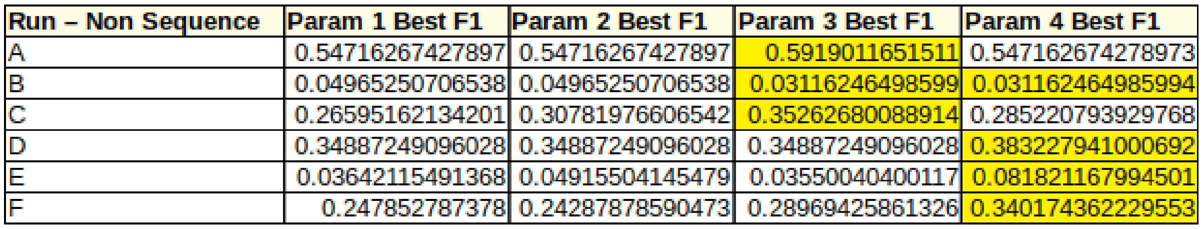
The average best rule for non-sequence models was found in Slate letter A, parameter set 3. For sequence models, this was Slate letter A, parameter set 4.

The overall best rule F1 score for a given parameter for non-sequence models is given in Figure 99 and for sequences models is given in Figure 100.



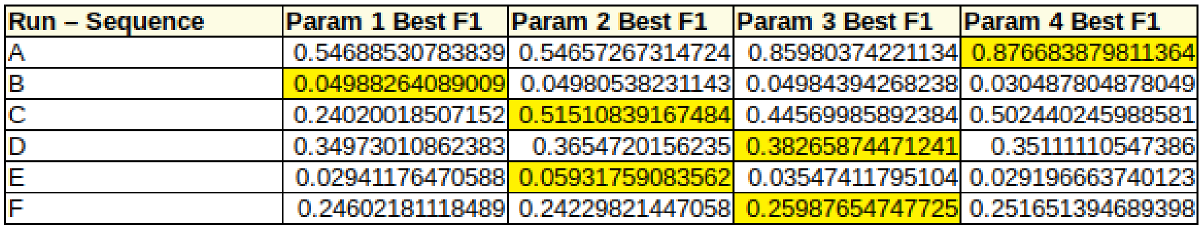
Run – Sequence	Param 1 mean F1	Param 2 Mean F1	Param 3 Mean F1	Param 4 Mean F1
A	0.546729267036519	0.417311339366024	0.84215109780216	0.871810053419761
B	0.049729218340282	0.042956551360897	0.04309195383749	0.03012912482066
C	0.234296380293117	0.413956336356723	0.444246511052511	0.490587786049088
D	0.349158363514794	0.240395135026177	0.36084455367379	0.350328673927608
E	0.026069085097805	0.032300846354487	0.02602685561871	0.021706618766524
F	0.242817462327664	0.216086343667469	0.24776059786829	0.245713431915389

Figure 98: Average best F1 score across runs for each parameter set for the vital sequence models.



Run – Non Sequence	Param 1 Best F1	Param 2 Best F1	Param 3 Best F1	Param 4 Best F1
A	0.54716267427897	0.54716267427897	0.5919011651511	0.547162674278973
B	0.04965250706538	0.04965250706538	0.03116246498599	0.031162464985994
C	0.26595162134201	0.30781976606542	0.35262680088914	0.285220793929768
D	0.34887249096028	0.34887249096028	0.34887249096028	0.383227941000692
E	0.03642115491368	0.04915504145479	0.03550040400117	0.081821167994501
F	0.247852787378	0.24287878590473	0.28969425861326	0.340174362229553

Figure 99: Best F1 score across runs for each parameter set for the vital non-sequence models.



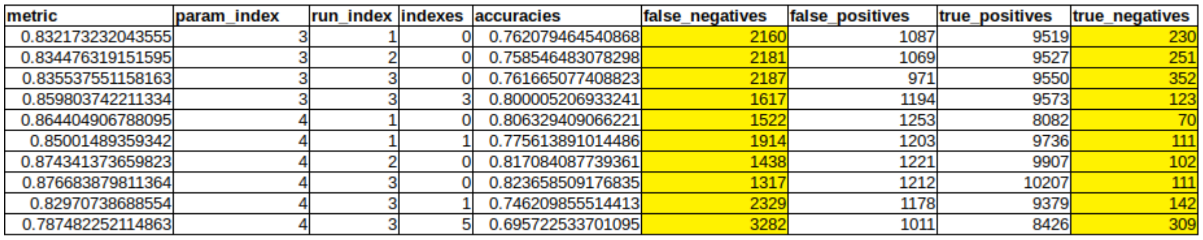
Run – Sequence	Param 1 Best F1	Param 2 Best F1	Param 3 Best F1	Param 4 Best F1
A	0.54688530783839	0.54657267314724	0.85980374221134	0.876683879811364
B	0.04988264089009	0.04980538231143	0.04984394268238	0.030487804878049
C	0.24020018507152	0.51510839167484	0.44569985892384	0.502440245988581
D	0.34973010862383	0.3654720156235	0.38265874471241	0.35111110547386
E	0.02941176470588	0.05931759083562	0.03547411795104	0.029196663740123
F	0.24602181118489	0.24229821447058	0.25987654747725	0.251651394689398

Figure 100: Best F1 score across runs for each parameter set for the vital sequence models.

The overall best rule was found in Slate A parameter set 3 for non-sequence models, and Slate A parameter set 4 for sequence models.

Only Slate letter A, which predicted the glucose risk factor for the VitalDB cases, had performance above 0.7 on the F1 score. While the sequence models slightly outperformed the non-sequence models, both performed very poorly except for the glucose risk. Even the glucose risk factor model only performed well for a specific set of parameters.

However, looking more closely at the glucose risk models in Slate A, the models are not quite as useful as the relatively high F1 score would make them out to be. Figure 101 displays the accuracies, metrics, and true and false positives and negatives for the top 10 rules in the sequence.



metric	param_index	run_index	indexes	accuracies	false_negatives	false_positives	true_positives	true_negatives
0.832173232043555	3	1	0	0.762079464540868	2160	1087	9519	230
0.834476319151595	3	2	0	0.758546483078298	2181	1069	9527	251
0.835537551158163	3	3	0	0.761665077408823	2187	971	9550	352
0.859803742211334	3	3	3	0.800005206933241	1617	1194	9573	123
0.864404906788095	4	1	0	0.806329409066221	1522	1253	8082	70
0.85001489359342	4	1	1	0.775613891014486	1914	1203	9736	111
0.874341373659823	4	2	0	0.817084087739361	1438	1221	9907	102
0.876683879811364	4	3	0	0.823658509176835	1317	1212	10207	111
0.82970738688554	4	3	1	0.746209855514413	2329	1178	9379	142
0.787482252114863	4	3	5	0.695722533701095	3282	1011	8426	309

Figure 101: Top 10 overall rules by fitness for Vital Sequence A.

The false and true negatives columns are highlighted in Figure 101. It is immediately evident that the test set for the glucose risk task is highly bent towards positive cases. There are comparatively few negative cases. Most of the rules mined for the glucose risk task skewed towards predicting mostly positive cases, and there were more false positive predictions than true negatives. This is a major limitation on the usefulness of any of the predicted rules despite the high F1 metric. For even the one model (Sequence A) that saw an F1 score above 0.7, the mined rules are not useful predictors.

This domain, however, is slightly different from the Jornada Basin Dataset in that the usefulness of the algorithm might not be limited to prediction models. For this medical dataset, having rules with decent support and decent lift might indicate correlated features with patient outcomes, even if they aren't strong predictors in themselves. Finding correlated features may help healthcare providers investigate vital sign interactions and/or help identify useful patient status metrics.

For this reason, rules with lift at or above 1.6 and support at or above 0.01 (1 % of the data) were taken from each sequence and scrutinized for logical relation to the related medical outcome (and, due to this researcher's lack of knowledge in the medical domain, discussed with medical doctors). Investigation into individual rules was conducted for Sequence models B and C, for the emergency operation and discharge risk prediction tasks (as the glucose risk task did not come up with rules that appeared to be highly statistically significant). The sequence and non-sequence models often came up with similar rules, with slightly different values or similar but not quite the same measurement parameters.

For Sequence B, (which predicted whether or not an operation was an emergency operation), rules are displayed in Figures 102 through 106.

```
{
"Orchestra_RFTN20_CT": {
    "lower_bound": 0.0,
    "upper_bound": 0.06108558775989881,
    "seq_lower_bound": 2,
    "seq_upper_bound": 2
    }
},
"support": 0.018364245564567807,
"confidence": 0.9269870609981515,
"lift": 6.2521321448796146,
"fitness": 0.02556893065043826
}
```

Figure 102: Rule for Vital Sequence model B for Orchestra_RFTN20_CT data feature.

```
{
"Orchestra_RFTN20_CE": {
    "lower_bound": 0.006904498730275301,
    "upper_bound": 0.3181121957791861,
    "seq_lower_bound": 5,
    "seq_upper_bound": 5
}
},
"support": 0.015775474640007336,
"confidence": 0.9976798143851509,
"lift": 6.728924599118398,
"fitness": 0.018493499817958347
}
```

Figure 103: Rule for Vital Sequence model B for Orchestra_RFTN20_CE data feature.

```
{
"Orchestra_RFTN20_CP": {
    "lower_bound": 0.0,
    "upper_bound": 0.2347718503013766,
    "seq_lower_bound": 10,
    "seq_upper_bound": 10
}
},
"support": 0.015548808538043978,
"confidence": 0.9135135135135135,
"lift": 6.1612588167842235,
"fitness": 0.016695474359485322
}
```

Figure 104: Rule for Vital Sequence model B for Orchestra_RFTN20_CP data feature.

```
{
"Orchestra_RFTN20_RATE": {
"lower_bound": 0.0,
"upper_bound": 3.091700112845868,
"seq_lower_bound": 7,
"seq_upper_bound": 7
}
},
"support": 0.03355556779987878,
"confidence": 0.3837429111531191,
"lift": 2.588182177652728,
"fitness": 0.01476182816528856
},
```

Figure 105: Rule for Vital Sequence model B for Orchestra_RFTN20_RATE data feature.

```
{
"Solar8000_VENT_MAWP": {
    "lower_bound": 6.901997556628691,
    "upper_bound": 7.191949062119614,
    "seq_lower_bound": 0,
    "seq_upper_bound": 0
}
},
"support": 0.02797842187071409,
"confidence": 0.9244712990936556,
"lift": 6.235164404407566,
"fitness": 0.016637439378973315
}
```

Figure 106: Rule for Vital Sequence model B for Solar8000_VENT_MAWP data feature.

Four of the five rules found dealt with the infusion pump. The Orchestra infusion pump delivered remifentinal (a pain medication) during anesthesia to the patients in these cases. Figure 107 and Figure 108 give the values for specific features of the pump for the training set cases for emergency operation and non-emergency operation patients, respectively.

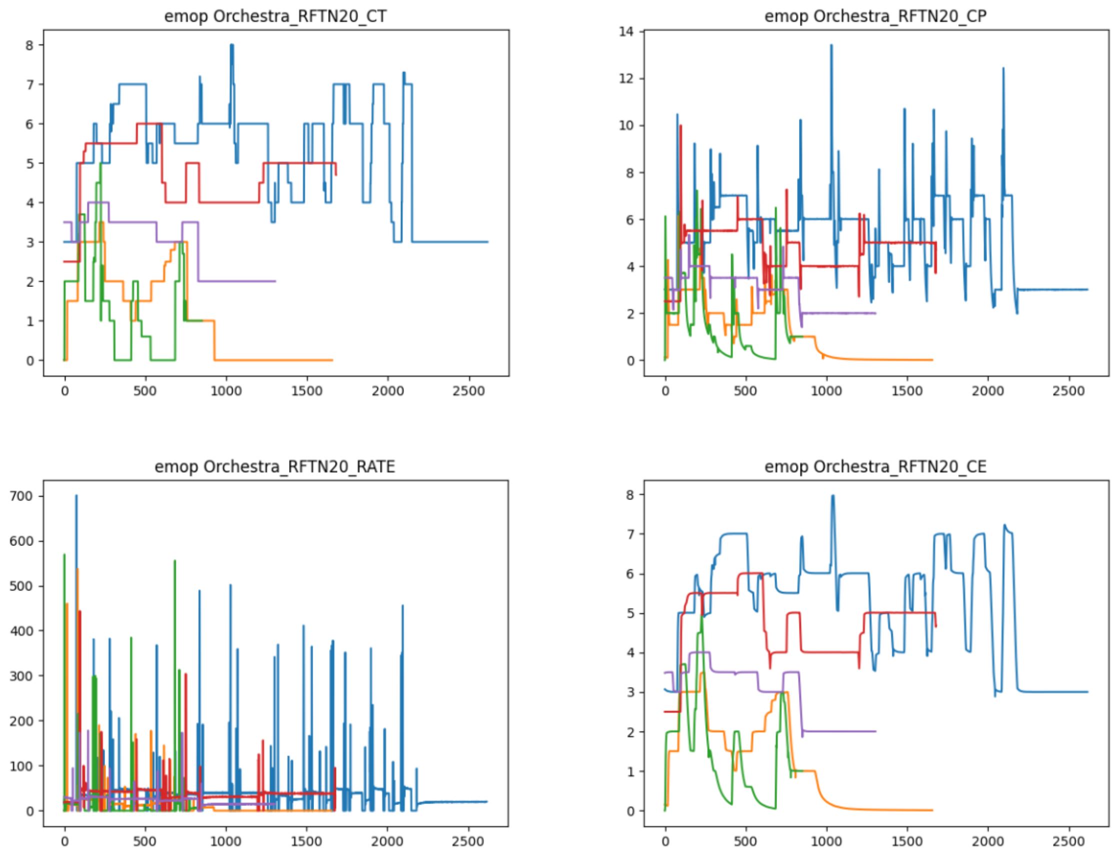


Figure 107: Graphing values from orchestra infusion pump for training set for emergency operation cases.

It can be seen that anesthesia rates and targets trend slightly lower to begin with for emergency operation patients than for non-emergency operation patients. It was explained to this researcher that this is not unexpected: an anesthesiologist will try to meet certain target values for a planned patient surgery generally before the surgery begins. For an emergency operation, the levels might not be fully met before the surgery

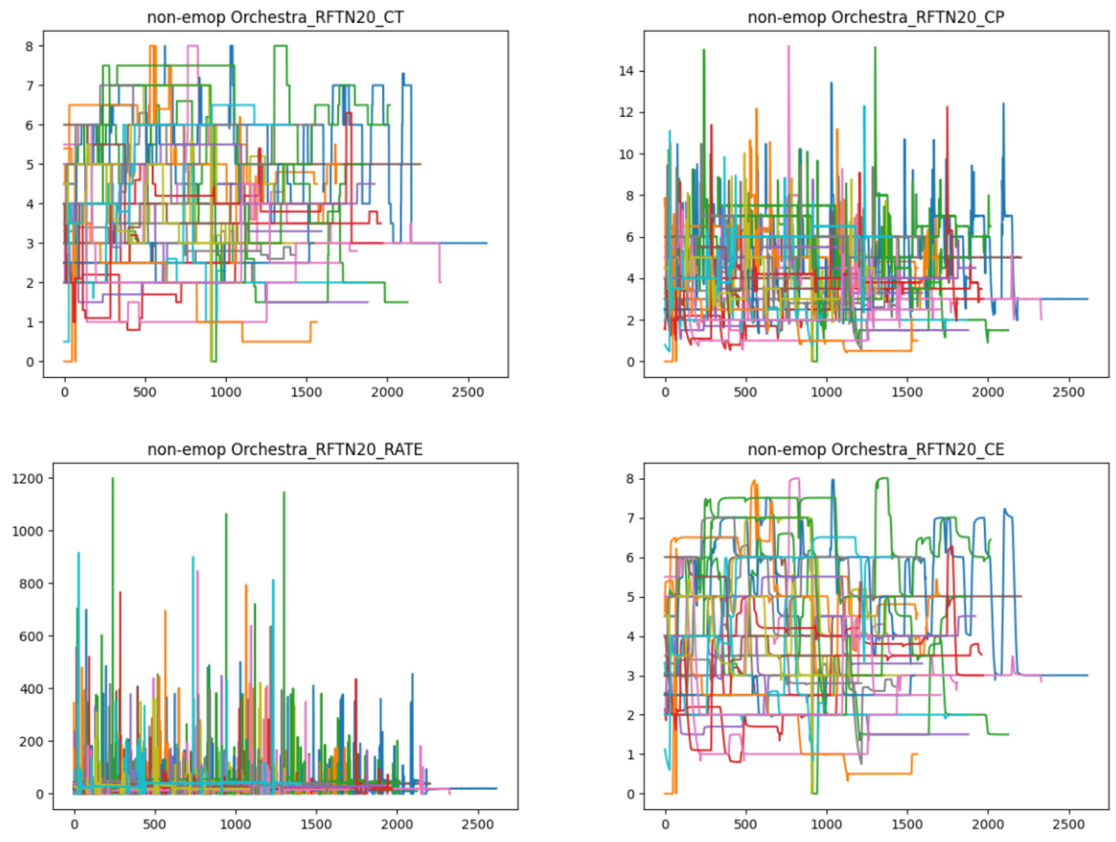


Figure 108: Graphing values from orchestra infusion pump for training set for non-emergency operation cases.

begins due to the time sensitive nature of the operation [167]. The dataset portion used for these experiments was cut in time to the seconds where most if not all the machines had started recording. For an emergency operation, it then makes sense that the other patients monitors would be recording while anesthesia levels are low if the surgery timeframe has been compressed.

Perhaps more interesting is the ventilator information from Solar8000_VENT_MAWP datastream, which measures mean airway pressure on the patient's lungs. Figure 109 displays the mean airway pressure of emergency operation patients compared to non-emergency operation patients.

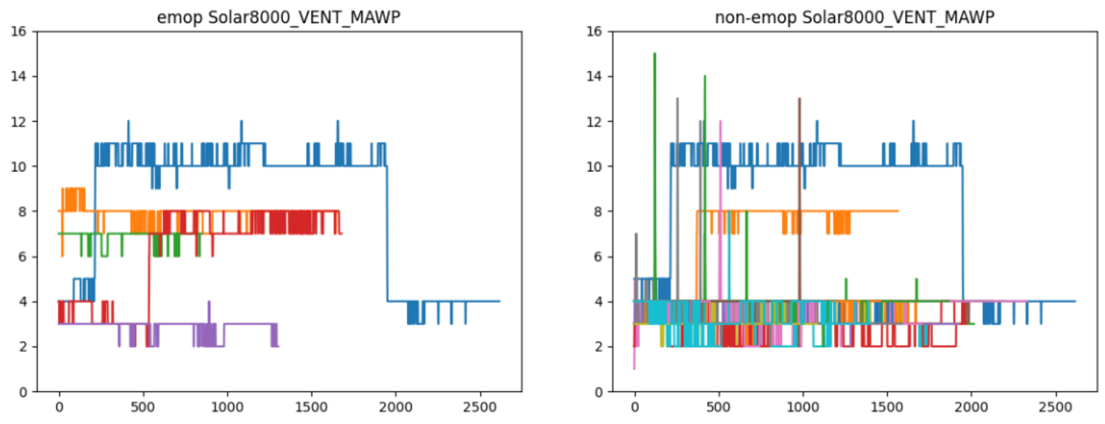


Figure 109: Mean airway pressure values for emergency operation cases compared to non-emergency operation cases.

Mean airway pressure gives an approximate measurements of lung stiffness (the pressure that must be overcome to ventilate the patient) [168]. The emergency operation patients in this set had slightly higher mean airway pressure values on average, and this rule mined a range not seen as often in the non-emergency operation patients. A 2021 study in the Journal of Intensive Medicine suggested mean airway pressure as a singular and highly useful metric correlated with negative patient outcomes [168] but noted that the metric needed further validation. The discovery of this metric in higher ranges by the QARM GA being associated with emergency operations is promising.

Sequence C predicted whether the hospital discharge time met or exceeded 14 days. It some similar infusion pump rules as the emergency operation task (unsurprising, as three of the five emergency operation cases also had the extended discharge time), and two additional types of rules, which are outlined in Figures 110 and 111.

```
{
"Solar8000_VENT_INSP_TM": {
    "lower_bound": 1.0968162494645863,
    "upper_bound": 1.1110976640944852,
    "seq_lower_bound": 1,
    "seq_upper_bound": 1
}
},
"support": 0.0765219300653236,
"confidence": 0.9258357316803187,
"lift": 1.7335248574586806,
"fitness": 0.021098401251895667
}
```

Figure 110: Rule for Vital Sequence model C for Solar8000_VENT_INSP_TM data feature.

The inspiration time in both rules notes a very small interval of values. Of more interest is the SNUADC_FEM data feature, which looks at the femoral arterial pressure wave from a patient.

This rule states that the inspiration time between 1.07 and 1.12 seconds (between 2 and 2.5 minutes prior to the event) and a femoral arterial pressure wave value between 95.46 and 104.01 mmHg (between 2 and 1.5 minutes before the event) typically see an event of an extended hospital stay. The support value of 0.011 notes that this correlation was found in 1.1% of the data, while the confidence notes that every time this rule occurred, the extended hospital stay event also occurred in the data 92.5% of the time. The lift notes that this is above 1.73 times more likely than expected.

```
{
"SNUADC_FEM": {
    "lower_bound": 95.46352258693517,
    "upper_bound": 104.00994272738527,
    "seq_lower_bound": 2,
    "seq_upper_bound": 3
},
"Solar8000_VENT_INSP_TM": {
    "lower_bound": 1.0729557488679515,
    "upper_bound": 1.1191693738796773,
    "seq_lower_bound": 3,
    "seq_upper_bound": 4
}
},
"support": 0.011267244380118353,
"confidence": 0.924812030075188,
"lift": 1.7316080895932908,
"fitness": -0.34785002214893773
}
```

Figure 111: Rule for Vital Sequence model C for SNUADC_FEM and Solar8000_VENT_INSP_TM data feature.

Figure 112 looks at a box plot the femoral arterial pressure wave values (only positive) for patients with an extended hospital stay versus patients who did not stay in the hospital as long.

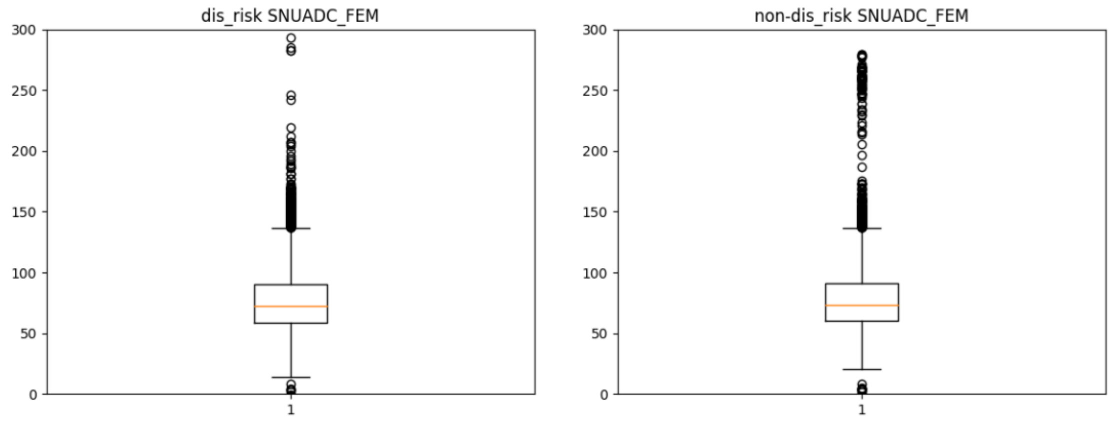


Figure 112: Femoral arterial pressure wave positive values for patients with extended hospital stay (dis risk) and non-extended hospital stay (non dis risk)

The rule in Figure 111 involving the arterial pressure wave (SNUADC_FEM and Solar800_VENT_INSP_TIME) noted values between 95 and 104 mV often lead to an extended hospital stay. These values were in the slightly higher end of the normal range from the cases, and could perhaps indicate less cardiac control during the surgery.

Discussion on VitalDB Models with Regard to Research Questions

This section looks at the performance of the QARM GA predictors specifically in relation to the research questions. The modularity question was left out as it was not applicable to this dataset since data is case rather than site specific.

What is the performance of a QARM GA predictor compared to a neural network predictor? For this dataset, the QARM-GA did not produce useful results for predicting the values set out in the prediction tasks. Even the model with the best metric did not create reasonable prediction output. The QARM GA was less useful than the neural network for this task. However, the resampling of the data to 30 seconds only may have been a problematic data wrangling choice.

For time-variable datasets, is there an advantage to using a sequence extension on QARM? Since no experiments predicted the tasks successfully for this dataset, it was difficult to evaluated whether or not the sequence or non-sequence version was preferable, especially since one sequence step was equivalent to 30 seconds of input in this case. This resampling occurred due to the vast size of the dataset. Additionally, the sequence penalty was such that the algorithm generally mined very short sequence intervals, making it difficult to determine a sequence benefit. Further research would be needed to see if there is an advantage of one technique over the other. However, since vital signs interactions are highly dependent on time, it is hypothesized that the sequence extension would be the best area of further research.

VitalDB QARM-GA Results Summary

For the VitalDB QARM-GA models:

- None of the algorithm runs produced useful prediction models.
- However, the algorithm mined an interesting correlation for emergency operations that aligns with another medical study as a patient outcome indicator.
- Further research is needed to determine if the sequence or non-sequence run of this algorithm would be more useful.

6.5 Results Summary

To summarize the experimental results, it can be seen that the Jornada Basin dataset was able to be successfully regressed and predicted by neural network models and by genetic algorithms, with neural networks producing slightly more accurate predictions and QARM GA models producing more explainable and less computationally-heavy ones. For both neural networks and the QARM-GA models, results were best when all data were processed together. For VitalDB models, performance was not quite as accurate, especially for prediction tasks, though neural networks fared better than the QARM-GA models. However, the QARM-GA models did find some interesting and sensible correlations of patient data with overall outcomes. The QARM-GA models used a slightly different sampling method than the neural network models, which may have decreased performance. The next section looks at the limitations of the experiment as well as potential directions of future work.

7 Study Limitations and Future Work

This section acknowledges the limitation of the experiments conducted in the dissertation particularly in the areas of datasets and model architectures. It also outlines several different directions of future work that could be pursued based on the results obtained.

7.1 Study Limitations

Overall, there are several limitations of the work presented in this dissertation, including limitations in the datasets and how they were used, in the deep learning model architectures, and in the QARM GA models.

Datasets

For both experiments, only two datasets were used: the Jornada Basin NPP weather station datasets and a subset of patient data from the VitalDB dataset. Neither dataset was collected firsthand for this research. For the Jornada Basin dataset, the prediction task was derived from one of the features provided to the models, which is a somewhat weaker metric than collecting observed frost data. For VitalDB, the data provided is likely much more case-specific (by virtue of being medical data) and the subset of cases chosen are not guaranteed to be representative of medical cases in general. For VitalDB, more missing data was found in the prediction models, and often repeated to fill gaps. VitalDB's predictors were also not balanced through the datasets, which may have thrown off results. Both datasets are very domain-specific, and results may not hold up for other domains. These datasets also had data with a simple structure; a more intensive data environment using images or videos may need separate techniques entirely. More intensive data streams would also likely need preprocessing before being added to the main model pipeline.

Deep Learning Models

The deep learning models in this dissertation used a very small subset of hyperparameters, and results may change drastically with other hyperparameter choices and network structures. While each deep learning model ran a few different network structures, the structures mainly varied in terms of numbers of nodes in the layer, not in terms of total layers. The experimental runs were often performant, but it would be difficult to determine if they were truly optimal. Additionally, the prediction tasks for all deep learning models were binary rather than categorical. Classification models with more than one objective of interest may not have similar results.

QARM GA Models

The QARM GA models also used a small subset of the potential possible hyperparameters, and other models with different hyperparameters are not guaranteed to fare the same. For these models, parameters like the mutation rate, mutation amount, number of rules to keep, the killing dominated rules remained the same throughout runs. However, mutation rates and amounts generally have a nontrivial impact on performance and dominated rule killing impacts the diversity of the rule population. The number of top rules impacts the overall performance in choosing only a subset of rules to evaluate. In reality, the fitness score of the individual was only a proxy measurement for prediction accuracy, and there may be more predictive rules in the population that did not make the top 10 list. Different hyperparameter choices could have led to very different models, whereas this set of experiments only looked at a subset of roughly 20 parameter combinations. Genetic algorithms in general are perhaps more sensitive to hyperparameter choice. Different parameter choices should be explored for overall impact to model performance.

7.2 Future Work

In this dissertation, several experiments were run to look at site-specific models versus data-specific models. Ultimately, models using some forms of site-specificity were more performant than the data-specific models for deep learning neural networks. However, the test cases for site vs data specificity were narrow in this dissertation, and the tradeoffs and/or performance differences between the two would likely be worth further study.

The domains for these experiments were also narrow, and further work should be conducted to see if the performance holds up for other uses. A planned future implementation of this work involves trying out several neural network types identified in this work on time sequenced weather data for a vineyard in Winchester, Virginia. Frost modeling with the QARM GA is also planned to be conducted at this location.

Another test site is the University of Idaho Sandpoint Organic Agriculture Center. In addition to weather parameters, one extension of this artificial intelligence research involves determining impact of light penetration to the apple tree canopy with regard to more optimal pruning techniques.

More intensive information types should also be looked at with regard to network structure. This dissertation was unable to look at spatial locality mapping in designing networks due to the unsuitability of the chosen datasets. However, this is very much an area of interest. To remedy this, a custom robotic dataset is slated for future work of the project. The University of Idaho owns four MyCobot320 Pi 6-axis robots, of which one will be used to generate this. The dataset will include motor encoder readings and images taken from a mounted camera on the robot, labeled with time of observation. The goal use of this dataset is to develop robot proprioception, allowing the robot to perform more complex tasks.

For the quantitative analysis rules with GAs, potential exists for application to domains who may not need prediction models as much as help investigating causation in event analysis. Two domains particularly could be included for this application: environmental science and manufacturing defects.

The sequence extension in this paper seeks to look at correlated factors in the same or different time steps with regard to observed phenomena. For environmental science, a lot of collected data is in time series, particularly regarding weather, flow, land use change, and more. QARM GA with sequence extension is a potential way for researchers to examine closely correlated factors in a vast search space to help identify areas of interest for causes. In manufacturing, process steps also often occur in sequence, especially in manufacturing areas that have some form of continuous flow. Identifying correlated factors to defects may assist in identifying defect causes or machine settings that may sub-optimal products more likely. The explainability of QARM GA may be helpful in find causes behind correlations outside of the normal prediction tasks.

Limitations and Future Work Summary

This section looked at a variety of limitations in the experimental setups and architectures, as well ways the research can be expanded in the future for both deep learning and QARM GA models and applications. The next section concludes the dissertation with high level discussion on the experimental results.

8 Conclusions

This dissertation explored two concepts in making artificial intelligence more usable to smaller organizations: reorganizing neural networks to modularize input data and exploring a non-neural network technique for prediction tasks with increased explainability. Neither technique was a complete solution to making artificial intelligence an easy implementation for a small organization. However, both exhibited properties that may aid in future construction and adoption of artificial intelligence systems that will be more accessible to smaller organizations. This section outlines final thoughts on the results of the experiments and their contributions to research in this area.

For the deep learning models, two organizational strategies were tested: maintaining multiple separate models or using combination models that used all datastreams and/or locations. Overwhelmingly, the combination models fared better for both the Jornada Basin and VitalDB datasets. This was unsurprising, given deep learning's ability to find interesting interactions between parameters that are useful for prediction. Ultimately, having more data features aided prediction tasks considerably.

For the deep learning models, modularizing data within a network structure was also tested by using autoencoders as shared feature sets for a given modality. For the Jornada Basin dataset, modalities were broken up by datastream, location, or both, and reassembled based on network types. For VitalDB, modalities were only broken up by datastream.

For the Jornada Basin dataset, the networks that broke up modalities by location in some form usually slightly outperformed monolithic networks for these experiments – though not in a statistically significant way. However, the shared feature sets of the experiments aid in reducing retraining time when adding locations to an overall system. The shared feature space models generally outperformed the monolithic models with less depth, and therefore, adding a new location could retrain a shared feature set once before retraining smaller top-level tasks, rather than completely retraining several large end-to-end tasks together. In this dataset, breaking up modalities by datastreams did not see the same advantages.

For the VitalDB dataset, breaking up modalities by datastream did not provide advantages over monolithic models, as was the case for Jornada. Since VitalDB did not have a sensical way of separating the data by location (case-based not location-based) the shared latent space and autoencoder models for VitalDB did not provide statistically significant improvements although they generally outperformed the monolithic models. VitalDB had a much less straightforward dataset and more difficult prediction tasks, which may have contributed to the lack of performance gain seen in this dataset. In these experiment sets, however, VitalDB did have a useful contribution: it demonstrated that trying to predict multiple tasks simultaneously fared worse than having one prediction model per task.

From a precision agriculture perspective, this set of experiments suggests more research into site-specific management artificial intelligence system construction. In the case where a grower has many high-level tasks to monitor (frost, pests, diseases, etc.) and plans on expanding their data collection, it might be beneficial to them to incorporate a shared feature space by location somewhere in their system. If multiple small task networks are drawing from one shared feature space, retraining for all the tasks with a newly added location is likely less training than trying to train an entire system. For a grower just beginning to explore artificial intelligence, these architectures deliver performance with slightly more flexibility than a monolithic architecture.

If retraining a system is daunting to a grower, however, these deep learning experiments demonstrated that training site-specific models outperforms training data-specific models, and results can be beneficial even without the helpful additional data.

The second part of this dissertation focused on applying QARM GA's to prediction tasks using non-sequence and sequence data. For Jornada, the QARM GA's mined for the sequence and non-sequence models did not yield as powerful predictions – however, they were close in performance to the neural network predictors and yielded simple and highly explainable rules. These results of these models were easy to interpret as well as being much more trivial to run as predictors.

The sequence extension did not perform as well as the non-sequence models but did perform more robustly for sub-optimal parameters. The QARM GA's model results are also interesting in their potential for exploring casual relationships between data and sites.

For the VitalDB dataset, the QARM GA models were not useful for finding predictors of patient outcomes. However, they did mine some interesting and explainable feature correlations with patient outcomes that could be worth further investigation and study. While the QARM GA could not approach neural network performance in looking to future patient states, the ability of the algorithm to quantify appearance of data features in time and value ranges with regard to dataset results may make it a helpful tool in determining places to look for causal relationships or factors that influence surgery outcomes.

The QARM GA models particularly brought into focus the importance of using the technique alongside, not instead of, domain expertise. While the models generated explainable input, the actual results are meaningful mainly for a member of the particular field and industry. This expert is needed to interpret the results and catch correlations that are already known, not useful, or meaningless. In these experiments, a cardiologist assisted in categorizing and explaining the results used to evaluate the VitalDB models, as this task was well beyond the purview of a computer scientist. While it is exciting to develop new tools for new applications, domain experts must be kept in the loop for the entire process, from design to interpretation, in order to make and evaluate tools that truly serve the industries well.

In summary, this dissertation suggests further research in modularity for site-specific management and use of QARM GAs to mine rule correlations for explainable output. This dissertation also presented a novel sequence extension to the QARM GA algorithm for use in time-sequenced environments and recommends further research into the use of this sequence extension for diagnostic and investigative contexts. These techniques can be explored further in a variety of domains to help make artificial intelligence more accessible to smaller organizations and aid in the adoption of precision agriculture techniques.

References

[1] J. B. LTR. Data catalog. [Online]. Available: https://lter.jornada.nmsu.edu/data-catalog/

[2] H.-C. Lee, Y. Park, S. B. Yoon, S. M. Yang, D. Park, and C.-W. Jung, "Vitaldb, a high-fidelity multi-parameter vital signs database in surgical patients," *Scientific Data*, vol. 9, no. 1, p. 279, 2022.

[3] J. Graziano Da Silva. (2012, 6) Feeding the world sustainably. [Online]. Available: https://www.un.org/en/chronicle/article/feeding-world-sustainably

[4] M. Wilde. Farmland loss threatens food supply. [Online]. Available: https://www.dtnpf.com/agriculture/web/ag/news/article/2021/02/01/land-use-farmland-loss-threatens

[5] E. Duncan, A. Glaros, D. Z. Ross, and E. Nost, "New but for whom? Discourses of innovation in precision agriculture," *Agriculture and Human Values*, vol. 38, no. 4, pp. 1181–1199, December 2021. [Online]. Available: https://ideas.repec.org/a/spr/agrhuv/v38y2021i4d10.1007_s10460-021-10244-8.html

[6] M. Boehlje, "The value of data/information and the payoff of precision farming."

[7] J. Hatfield, "Precision agriculture and environmental quality: Challenges for research and education."

[8] A. Balafoutis, B. Beck, S. Fountas, J. Vangeyte, T. V. d. Wal, I. Soto, M. Gmez-Barbero, A. Barnes, and V. Eory, "Precision agriculture technologies positively contributing to ghg emissions mitigation, farm productivity and economics," *Sustainability*, vol. 9, no. 8, 2017. [Online]. Available: https://www.mdpi.com/2071-1050/9/8/1339

[9] P. A. C. T. Force, "Task force for reviewing the connectivity and technology needs of precision agriculture in the united states."

[10] N. H. Pablo J. Zarco-Tejada and P. Loudjani, “Precision agriculture: An opportunity for eu-farmers potential support with the cap 2014-2020.”

[11] J. Ammann, C. Umstätter, and N. El Benni, “The adoption of precision agriculture enabling technologies in swiss outdoor vegetable production: a delphi study,” *Precision Agriculture*, vol. 23, no. 4, pp. 1354–1374, Aug 2022. [Online]. Available: https://doi.org/10.1007/s11119-022-09889-0

[12] D. Maloku, P. Balogh, A. Bai, Z. Gabnai, and P. Lengyel, “Trends in scientific research on precision farming in agriculture using science mapping method,” *International Review of Applied Sciences and Engineering*, vol. 11, 09 2020.

[13] G. V. Research. (2021) Precision farming market size, share and trends analysis report by offering (hardware, software, services), by application (yield monitoring, weather tracking, field mapping, crop scouting), by region, and segment forecasts, 2022 - 2023. [Online]. Available: https://www.grandviewresearch.com/industry-analysis/precision-farming-market

[14] technavio. (2022, 5) Precision agriculture market by product and geography - forecast and analysis 2022-2026. [Online]. Available: https://www.technavio.com/report/precision-agriculture-market-industry-analysis

[15] A. M. Research. (2021, 9) Precision agriculture market by component (hardware, software, and service), application (yield monitoring, field mapping, crop scouting, weather tracking and forecasting, irrigation management, inventory management, farm labor management, and others), technology (guidance technology, remote sensing technology, and variable-rate technology): Global opportunity analysis and industry forecast, 2021-2030. [Online]. Available: https://www.alliedmarketresearch.com/precision-agriculture-market

[16] E. Research. (2022, 7) Precision agriculture market, by component (hardware, software, services), by application (yield monitoring, field mapping, crop scouting), by technology (guidance technology, remote sensing, variable-rate technology), and by region forecast to 2030. [Online]. Available: https://www.emergenresearch.com/industry-report/precision-agriculture-market

[17] GlobeNewswire. Precision agriculture market size is projected to reach usd 19.24 billion by 2030, growing at a cagr of 14.95[Online]. Available: https://shorturl.at/yzBC5

[18] F. M. Insights. (2022, 12) Precision agriculture market. [Online]. Available: https://www.futuremarketinsights.com/reports/precision-agriculture-market

[19] bcc Research. (2019, 12) Global markets for precision farming. [Online]. Available: https://www.bccresearch.com/market-research/food-and-beverage/precision-farming-market-report.html

[20] Fact.MR. (2022, 1) Precision farming market. [Online]. Available: https://www.factmr.com/report/4593/precision-farming-market

[21] Markets and Markets. (2022, 3) Precision farming market by technology (guidance, remote sensing and variable rate technology), offering, application, and region (icas, europe, asia pacific, rest of the world) (2022-2030). [Online]. Available: https://www.marketsandmarkets.com/Market-Reports/precision-farming-market-1243.html

[22] V. M. Research. (2021, 12) Global precision farming market size by offering (hardware, software, services), by technology (guidance technology, remote sensing), by application (yield monitoring, crop scouting, field mapping), by geographic scope and forecast. [Online]. Available: https://www.verifiedmarketresearch.com/product/precision-farming-market/

[23] EMR. (2020) Global precision agriculture market outlook. [Online]. Available: https://www.expertmarketresearch.com/reports/precision-agriculture-market

[24] C. Torrez, N. Miller, S. Ramsey, and T. Griffin, “Factors influencing the adoption of precision agricultural technologies by kansas farmers.”

[25] J. Blasch, B. van der Kroon, P. van Beukering, R. Munster, S. Fabiani, P. Nino, and S. Vanino, “Farmer preferences for adopting precision farming technologies: a case study from Italy,” *European Review of Agricultural Economics*, vol. 49, no. 1, pp. 33–81, 12 2020. [Online]. Available: https://doi.org/10.1093/erae/jbaa031

[26] L. S. Antolini and R. F. Scare, “Adoption of precision agriculture technologies by farmers : A systematic literature review and proposition of an integrated conceptual framework,” 2015.

[27] W. Guo, S. Cui, J. Torrion, and N. Rajan, *Data-Driven Precision Agriculture: Opportunities and Challenges*, 08 2015, pp. 353–372.

[28] A. Mcbratney, B. Whelan, T. Ancev, and J. Bouma, “Future directions of precision agriculture,” *Precision Agriculture*, vol. 6, pp. 7–23, 02 2005.

[29] M. Akbari, P. Foroudi, M. Shahmoradi, H. Padash, Z. S. Parizi, A. Khosravani, P. Ataei, and M. T. Cuomo, “The evolution of food security: where are we now, where should we go next?” *Sustainability*, vol. 14, no. 6, p. 3634, 2022.

[30] M. I. of Technology. Explained: Neural networks. [Online]. Available: https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

[31] G. Cloud. Automl beginner’s guide. [Online]. Available: https://cloud.google.com/vertex-ai/docs/beginner/beginners-guide/

[32] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[33] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” 2014. [Online]. Available: https://arxiv.org/abs/1411.4038

[34] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, “Squeeze-and-excitation networks,” 2017. [Online]. Available: https://arxiv.org/abs/1709.01507

[35] R. Pahi, Z. Lonarevi, A. Gams, and A. Ude, "Robot skill learning in latent space of a deep autoencoder neural network," *Robotics and Autonomous Systems*, vol. 135, p. 103690, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889020305303

[36] S. Felton, P. Brault, E. Fromont, and E. Marchand, "Visual servoing in autoencoder latent space," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3234–3241, 2022.

[37] C. Finn, X. Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbeel, "Deep spatial autoencoders for visuomotor learning," 2015. [Online]. Available: https://arxiv.org/abs/1509.06113

[38] D. P. Losey, H. J. Jeon, M. Li, K. Srinivasan, A. Mandlekar, A. Garg, J. Bohg, and D. Sadigh, "Learning latent actions to control assistive robots," *Autonomous Robots*, vol. 46, no. 1, pp. 115–147, Jan 2022. [Online]. Available: https://doi.org/10.1007/s10514-021-10005-w

[39] M. Lippi, P. Poklukar, M. C. Welle, A. Varava, H. Yin, A. Marino, and D. Kragic, "Latent space roadmap for visual action planning of deformable and rigid object manipulation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5619–5626.

[40] L. Sintini and L. Kunze, "Unsupervised and semi-supervised novelty detection using variational autoencoders in opportunistic science missions," in *BMVC*, 2020.

[41] T. Inoue, S. Choudhury, G. De Magistris, and S. Dasgupta, "Transfer learning from synthetic to real images using variational autoencoders for precise position detection," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, 2018, pp. 2725–2729.

[42] A. Spielberg, A. Zhao, T. Du, Y. Hu, D. Rus, and W. Matusik, *Learning-in-the-Loop Optimization: End-to-End Control and Co-Design of Soft Robots through Learned Deep Latent Representations.* Red Hook, NY, USA: Curran Associates Inc., 2019.

[43] A. Hottung, B. Bhandari, and K. Tierney, "Learning a latent search space for routing problems using variational autoencoders," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=90JprVrJBO

[44] S. Zhang, Y. Yao, J. Hu, Y. Zhao, S. Li, and J. Hu, "Deep autoencoder neural networks for short-term traffic congestion prediction of transportation networks," *Sensors*, vol. 19, no. 10, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/10/2229

[45] G. Liu, H. Bao, and B. Han, "A stacked autoencoder-based deep neural network for achieving gearbox fault diagnosis," *Mathematical Problems in Engineering*, 2018.

[46] S. Rongali, A. Rose, D. Mcmanus, A. Bajracharya, A. Kapoor, E. Granillo, and H. Yu, "Learning latent space representations to predict patient outcomes: Model development and validation," 03 2020.

[47] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," 2017. [Online]. Available: https://arxiv.org/abs/1712.00559

[48] J.-M. Perez-Rua, M. Baccouche, and S. Pateux, "Efficient progressive neural architecture search," 2018. [Online]. Available: https://arxiv.org/abs/1808.00391

[49] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," 2018. [Online]. Available: https://arxiv.org/abs/1802.03268

[50] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Smash: One-shot model architecture search through hypernetworks," 2017. [Online]. Available: https://arxiv.org/abs/1708.05344

[51] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," 2017. [Online]. Available: https://arxiv.org/abs/1711.00436

[52] J.-M. Prez-Ra, V. Vielzeuf, S. Pateux, M. Baccouche, and F. Jurie, "Mfas: Multimodal fusion architecture search," 2019. [Online]. Available: https://arxiv.org/abs/1903.06496

[53] I. T. Christou, N. Kefalakis, A. Zalonis, and J. Soldatos, "Predictive and explainable machine learning for industrial internet of things applications," in *2020 16th international conference on distributed computing in sensor systems (DCOSS)*. IEEE, 2020, pp. 213–218.

[54] I. T. Christou, N. Kefalakis, J. K. Soldatos, and A.-M. Despotopoulou, "End-to-end industrial iot platform for quality 4.0 applications," *Computers in Industry*, vol. 137, p. 103591, 2022.

[55] I. T. Christou, E. Amolochitis, and Z.-H. Tan, "A parallel/distributed algorithmic framework for mining all quantitative association rules," *arXiv preprint arXiv:1804.06764*, 2018.

[56] V. P. Alvarez and J. M. Vazquez, "An evolutionary algorithm to discover quantitative association rules from huge databases without the need for an a priori discretization," *Expert Systems with Applications*, vol. 39, no. 1, pp. 585–593, 2012.

[57] X. Yan, C. Zhang, and S. Zhang, "Genetic algorithm-based strategy for identifying association rules without specifying actual minimum support," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3066–3076, 2009.

[58] B. Alataş and E. Akin, "An efficient genetic algorithm for automated mining of both positive and negative quantitative association rules," *Soft Computing*, vol. 10, pp. 230–237, 2006.

[59] A. Salleb-Aouissi, C. Vrain, C. Nortet, X. Kong, V. Rathod, and D. Cassard, "Quantminer for mining quantitative association rules," *Journal of Machine Learning Research*, vol. 14, pp. 3153–3157, 2013.

[60] S. Sinisterra-Sierra, S. Godoy-Calderón, and M. Pescador-Rojas, "Covid-19 data analysis with a multi-objective evolutionary algorithm for causal association rule mining," *Mathematical and Computational Applications*, vol. 28, no. 1, p. 12, 2023.

[61] D. Martín, A. Rosete, J. Alcalá-Fdez, and F. Herrera, "Qar-cip-nsga-ii: A new multi-objective evolutionary algorithm to mine quantitative association rules," *Information Sciences*, vol. 258, pp. 1–28, 2014.

[62] M. Almasi and M. S. Abadeh, "Rare-pears: A new multi objective evolutionary algorithm to mine rare and non-redundant quantitative association rules," *Knowledge-Based Systems*, vol. 89, pp. 366–384, 2015.

[63] L. Li, C. Xue, Y. Xu, C. Wu, and C. Niu, "Posdms: A mining system for oceanic dynamics with time series of raster-formatted datasets," *Remote Sensing*, vol. 14, no. 13, p. 2991, 2022.

[64] A. Troncoso-García, M. Martínez-Ballesteros, F. Martínez-Álvarez, and A. Troncoso, "A new approach based on association rules to add explainability to time series forecasting models," *Information Fusion*, vol. 94, pp. 169–180, 2023.

[65] M. Martínez-Ballesteros, F. Martínez-Álvarez, A. Troncoso, and J. C. Riquelme, "An evolutionary algorithm to discover quantitative association rules in multidimensional time series," *Soft Computing*, vol. 15, pp. 2065–2084, 2011.

[66] W. M. Center. Human brain facts. [Online]. Available: https://www.winstonmedical.org/human-brain-facts/

[67] J. Stiles and T. Jernigan, “The basics of brain development,” *Neuropsychology review*, vol. 20, pp. 327–48, 11 2010.

[68] G. Auda and M. Kamel, “Modular neural networks: a survey,” *International journal of neural systems*, vol. 9, no. 02, pp. 129–151, 1999.

[69] M. Bertolero, B. T. Yeo, and M. D’Esposito, “The modular and integrative functional architecture of the human brain,” *Proceedings of the National Academy of Sciences*, vol. 112, p. 201510619, 11 2015.

[70] Z. Zou, R. Zhao, Y. Wu, Z. Yang, L. Tian, S. Wu, G. Wang, Y. Yu, Q. Zhao, M. Chen, J. Pei, F. Chen, Y. Zhang, S. Song, M. Zhao, and L. Shi, “A hybrid and scalable brain-inspired robotic platform,” *Scientific Reports*, vol. 10, no. 1, p. 18160, Oct 2020. [Online]. Available: https://doi.org/10.1038/s41598-020-73366-9

[71] D. Filan, S. Hod, C. Wild, A. Critch, and S. Russell, “Pruned neural networks are surprisingly modular,” 2020. [Online]. Available: https://arxiv.org/abs/2003.04881

[72] R. Alho, “A systematic literature review on the modularity of modular neural networks and comparison to monolithic solutions.”

[73] M. Amer and T. Maul, “A review of modularization techniques in artificial neural networks,” *Artificial Intelligence Review*, vol. 52, no. 1, pp. 527–561, apr 2019. [Online]. Available: https://doi.org/10.1007%2Fs10462-019-09706-7

[74] A. Schmidt and Z. Bandar, “Modularity - a concept for new neural network architectures,” 11 2001.

[75] G. Auda and M. Kamel, “Modular neural network classifiers: A comparative study,” *Journal of Intelligent and Robotic Systems*, vol. 21, no. 2, pp. 117–129, Feb 1998. [Online]. Available: https://doi.org/10.1023/A:1007925203918

[76] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, “Neural module networks,” 2015. [Online]. Available: https://arxiv.org/abs/1511.02799

[77] V. D'Amario, T. Sasaki, and X. Boix, "How modular should neural module networks be for systematic generalization?" 2021. [Online]. Available: https://arxiv.org/abs/2106.08170

[78] Y. Krishnamurthy and C. Watkins, "Interpretability in gated modular neural networks," in *eXplainable AI approaches for debugging and diagnosis.*, 2021. [Online]. Available: https://openreview.net/forum?id=VMwAfi-c92r

[79] L. Kirsch, J. Kunze, and D. Barber, "Modular networks: Learning to decompose neural computation," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/310ce61c90f3a46e340ee8257bc70e93-Paper.pdf

[80] K. O. Ellefsen, J.-B. Mouret, and J. Clune, "Neural modularity helps organisms evolve to learn new skills without forgetting old skills," *PLoS computational biology*, vol. 11, p. e1004128, 04 2015.

[81] G. Auda, M. Kamel, and H. Raafat, "Modular neural network architectures for classification," in *Proceedings of International Conference on Neural Networks (ICNN'96)*, vol. 2, 1996, pp. 1279–1284 vol.2.

[82] M. Singh and G. Singh, "Two phase learning technique in modular neural network for pattern classification of handwritten hindi alphabets," *Machine Learning with Applications*, vol. 6, p. 100174, 10 2021.

[83] L. Jing, T. Wang, M. Zhao, and P. Wang, "An adaptive multi-sensor data fusion method based on deep convolutional neural networks for fault diagnosis of planetary gearbox," *Sensors*, vol. 17, no. 2, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/2/414

[84] H. R. V. Joze, A. Shaban, M. L. Iuzzolino, and K. Koishida, “MMTM: multimodal transfer module for CNN fusion,” *CoRR*, vol. abs/1911.08670, 2019. [Online]. Available: http://arxiv.org/abs/1911.08670

[85] B. Moslem, M. Khalil, M. O. Diab, A. Chkeir, and C. Marque, “A multisensor data fusion approach for improving the classification accuracy of uterine emg signals,” in *2011 18th IEEE International Conference on Electronics, Circuits, and Systems*, 2011, pp. 93–96.

[86] M. Abavisani, H. R. V. Joze, and V. M. Patel, “Improving the performance of unimodal dynamic hand-gesture recognition with multimodal training,” 2018. [Online]. Available: https://arxiv.org/abs/1812.06145

[87] C. G. M. Snoek, M. Worring, and A. W. M. Smeulders, “Early versus late fusion in semantic video analysis,” in *Proceedings of the 13th Annual ACM International Conference on Multimedia*, ser. MULTIMEDIA ’05. New York, NY, USA: Association for Computing Machinery, 2005, p. 399402. [Online]. Available: https://doi.org/10.1145/1101149.1101236

[88] A. Shahroudy, T.-T. Ng, Y. Gong, and G. Wang, “Deep multimodal feature analysis for action recognition in rgb+d videos,” 2016. [Online]. Available: https://arxiv.org/abs/1603.07120

[89] X. Zhou and B. Bhanu, “Feature fusion of side face and gait for video-based human identification,” *Pattern Recognition*, vol. 41, no. 3, pp. 778–795, 2008, part Special issue: Feature Generation and Machine Learning for Robust Multimodal Biometrics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132030700310X

[90] X. Yang, P. Molchanov, and J. Kautz, “Multilayer and multimodal fusion of deep neural networks for video classification,” in *Proceedings of the 24th ACM International Conference on Multimedia*, ser. MM ’16. New York, NY, USA: Association for Computing Machinery, 2016, p. 978987. [Online]. Available: https://doi.org/10.1145/2964284.2964297

[91] J. Arevalo, T. Solorio, M. Montes-y Gmez, and F. A. Gonzlez, “Gated multimodal units for information fusion,” 2017. [Online]. Available: https://arxiv.org/abs/1702.01992

[92] A. Zadeh, M. Chen, S. Poria, E. Cambria, and L. Morency, “Tensor fusion network for multimodal sentiment analysis,” *CoRR*, vol. abs/1707.07250, 2017. [Online]. Available: http://arxiv.org/abs/1707.07250

[93] S. Li, H. Wang, L. Song, P. Wang, L. Cui, and T. Lin, “An adaptive data fusion strategy for fault diagnosis based on the convolutional neural network,” *Measurement*, vol. 165, p. 108122, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224120306606

[94] D. Hu, F. Nie, and X. Li, “Dense multimodal fusion for hierarchically joint representation,” 2018. [Online]. Available: https://arxiv.org/abs/1810.03414

[95] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Ng, “Multimodal deep learning,” in *International Conference on Machine Learning*, 2011.

[96] V. Vielzeuf, A. Lechervy, S. Pateux, and F. Jurie, “Centralnet: a multilayer approach for multimodal fusion,” 2018. [Online]. Available: https://arxiv.org/abs/1808.07275

[97] C. Hori, T. Hori, T.-Y. Lee, K. Sumi, J. R. Hershey, and T. K. Marks, “Attention-based multimodal fusion for video description,” 2017. [Online]. Available: https://arxiv.org/abs/1701.03126

[98] X. Long, C. Gan, G. de Melo, X. Liu, Y. Li, F. Li, and S. Wen, "Multimodal keyless attention fusion for video classification," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[99] W. Yu, I. Y. Kim, and C. Mechefske, "Analysis of different rnn autoencoder variants for time series classification and machine prognostics," *Mechanical Systems and Signal Processing*, vol. 149, p. 107322, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0888327020307081

[100] Y. Wei, J. Jang-Jaccard, W. Xu, F. Sabrina, S. Camtepe, and M. Boulic, "Lstm-autoencoder based anomaly detection for indoor air quality time series data," 2022. [Online]. Available: https://arxiv.org/abs/2204.06701

[101] Z. Shi, A. A. Mamun, C. Kan, W. Tian, and C. Liu, "An lstm-autoencoder based online side channel monitoring approach for cyber-physical attack detection in additive manufacturing," *Journal of Intelligent Manufacturing*, 01 2022.

[102] P. Mobtahej, X. Zhang, M. Hamidi, and J. Zhang, "An lstm-autoencoder architecture for anomaly detection applied on compressors audio data," *Computational and Mathematical Methods*, vol. 2022, pp. 1–22, 09 2022.

[103] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network anomaly detection using lstm based autoencoder," in *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, ser. Q2SWinet '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 3745. [Online]. Available: https://doi.org/10.1145/3416013.3426457

[104] S. Ghimire, R. Deo, H. Wang, M. AL-Musaylh, D. Casillas-Perez, and S. Salcedo-Sanz, “Stacked lstm sequence-to-sequence autoencoder with feature selection for daily solar radiation prediction: A review and new modeling results,” *Energies*, vol. 15, p. 1061, 01 2022.

[105] T. Rama and . ltekin, “Lstm autoencoders for dialect analysis,” *Third Workshop on NLP for Similar Languages, Varieties and Dialects*, 12 2016.

[106] S. Dana, “Comparison of lstm autoencoder based deep learning enabled bayesian inference using two time series reconstruction approaches,” 2022. [Online]. Available: https://arxiv.org/abs/2203.01936

[107] H. Nguyen, K. Tran, S. Thomassey, and M. Hamad, “Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management,” *International Journal of Information Management*, vol. 57, p. 102282, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S026840122031481X

[108] Y. Cheng, K. Hu, J. Wu, H. Zhu, and X. Shao, “Autoencoder quasi-recurrent neural networks for remaining useful life prediction of engineering systems,” *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 2, pp. 1081–1092, 2022.

[109] A. Sagheer and M. Kotb, “Unsupervised pre-training of a deep lstm-based stacked autoencoder for multivariate time series forecasting problems,” *Scientific Reports*, vol. 9, no. 1, p. 19038, Dec 2019. [Online]. Available: https://doi.org/10.1038/s41598-019-55320-6

[110] D. Thakur, S. Biswas, E. S. L. Ho, and S. Chattopadhyay, “Convae-lstm: Convolutional autoencoder long short-term memory network for smartphone-based human activity recognition,” *IEEE Access*, vol. 10, pp. 4137–4156, 2022.

[111] R. A. Shaikh and S. V. Shashikala, “An autoencoder and lstm based intrusion detection approach against denial of service attacks,” in *2019 1st International Conference on Advances in Information Technology (ICAIT)*, 2019, pp. 406–410.

[112] W. Wei, H. Wu, and H. Ma, “An autoencoder and lstm-based traffic flow prediction method,” *Sensors*, vol. 19, no. 13, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/13/2946

[113] S. Ding, “A detachable lstm with residual-autoencoder features method for motion recognition in video sequences,” Ph.D. dissertation, 2020. [Online]. Available: http://rave.ohiolink.edu/etdc/view?acc_num=osu160673417735023

[114] Z. Que, Y. Liu, C. Guo, X. Niu, Y. Zhu, and W. Luk, “Real-time anomaly detection for flight testing using autoencoder and lstm,” in *2019 International Conference on Field-Programmable Technology (ICFPT)*, 2019, pp. 379–382.

[115] F. Karim, S. Majumdar, H. Darabi, and S. Harford, “Multivariate LSTM-FCNs for time series classification,” *Neural Networks*, vol. 116, pp. 237–245, aug 2019. [Online]. Available: https://doi.org/10.1016%2Fj.neunet.2019.04.014

[116] A. G. Dastider, F. Sadik, and S. A. Fattah, “An integrated autoencoder-based hybrid cnn-lstm model for covid-19 severity prediction from lung ultrasound,” *Computers in Biology and Medicine*, vol. 132, p. 104296, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482521000901

[117] H. Zou, Y. Zhou, J. Yang, H. Jiang, L. Xie, and C. J. Spanos, “Deepsense: Device-free human activity recognition via autoencoder long-term recurrent convolutional network,” in *2018 IEEE International Conference on Communications (ICC)*, 2018, pp. 1–6.

[118] M. Ibrahim, J. Haworth, A. Lipani, N. Sari Aslam, T. Cheng, and N. Christie, “Variational-lstm autoencoder to forecast the spread of coronavirus across the globe,” *PLOS ONE*, vol. 16, p. e0246120, 01 2021.

[119] C. Collins, D. Dennehy, K. Conboy, and P. Mikalef, "Artificial intelligence in information systems research: A systematic literature review and research agenda," *International Journal of Information Management*, vol. 60, p. 102383, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0268401221000761

[120] A. Kamilaris and F. X. Prenafeta-Bold, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169917308803

[121] M. Gomez Selvaraj, A. Vergara, F. Montenegro, H. Alonso Ruiz, N. Safari, D. Raymaekers, W. Ocimati, J. Ntamwira, L. Tits, A. B. Omondi, and G. Blomme, "Detection of banana plants and their major diseases through aerial images and machine learning methods: A case study in dr congo and republic of benin," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 169, pp. 110–124, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271620302410

[122] P. Chen, Q. Xiao, J. Zhang, C. Xie, and B. Wang, "Occurrence prediction of cotton pests and diseases by bidirectional long short-term memory networks with climate and atmosphere circulation," *Computers and Electronics in Agriculture*, vol. 176, p. 105612, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169918317794

[123] J. P. Schwarz Schuler, S. Roman, M. Abdel-nasser, H. Rashwan, and D. Puig, *Reliable Deep Learning Plant Leaf Disease Classification Based on Light-Chroma Separated Branches*, 10 2021.

[124] M. Kerkech, A. Hafiane, and R. Canals, "Deep leaning approach with colorimetric spaces and vegetation indices for vine diseases detection in uav images," *Computers and Electronics in Agriculture*, vol. 155, pp. 237–243, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169918310044

[125] ——, “Vine disease detection in uav multispectral images using optimized image registration and deep learning segmentation approach,” *Computers and Electronics in Agriculture*, vol. 174, p. 105446, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016816991932558X

[126] J. Albetis, S. Duthoit, F. Guttler, A. Jacquin, M. Goulard, H. Poilv, J.-B. Fret, and G. Dedieu, “Detection of flavescence dore grapevine disease using unmanned aerial vehicle (uav) multispectral imagery,” *Remote Sensing*, vol. 9, no. 4, 2017. [Online]. Available: https://www.mdpi.com/2072-4292/9/4/308

[127] M. Kerkech, A. Hafiane, and R. Canals, “Vddnet: Vine disease detection network based on multispectral images and depth map,” *Remote Sensing*, vol. 12, no. 20, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/20/3305

[128] A. Meunkaewjinda, P. Kumsawat, K. Attakitmongcol, and A. Srikaew, “Grape leaf disease detection from color imagery using hybrid intelligent system,” in *2008 5th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*, vol. 1, 2008, pp. 513–516.

[129] M. E. Cintra, C. A. A. Meira, M. C. Monard, H. A. Camargo, and L. H. A. Rodrigues, “The use of fuzzy decision trees for coffee rust warning in brazilian crops,” in *2011 11th International Conference on Intelligent Systems Design and Applications*, 2011, pp. 1347–1352.

[130] S. G. Wu, F. S. Bao, E. Y. Xu, Y.-X. Wang, Y.-F. Chang, and Q.-L. Xiang, “A leaf recognition algorithm for plant classification using probabilistic neural network,” in *2007 IEEE International Symposium on Signal Processing and Information Technology*, 2007, pp. 11–16.

[131] A. Fuentes, S. Yoon, S. C. Kim, and D. S. Park, “A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition,” *Sensors*, vol. 17, no. 9, 2017. [Online]. Available: https://www.mdpi.com/1424-8220/17/9/2022

[132] S. Rodrguez, T. Gualotua, and C. Grilo, “A system for the monitoring and predicting of data in precision agriculture in a rose greenhouse based on wireless sensor networks,” *Procedia Computer Science*, vol. 121, pp. 306–313, 2017, cENTERIS 2017 - International Conference on ENTERprise Information Systems / ProjMAN 2017 - International Conference on Project MANagement / HCist 2017 - International Conference on Health and Social Care Information Systems and Technologies, CENTERIS/ProjMAN/HCist 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050917322330

[133] A. Jagarlapudi, S. Dhanachandran, K. Vijayalakshmi, S. Merchant, and U. Desai, “Data mining and wireless sensor network for groundnut pest/disease interaction and predictions - a preliminary study,” *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 5, pp. 427–436, 12 2013.

[134] A. Khattab, S. E. Habib, H. Ismail, S. Zayan, Y. Fahmy, and M. M. Khairy, “An iot-based cognitive monitoring system for early plant disease forecast,” *Computers and Electronics in Agriculture*, vol. 166, p. 105028, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919311949

[135] S. Trilles, J. Torres-Sospedra, scar Belmonte, F. J. Zarazaga-Soria, A. Gonzlez-Prez, and J. Huerta, “Development of an open sensorized platform in a smart agriculture context: A vineyard support system for monitoring mildew disease,” *Sustainable Computing: Informatics and Systems*, vol. 28, p. 100309, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210537918302270

[136] S. S. Patil and S. A. Thorat, “Early detection of grapes diseases using machine learning and iot,” in *2016 Second International Conference on Cognitive Computing and Information Processing (CCIP)*, 2016, pp. 1–5.

[137] H. Wani and N. Ashtankar, “An appropriate model predicting pest/diseases of crops using machine learning algorithms,” in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2017, pp. 1–4.

[138] N. Materne and M. Inoue, "Iot monitoring system for early detection of agricultural pests and diseases," in *2018 12th South East Asian Technical University Consortium (SEATUC)*, vol. 1, 2018, pp. 1–5.

[139] S. Khan and M. Narvekar, "Disorder detection of tomato plant(solanum lycopersicum) using iot and machine learning," *Journal of Physics: Conference Series*, vol. 1432, no. 1, p. 012086, jan 2020. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1432/1/012086

[140] K. Bhargava, A. Kashyap, and T. A. Gonsalves, "Wireless sensor network based advisory system for apple scab prevention," in *2014 Twentieth National Conference on Communications (NCC)*, 2014, pp. 1–6.

[141] J. Burrell, T. Brooke, and R. Beckwith, "Vineyard computing: sensor networks in agricultural production," *IEEE Pervasive Computing*, vol. 3, no. 1, pp. 38–45, 2004.

[142] G. Anastasi, O. Farruggia, G. Lo Re, and M. Ortolani, "Monitoring high-quality wine production using wireless sensor networks," in *2009 42nd Hawaii International Conference on System Sciences*, 2009, pp. 1–7.

[143] T. Togami, K. Yamamoto, A. Hashimoto, N. Watanabe, K. Takata, H. Nagai, and T. Kameoka, "A wireless sensor network in a vineyard for smart viticultural management," in *SICE Annual Conference 2011*, 2011, pp. 2450–2454.

[144] X. Ding, G. Xiong, B. Hu, L. Xie, and S. Zhou, "Environment monitoring and early warning system of facility agriculture based on heterogeneous wireless networks," in *Proceedings of 2013 IEEE International Conference on Service Operations and Logistics, and Informatics*, 2013, pp. 307–310.

[145] M. Ouhami, A. Hafiane, Y. Es-Saady, M. El Hajji, and R. Canals, "Computer vision, iot and data fusion for crop disease detection using machine learning: A survey and ongoing research," *Remote Sensing*, vol. 13, no. 13, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/13/2486

[146] I. E. Livieris, S. D. Dafnis, G. K. Papadopoulos, and D. P. Kalivas, "A multiple-input neural network model for predicting cotton production quantity: A case study," *Algorithms*, vol. 13, no. 11, 2020. [Online]. Available: https://www.mdpi.com/1999-4893/13/11/273

[147] Z. Chu and J. Yu, "An end-to-end model for rice yield prediction using deep learning fusion," *Computers and Electronics in Agriculture*, vol. 174, p. 105471, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169920300314

[148] M. Maimaitijiang, V. Sagan, P. Sidike, S. Hartling, F. Esposito, and F. B. Fritschi, "Soybean yield prediction from uav using multimodal data fusion and deep learning," *Remote Sensing of Environment*, vol. 237, p. 111599, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425719306194

[149] M. Maimaitijiang, A. Ghulam, P. Sidike, S. Hartling, M. Maimaitiyiming, K. Peterson, E. Shavers, J. Fishman, J. Peterson, S. Kadam, J. Burken, and F. Fritschi, "Unmanned aerial system (uas)-based phenotyping of soybean using multi-sensor data fusion and extreme learning machine," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 134, pp. 43–58, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271617303246

[150] J. Zhang, R. Pu, L. Yuan, W. Huang, C. Nie, and G. Yang, "Integrating remotely sensed and meteorological observations to forecast wheat powdery mildew at a regional scale," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 11, pp. 4328–4339, 2014.

[151] Y. Zhao, L. Liu, C. Xie, R. Wang, F. Wang, Y. Bu, and S. Zhang, "An effective automatic system deployed in agricultural internet of things using multi-context fusion network towards crop disease recognition in the wild," *Appl. Soft Comput.*, vol. 89, no. C, apr 2020. [Online]. Available: https://doi.org/10.1016/j.asoc.2020.106128

[152] S. H. Lee, C. S. Chan, S. J. Mayo, and P. Remagnino, "How deep learning extracts and learns leaf features for plant classification," *Pattern Recognition*, vol. 71, pp. 1–13, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S003132031730198X

[153] R. Chandra, M. Swaminathan, T. Chakraborty, J. Ding, Z. Kapetanovic, P. Kumar, and D. Vasisht, "Democratizing data-driven agriculture using affordable hardware," *IEEE Micro*, vol. 42, no. 1, pp. 69–77, 2022.

[154] T. Chakraborty, H. Shi, Z. Kapetanovic, B. Priyantha, D. Vasisht, B. Vu, P. Pandit, P. Pillai, Y. Chabria, A. Nelson, M. Daum, and R. Chandra, "Whisper: IoT in the TV white space spectrum," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. Renton, WA: USENIX Association, Apr. 2022, pp. 401–418. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/chakraborty

[155] P. Kumar, R. Chandra, C. Bansal, S. Kalyanaraman, T. Ganu, and M. Grant, "Micro-climate prediction - multi scale encoder-decoder based deep learning framework," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 31283138. [Online]. Available: https://doi.org/10.1145/3447548.3467173

[156] M. Zhao, P. A. Olsen, and R. Chandra, "Seeing through clouds in satellite images," 2021. [Online]. Available: https://arxiv.org/abs/2106.08408

[157] S. Jha, Y. Li, S. Noghabi, V. Ranganathan, P. Kumar, A. Nelson, M. Toelle, S. Sinha, R. Chandra, and A. Badam, "Visage: Enabling timely analytics for drone imagery," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 789803. [Online]. Available: https://doi.org/10.1145/3447993.3483273

[158] D. Vasisht, J. Shenoy, and R. Chandra, "L2d2: Low latency distributed downlink for leo satellites," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, ser. SIGCOMM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 151164. [Online]. Available: https://doi.org/10.1145/3452296.3472932

[159] agStack. Open-source digital infrastructure for the agriculture ecosystem. [Online]. Available: https://agstack.org/

[160] OpenTEAM. Welcome to openteam. [Online]. Available: https://openteam.community/

[161] OurSci. Oursci. [Online]. Available: https://www.our-sci.net/

[162] V. Rossi, F. Salinari, S. Poni, T. Caffi, and T. Bettati, "Addressing the implementation problem in agricultural decision support systems: the example of vite.net," *Computers and Electronics in Agriculture*, vol. 100, pp. 88–99, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169913002536

[163] Keras. Classification metrics based on true/false positives and negatives. [Online]. Available: https://keras.io/api/metrics/classification_metrics/

[164] E. Bennett-Guerrero, D. E. Feierman, G. R. Barclay, M. K. Parides, P. A. Sheiner, M. G. Mythen, D. M. Levine, T. S. Parker, S. F. Carroll, M. L. White *et al.*, "Preoperative and intraoperative predictors of postoperative morbidity, poor graft function, and early rejection in 190 patients undergoing liver transplantation," *Archives of surgery*, vol. 136, no. 10, pp. 1177–1183, 2001.

[165] V. Sharma, C. Kleb, C. Sheth, B. R. Verma, V. Jain, R. Sharma, P. Parikh, J. Cywinski, K. N. Menon, J. M. Esfeh *et al.*, "Cardiac considerations in liver transplantation," *Cleveland Clinic Journal of Medicine*, vol. 89, no. 1, pp. 46–55, 2022.

[166] J. H. McDonald, *Handbook of biolological statistics.* New York, 2014.

[167] J. Abate, “Personal conversation,” 2023.

[168] L. Su, P. Pan, D. Liu, and Y. Long, “Mean airway pressure has the potential to become the core pressure indicator of mechanical ventilation: raising to the front from behind the clinical scenes,” *Journal of Intensive Medicine*, vol. 1, no. 02, pp. 96–98, 2021.