Department of Computer Science and Engineering, University of Nevada – Reno

Gigafactory Systems Machine Learning Project

Team 15: Adam Cassell, Braeden Richards, Ashlee Ladouceur

Project Part 2: Revised Specification and Design

Instructors: Sergiu Dascalu, Devrin Lee

External advisor(s): Dr. Emily Hand (UNR), Gavin Hall (Tesla)

22 February 2019

Table of Contents

Abstract

Team 15 is creating a machine learning system that provides real-time and offline analysis of Tesla's Gigafactory systems. The pipeline of the system design includes building an interface for users to connect to Tesla's mySQL database containing both historical and current data of sensors in the factory, building models, training and testing those models, and building a visual analysis of the data received from the training and testing. The team plans to implement a Python backend, a SQL server to hold both Gigafactory and user data, and a simple Interface to interact with the system.

Recent Project Changes

There has been only one recent project change since the Revised Concept and Project Management document. Instead of using .NET to host the frontend, the frontend will now be hosted through using the Python library Flask, which will be integrated into our primary python script. This is to reduce dependencies in the system, reduce time of design, and make communication between the frontend and backend quicker and more reliable.

Updated Specification

Summary of Changes in Project Specification

The most important modification in for the project's specification was the addition of differentiating between admin and user functionality and experience. The functional requirements added FR01 and FR02, shown in Table 1, that describe user and admin interface experience. In addition, many functional requirements and use cases were modified to specify the permissions that differentiate admins and users. This difference in permissions is displayed in the addition of the "Admin" character in the use case diagram shown in Figure 1, whereas the original specification document only contained "User". This change was necessary to ensure only those trained to train models could access the training and testing data.

Any requirement or use case that described modifying outputs of models was either removed or edited to remove that content. This change was necessary as the models will not allow output modification, as that is not how TensorFlow will operate within this system. The use of "Ignition" was removed and changed to "MySQL" in all requirements and use cases. Ignition is a database that will be providing data to a MySQL server that will have designated databases for the system to use. This change was necessary to be more accurate in specifications. Finally, after the changes to the functional requirements and the use cases were modified, the traceability matrix shown in Figure 2 was updated and improved for accuracy from the original version, using input from the grading team of CS425.

Updated Technical Requirements Specification

The following functional and non-functional requirements are grouped based on priority. Those are listed as: type [1] priority to be implemented by the end of the Spring semester, type [2] priority to be implemented by the end of the Spring semester depending on time restraints, and type [3] priority will not be implemented by the end of the Spring Semester. Type [3] requirements are those that would be useful to implement later in the project's life.

Functional Requirements

ID	Priority	Requirement Description
FR01	[1]	System will allow admins to have their own interface and permissions after sign-in
FR02	[1]	System will allow users to have their own interface and permissions after sign-in
FR03	[1]	Allow admins to modify the inputs for model training
FR04	[1]	Allow admins to update a model after training with new or old data
FR05	[1]	Allow admins to check in a model that is checked out
FR06	[1]	Allow admins to run the neural network multiple times using different inputs
FR07	[1]	Show admins progress for training as a progress bar
FR08	[1]	Allow admins and users to visualize each variable as a graph
FR09	[1]	Allow admins to choose the activation function for training
FR10	[1]	Allow admins and users to view past data of model(s)
FR11	[1]	Allow the admins and users to see the accuracy of the test data
FR12	[1]	System will alert admins and users of maintenance needs
FR13	[1]	System will show admins and users the top-level details of each model
FR14	[1]	System will show admins and users what time windows the data in the model(s) correspond to
FR15	[1]	System will show admins and users how many total timesteps were used to approximate the model

FR16	[1]	System will show admins and users how deep the model is (hidden nodes and hidden layers)
FR17	[1]	System will show admins and users the number of bias nodes in the model
FR18	[1]	System will show admins and users the training parameters used in each model
FR19	[1]	Allow admins to select the number of names and inputs
FR20	[1]	Allow admins to set recorded data bounds of each variable
FR21	[1]	Allow admins to select the input/output type (bool, int)
FR22	[2]	System can run multiple models training at the same time
FR23	[3]	System will be running in real-time, so the user does not have to change inputs
FR24	[3]	System will use a predictive model based on past data

Table 1: Functional requirements detailed and prioritized.

Non-functional Requirements

ID	Priority	Requirement Description
NF01	[1]	System shall be implemented using Python3
NF02	[1]	System shall be Debian compatible
NF03	[1]	System will use the TensorFlow library
NF04	[1]	Machine learning models will have minimal bias and variance
NF05	[1]	System interface will have a short user learning curve
NF06	[1]	System interface will have intuitive design
NF07	[1]	System shall maintain a simple user interface
NF08	[1]	System shall accept user input via keyboard
NF09	[1]	System shall accept input via mouse
NF10	[1]	System shall accept input via txt file
NF11	[1]	System shall accept input via JSON file
NF12	[1]	System will be able to output to txt file
NF13	[1]	System will be able to output to JSON file
NF14	[1]	System will be able to read data from the MySQL server
NF15	[1]	System will be able to write data onto the MySQL server
NF16	[2]	System shall be macOS compatible
NF17	[2]	System will minimize system resource usage
NF18	[2]	System will calculate output within 4 seconds
NF19	[2]	System will calculate input within 6 seconds
NF20	[3]	System shall be Windows 10 compatible

Table 2: Non-functional requirements detailed and prioritized.

Updated Case Modeling

Use Case Diagram

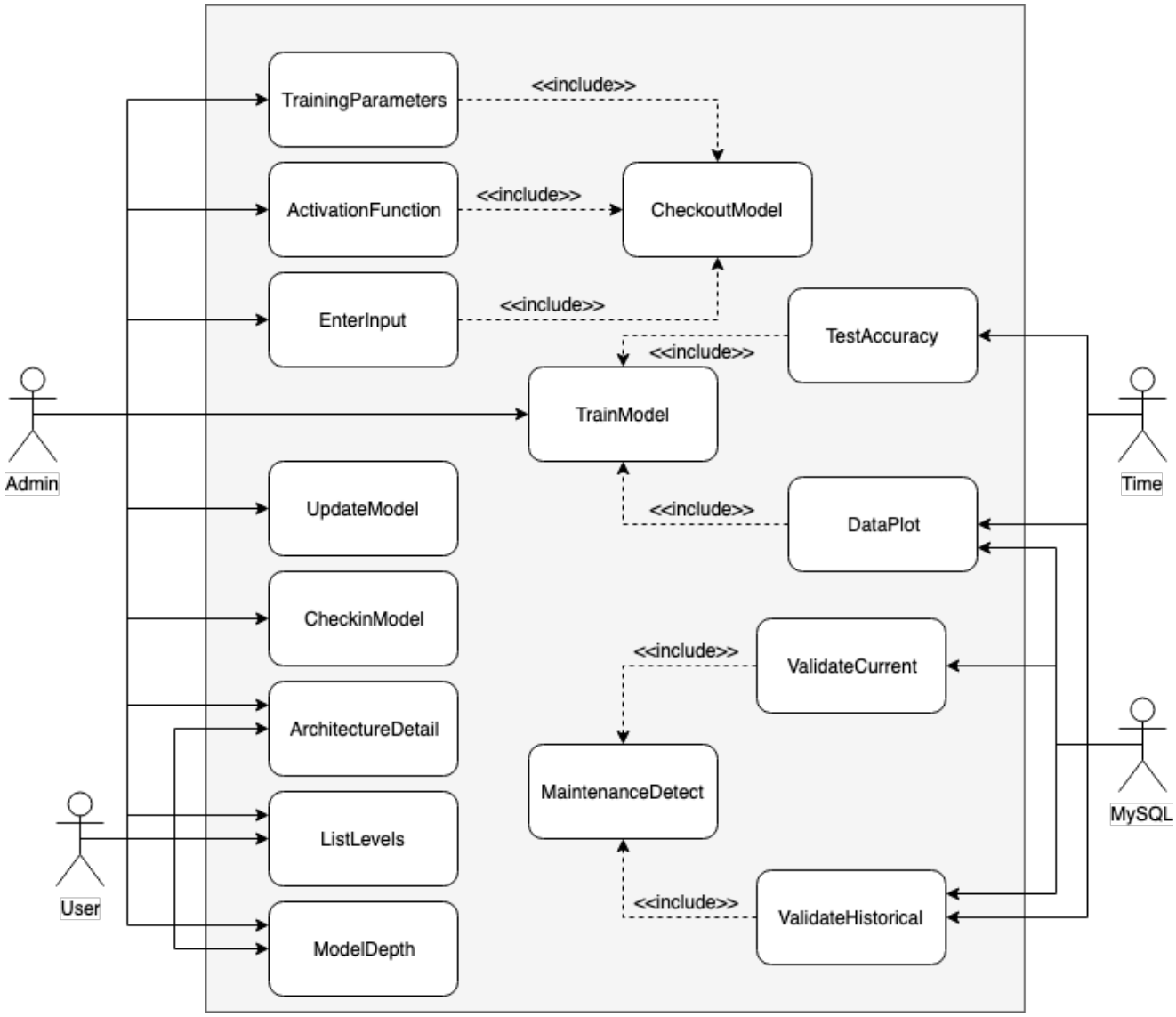


Figure 1: Use Case diagram.

Detailed Use Cases

ID	Use Case	Description
UC01	ValidateCurrent	Admins and users will be able to validate the current state and quality of the data coming from MySQL in order to verify where more sensors are needed or where they are not functioning properly.
UC02	ValidateHistorical	Admins and users will be able to validate the historical state and quality of the data from MySQL in order to verify where more sensors are needed or where they are not functioning properly.

UC03	CheckoutModel	Admins can check out multiple models from the system to be able to train, test, and update.
UC04	TrainingParameters	Admins can manually enter the training parameters before training a checked-out model. This includes the back-prop equality, the number of epochs, and the batch size.
UC05	ActivationFunction	Admins will be able to choose the activation function between layers of a model before training, such as ReLu, Linear, Gigmoid, or Tanh.
UC06	EnterInputs	Admins can manually enter the details for inputs in each model that is checked out. This includes the number and names of inputs, the variable type (bool or int), and the data bounds or each variable.
UC07	TrainModel	Admins can enter a model checked out into training after it is checked out with the data provided from MySQL following the systems settings.
UC08	TestAccuracy	The system will provide admins and users with the testing accuracy of the data after training the model.
UC09	UpdateModel	Before checking in a model that was trained, admins can update the model to fill in the new acquired data into the system.
UC10	CheckinModel	After checking out and optionally updating a model, admins can check-in the model into the system.
UC11	MaintenanceDetect	The system will provide admins and users a preventative maintenance detection. This will detect anomalies and predict remaining useful life or failure within each model.
UC12	ArchitectureDetail	Admins and users can request to see an architecture breakdown of each model. This includes detailing the type of model, time series dependence, accuracy, last update, time windows, depth of nodes and layers, activation function, number of bias nodes, training parameters, inputs/output details, and data bounds.
UC13	DataPlot	The system will display a real-time plotting and report of machine learning prediction versus actual results during training.
UC14	ListLevels	Admins and users can request to see the top-level system of each model in the system, as each system will have multiple models due to the need to observe different input/output combinations.
UC15	ModelDepth	Admins and users can request to see the depth of the model. This includes showing hidden nodes and hidden layers.

Table 3: Details for each use case in the use case diagram.

Updated Requirement Traceability Matrix

	UC01	UC02	UC03	UC04	UC05	UC06	UC07	UC08	UC09	UC10	UC11	UC12	UC13	UC14	UC15
FR01	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
FR02	X	X						X			X	X	X	X	X
FR03						X									
FR04									X						
FR05										X					
FR06					X		X								
FR07							X				X				
FR08											X		X		
FR09					X										
FR10		X											X		
FR11								X							
FR12											X				
FR13														X	
FR14												X			
FR15												X			
FR16															X
FR17												X			
FR18												X			
FR19						X									
FR20				X											
FR21				X		X									
FR22							X								
FR23	X										X		X		
FR24	X	X							X						

Figure 2: Requirement traceability matrix between use cases and functional requirements.

Updated Design

Summary of Changes in Project Design

The design of GigaML has undergone few changes since the original 2018 design document. The primary changes have been in how the web frontend is hosted, and how the user interface is organized. For the frontend hosting, we have switched from a .NET backend to using the Flask python hosting library. This allows us to keep our web app hosting in the same python environment that our data processing and machine learning scripts will live, greatly simplifying the code organization and communication between the frontend and backend components. On the UI side, all features are now organized into three main categories for each model that a user can look at: Details, Predict, and Train. These are the three high-level useful features a user can use in relation to each model. Within Details, there are a further three options: Performance, Summary, and Inspect Data. This organization is more coherent and in line with the workflow requirements of Gigafactory engineers.

Updated High-Level and Medium-Level Design

System-Level Diagram

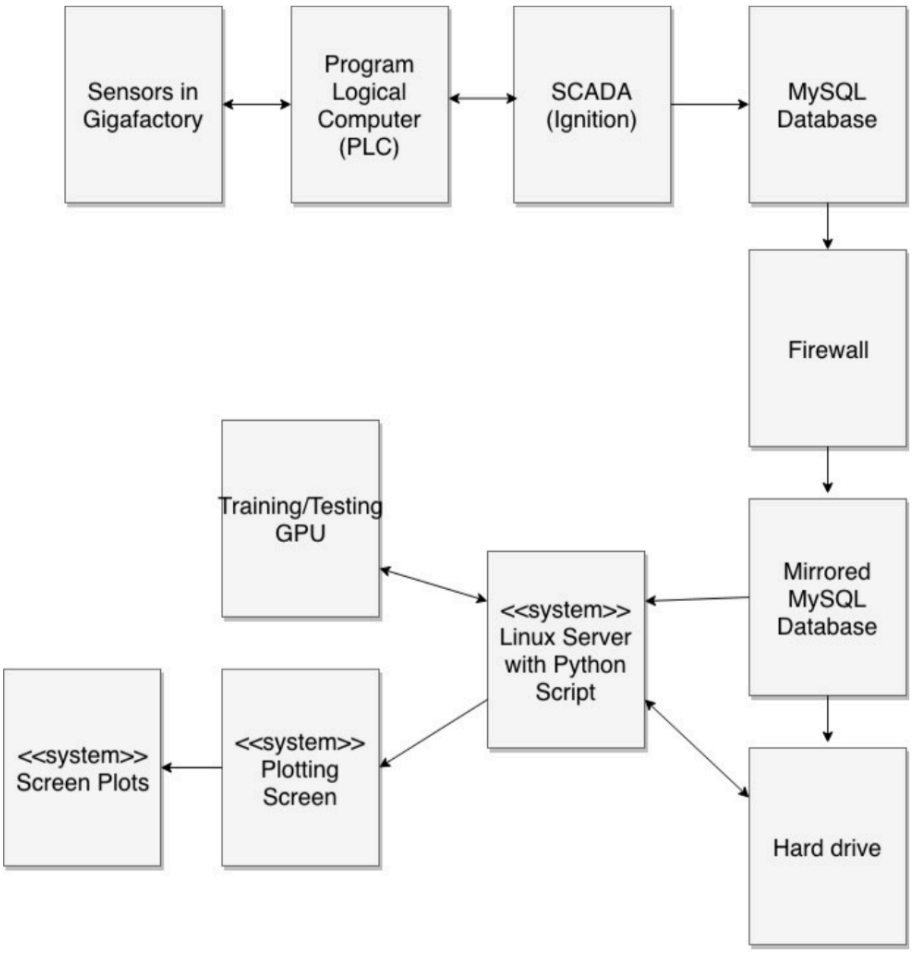


Figure 3: System-level Diagram

Figure 3 shows the systems context model to define what Team 15 will be creating for the system and how the system interacts with the Gigafactory environment. The system interacts with databases already in existence within Tesla's factory, and thus it is important to understand the environment to plan design before implementation. Sensors are placed throughout the factory in the form of actuators or valves. These sensors send data to the Program Logical Computer, or PLC. The PLC tells field sensors how to operate, thus there is a bidirectional relationship between these two systems. For example, if a temperature read from a sensor reaches a certain threshold, the PLC will tell the sensors to turn on a fan. Data from the PLC is then stored in a supervisory control and data acquisition (SCADA) called Ignition. There are read and write capability going either way between the PLC and Ignition. All data from Ignition is stored in a MySQL Database that is protected by a firewall. Team 15 will be developing the system using data from a mirrored MySQL Database across the firewall to avoid privilege issues. A hard drive will be used to store and preload data onto a Linux server with the team's python script for the system. This hard drive can store weights and other data from the mirrored MySQL Database. The Linux server will be where the system developed lives. The Linux server will send data to Tesla's GPU located in Santa Clara, California for training and testing. All data from the training and testing will come back to the Linux server, which can be referenced by the frontend web application for plotting and analysis.

Structure of Software

Team 15's focus in this project is on the design and implementation of a model creation and testing tool. To aid Team 15, the use of machine learning libraries for increased efficiency and speed will be used. Due to the use of libraries, what would otherwise be an object-oriented project will be designed and created as a non-object-oriented project. Therefore, modules based on a linear workflow pipeline will be the basis of the software structure. Note: Model Data refers to model parameters like weight values, activation functions, etc. throughout the module descriptions, whereas Table Data refers to actual factory sensor data.

Data Transferring Module

_Name	**ServerDataToScript**
_Description	Gathers the desired data from the MySQL database and places it into a table (and/or) gathers model data
_Higher Level Unit	Data Transferring Module
_Input(s)	Data timestamp start; Data timestamp end; Table/Column(s) of Input(s); (and/or) Model choice
_Output(s)	Table Data (and/or) Model Data
_Program(s)/_Module(s) Called	Void

_Name	**ScriptDataToServer**
_Description	Sends data to be stored in the MySQL database
_Higher Level Unit	Data Transferring Module
_Input(s)	Table Data to send (and/or) model data;
_Output(s)	Table Data
_Program(s)/_Module(s) Called	Void

_Name	**HardDriveDataToScript**
_Description	Gathers the desired data from the hard drive and places it into a table (and/or) gathers model data
_Higher Level Unit	Data Transferring Module
_Input(s)	Data timestamp start; Data timestamp end; Table/Column(s) of Input(s); (and/or) Model choice
_Output(s)	Table Data (and/or) Model Data
_Program(s)/_Module(s) Called	Void

_Name	**ScriptDataToHardDrive**
_Description	Sends data to the hard drive to be stored
_Higher Level Unit	Data Transferring Module
_Input(s)	Table Data to send (and/or) model data
_Output(s)	Table Data (and/or) Model Data
_Program(s)/_Module(s) Called	Void

Model Module

_Name	**TrainModel**
_Description	Uses the loaded data to train the model
_Higher Level Unit	Model Module
_Input(s)	Table Data with known inputs and outputs, Model Data
_Output(s)	Model Data; Table Data with Predicted Outputs; Training Accuracy
_Program(s)/_Module(s) Called	Void

_Name	**Test Model**
_Description	Uses the loaded data to test the model
_Higher Level Unit	Model Module
_Input(s)	Table Data with known inputs and outputs
_Output(s)	Model Data; Table Data with Predicted Outputs, Test Accuracy
_Program(s)/_Module(s) Called	Void

User Interface Module

_Name	**ServerDataToUI**
_Description	Obtains the desired data from the MySQL database
_Higher Level Unit	UI Module
_Input(s)	Data timestamp start; Data timestamp end; Table/Column(s) of Input(s); Model Data

_Output(s)	Table Data (and/or Model Data
_Program(s)/_Module(s) Called	ServerDataToScript<DataTransferring Module>

_Name	**HardDriveDataToUI**
_Description	Obtains the desired data from the hard drive
_Higher Level Unit	UI Module
_Input(s)	Data timestamp start; Data timestamp end; Table/Column(s) of Input(s); Model Data
_Output(s)	Table Data (and/or Model Data
_Program(s)/_Module(s) Called	HardDriveDataToScript<Data TransferringModule>

_Name	**LoadData**
_Description	Obtains the desired data from desire location
_Higher Level Unit	UI Module
_Input(s)	Data timestamp start; Data timestamp end; Table/Column(s) of Input(s); Source Location (Server or Hard Drive)
_Output(s)	Table Data
_Program(s)/_Module(s) Called	ServerDataToUI<UIModule> (or) HardDriveToUI<UIModule>

_Name	**PlotData**
_Description	Visually plots the data
_Higher Level Unit	UI Module
_Input(s)	Table Data
_Output(s)	Void
_Program(s)/_Module(s) Called	HardDriveDataToUI<UIModule> (or) ServerDataToUI<UIModule>

_Name	**LoadModel**
_Description	Loads the data for a specified model
_Higher Level Unit	UI Module
_Input(s)	Model to Load
_Output(s)	Model Data
_Program(s)/_Module(s) Called	HardDriveDataToUI<UIModule> (or) ServerDataToUI<UIModule>

_Name	**TrainModel**
_Description	Trains the loaded model with the chosen dataset
_Higher Level Unit	UI Module
_Input(s)	Model Data; Table Data
_Output(s)	Refined Model Parameters; Predicted Data Table

_Program(s)/_Module(s) Called	LoadModel<UIModule>; LoadData<UIModule>; TrainModel<ModelModule>

_Name	**TestModel**
_Description	Tests the loaded model with the chosen dataset
_Higher Level Unit	UI Module
_Input(s)	Model Data; Table Data
_Output(s)	Predicted Table Data; Model Accuracy
_Program(s)/_Module(s) Called	LoadModel<UIModule>; LoadData<IUIModule>; TestModel<ModelModule>

_Name	**ShowData**
_Description	Displays the loaded system data to the user
_Higher Level Unit	UI Module
_Input(s)	Table Data
_Output(s)	Void
_Program(s)/_Module(s) Called	LoadData<UIModule>

_Name	**ShowModel**
_Description	Displays the loaded model data to the user
_Higher Level Unit	UI Module
_Input(s)	Model Data
_Output(s)	Void
_Program(s)/_Module(s) Called	LoadModel<UIModule>

_Name	**ShowAccuracy**
_Description	Visualizes Accuracy through plotting of a graph
_Higher Level Unit	UI Module
_Input(s)	Table Data (two tables, predicted and actual)
_Output(s)	Void
_Program(s)/_Module(s) Called	LoadData<UIModule>

Data Structures

Data will be retrieved from MySQL databases, with tables being the primary data structure driving the platform. Since the models being trained and used will use different data sets with varying inputs and outputs, there is not one single database table that will be used. Each model will be using a different table that will include different sensor data inputs such as temperature and humidity. There will also be separate data tables used for training the models that will include the output data as well as the input data.

Format

Each grouping of data available from the database will start with the timestamp of when the data was captured. The data following the timestamp will be dependent on the model being used. The data can be of <int>, <bool>, or <float> types. See figures 4 and 5 for reference.

Timestamp	Sensory Data 1	Sensory Data 2	Sensory Data 3	Sensory Data 4	Sensory Data 5

Figure 4: Example of database table for a trained model to make predictions off.

Timestamp	Sensory Data 1	Sensory Data 2	Sensory Data 3	Sensory Data 4	Correct Output Data 1	Correct Output Data 2

Figure 5: Example of database table for training/testing a model.

Updated User Interface design

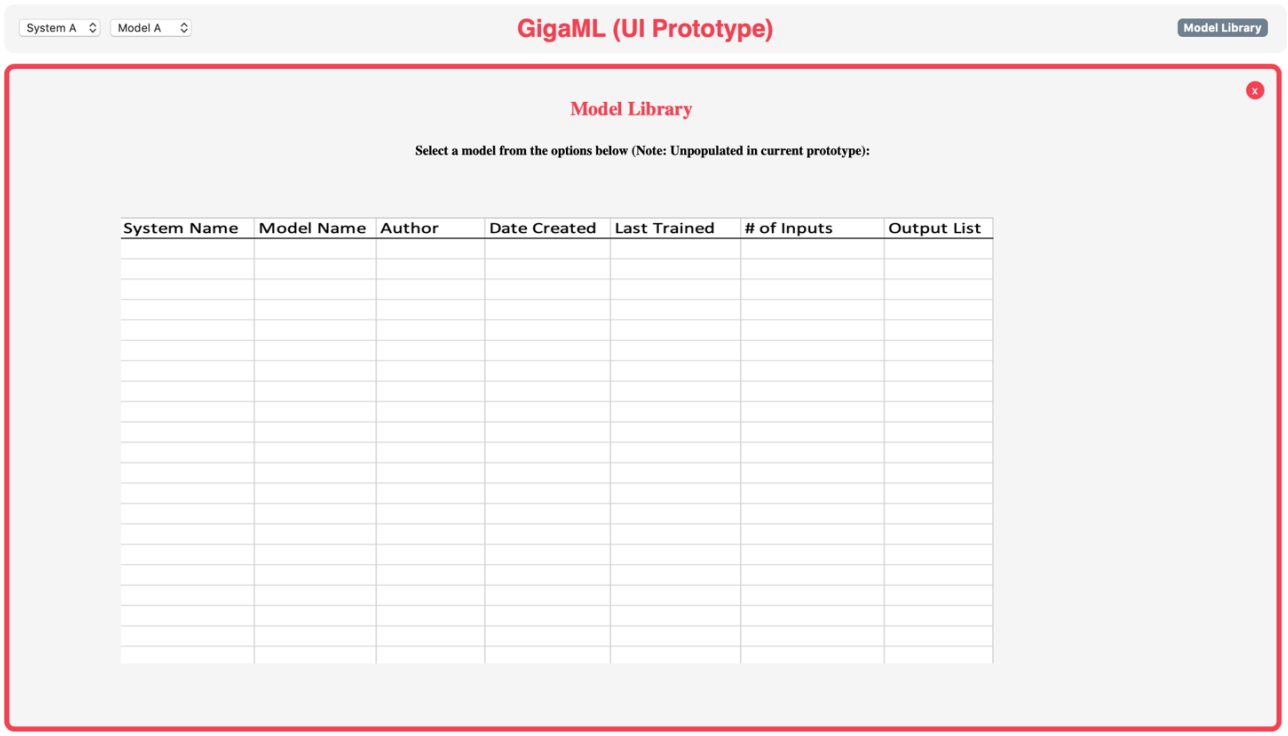


Figure 6: Screenshot of the model library.

Figure 6 shows the current landing page of the system. The user is immediately presented with a table representing all the models and relevant metadata saved in the platform. The user is expected to select one to proceed to the rest of the interface.

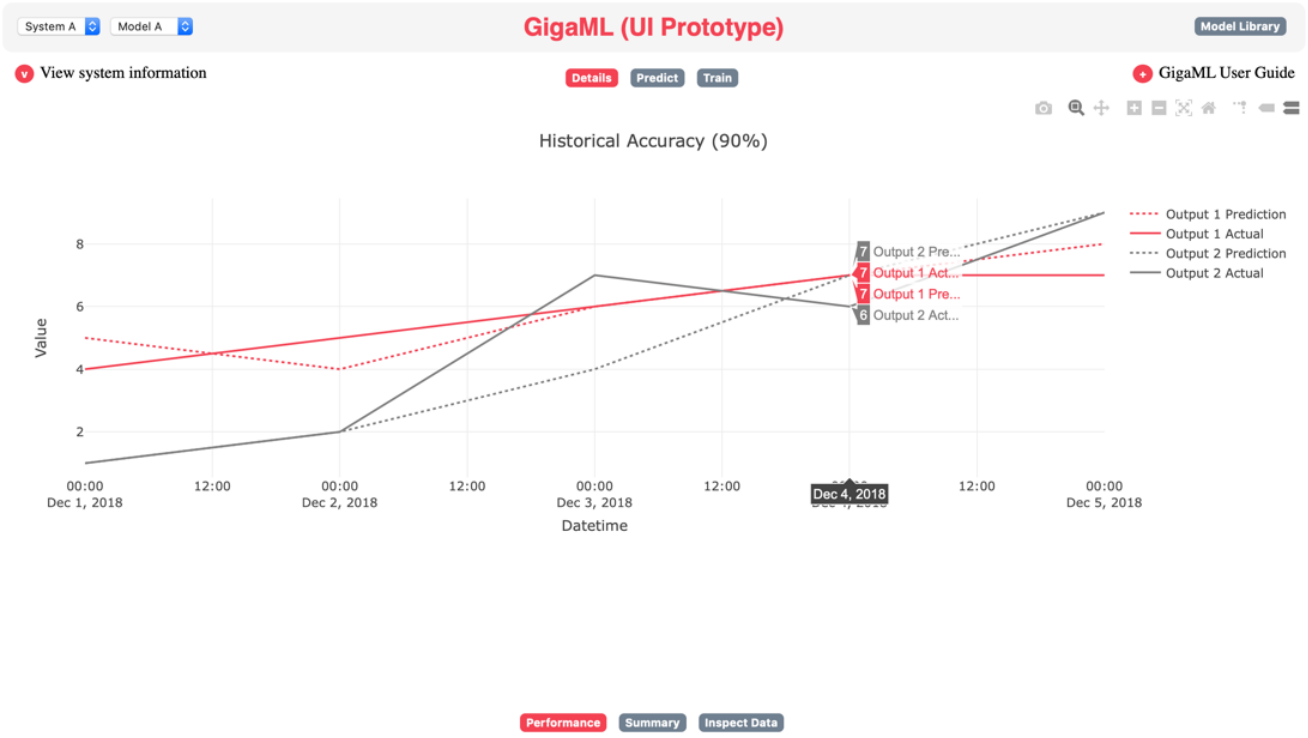


Figure 7: Screenshot of the Model Details – Performance page.

Figure 7 shows the Model Details – Performance page. It shows a graph displaying historical accuracy and a fake neural network graphic between changeable inputs and outputs. As in other portions of the interface, the user has drop-down options in the top left to change between models and systems.

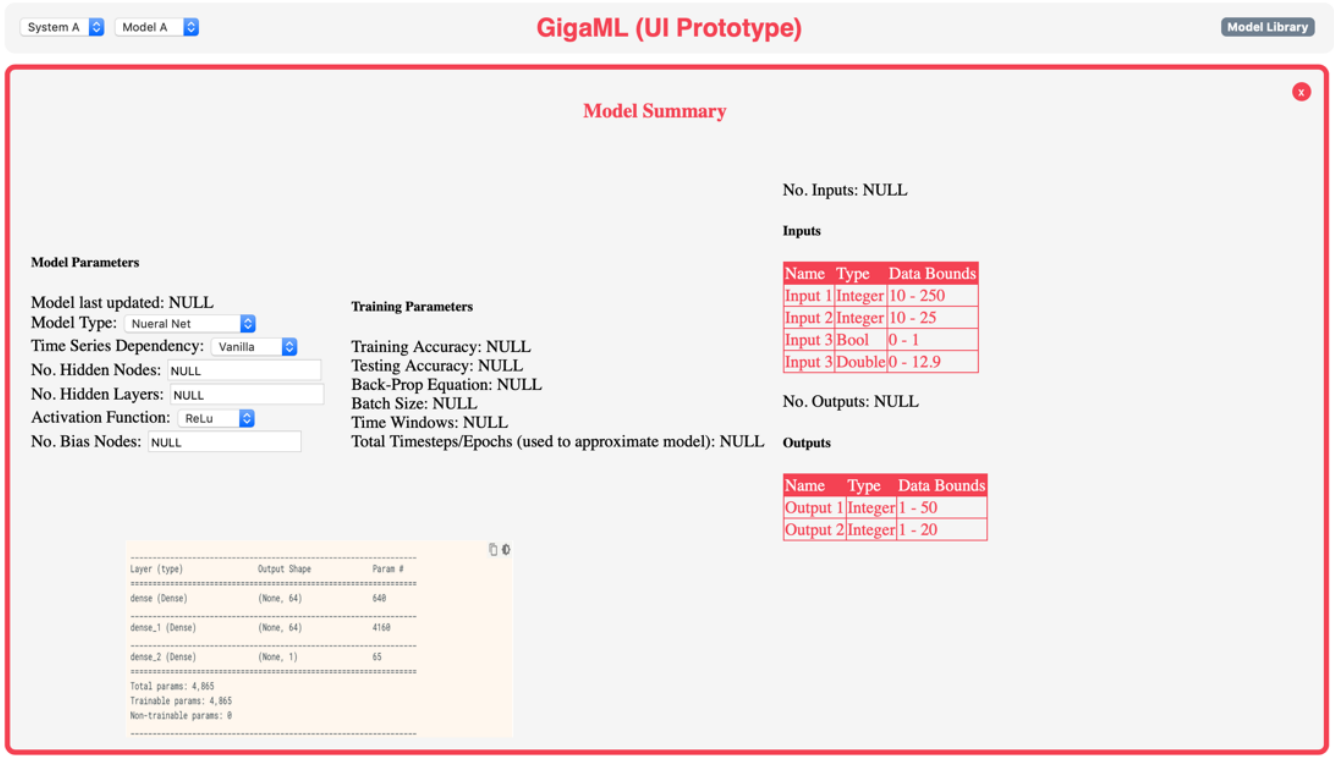
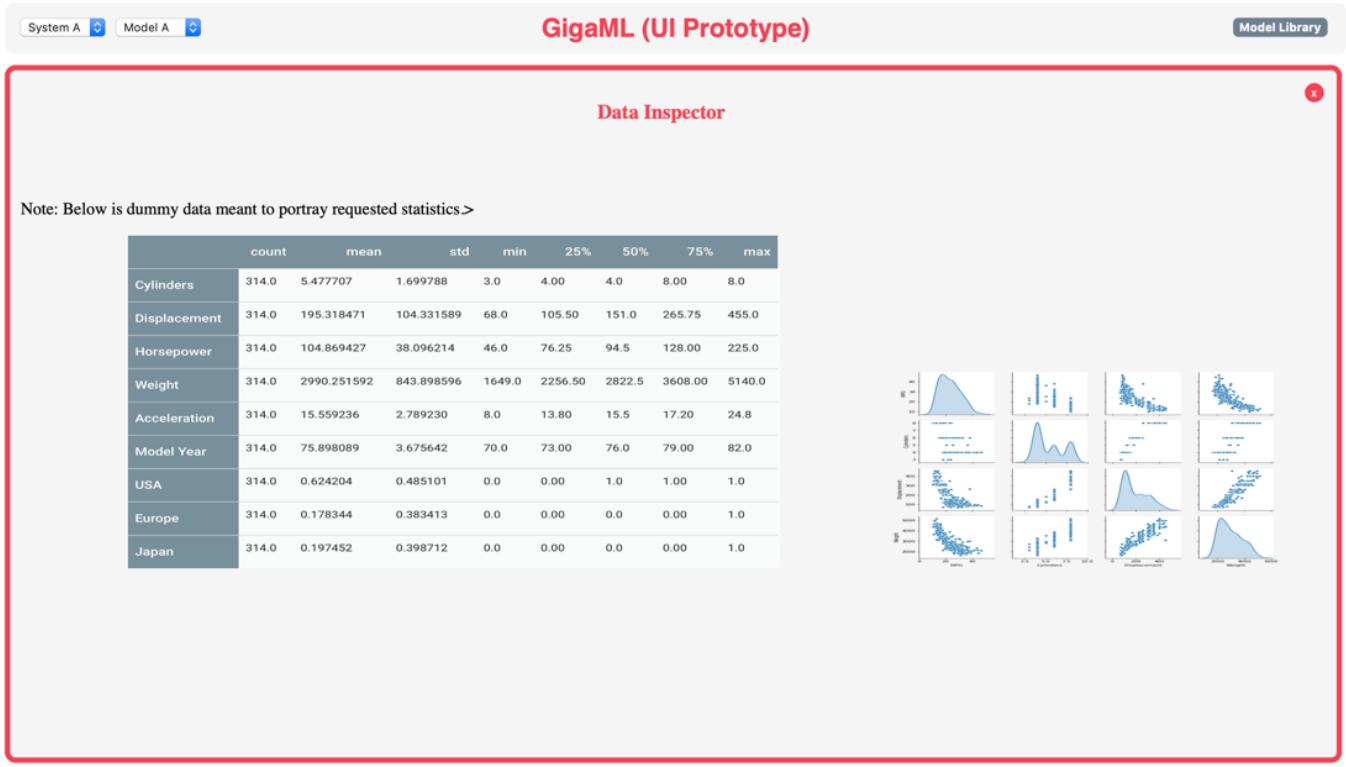


Figure 8: Screenshot showing the Model Details – Summary page.

Figure 8 shows Model Details - Summary. Here, all relevant neural network information is listed for the user. This includes details such as its inputs and outputs, model architecture, number of parameters, and more.



	count	mean	std	min	25%	50%	75%	max
Cylinders	314.0	5.477707	1.699788	3.0	4.00	4.0	8.00	8.0
Displacement	314.0	195.318471	104.331589	68.0	105.50	151.0	265.75	455.0
Horsepower	314.0	104.869427	38.096214	46.0	76.25	94.5	128.00	225.0
Weight	314.0	2990.251592	843.898596	1649.0	2256.50	2822.5	3608.00	5140.0
Acceleration	314.0	15.559236	2.789230	8.0	13.80	15.5	17.20	24.8
Model Year	314.0	75.898089	3.675642	70.0	73.00	76.0	79.00	82.0
USA	314.0	0.624204	0.485101	0.0	0.00	1.0	1.00	1.0
Europe	314.0	0.178344	0.383413	0.0	0.00	0.0	0.00	1.0
Japan	314.0	0.197452	0.398712	0.0	0.00	0.0	0.00	1.0

Figure 9: Screenshot of the Model Details – Data Inspector page.

Figure 9 shows Model Details – Data Inspector. This pane allows users to explore the input data that is feeding the currently selected mode. Useful statistics such as mean and standard deviation, as well as associated plots for visualizing data, are presented here.

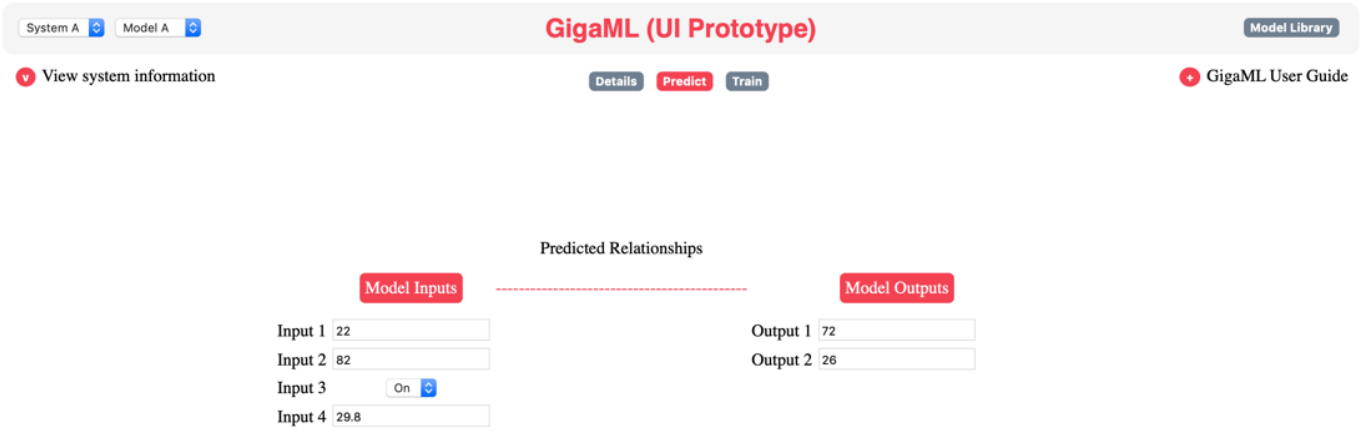


Figure 10: Screenshot showing the Model Predictions page.

Figure 10 shows the Predict pane. This is where users can use the previously trained models in a 'sandbox'-like environment. Various input and output values can be tested to see what the model would predict (bidirectionally).

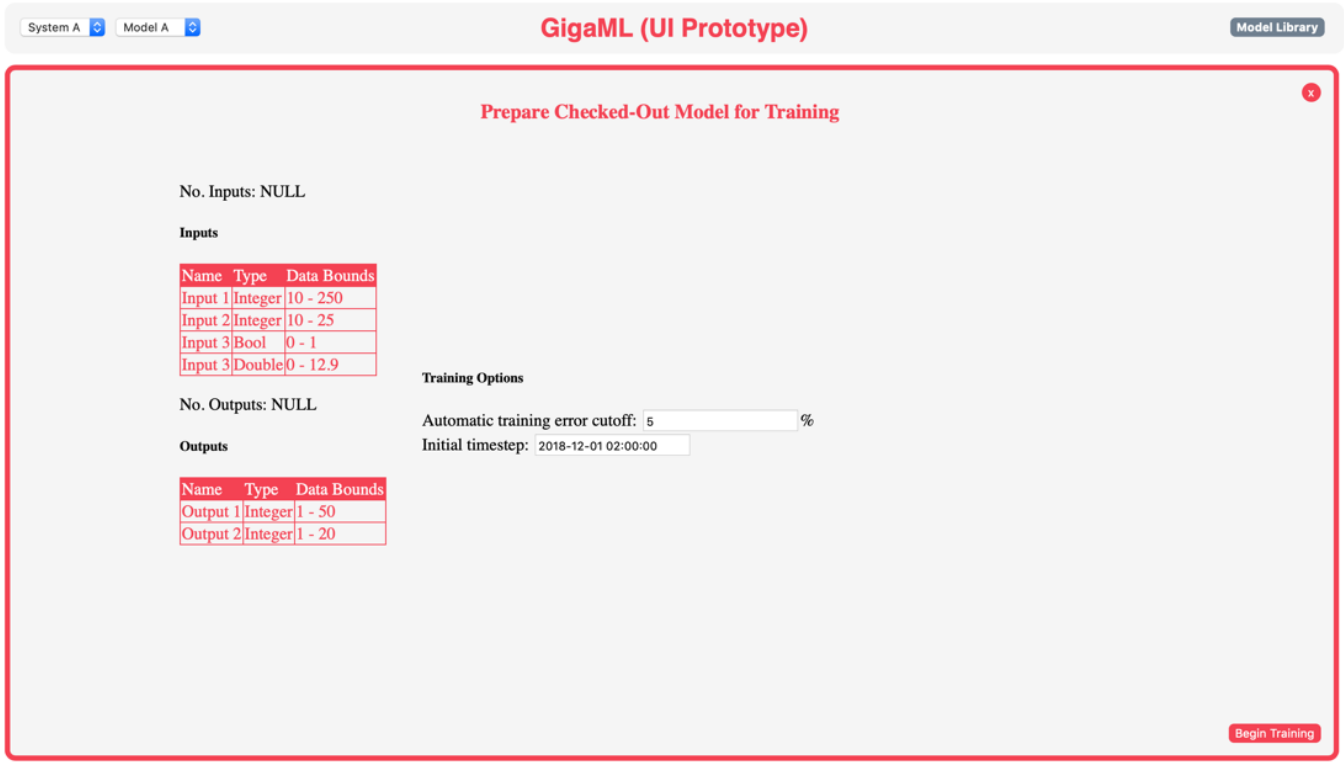


Figure 11: Screenshot showing the Check Out Model page.

Figure 11 shows the Check Out Model for Training page. The user can manually set training parameters to avoid error. The user can “Begin Training” at the bottom of the page, which leads into another panel that allows the user to monitor training.

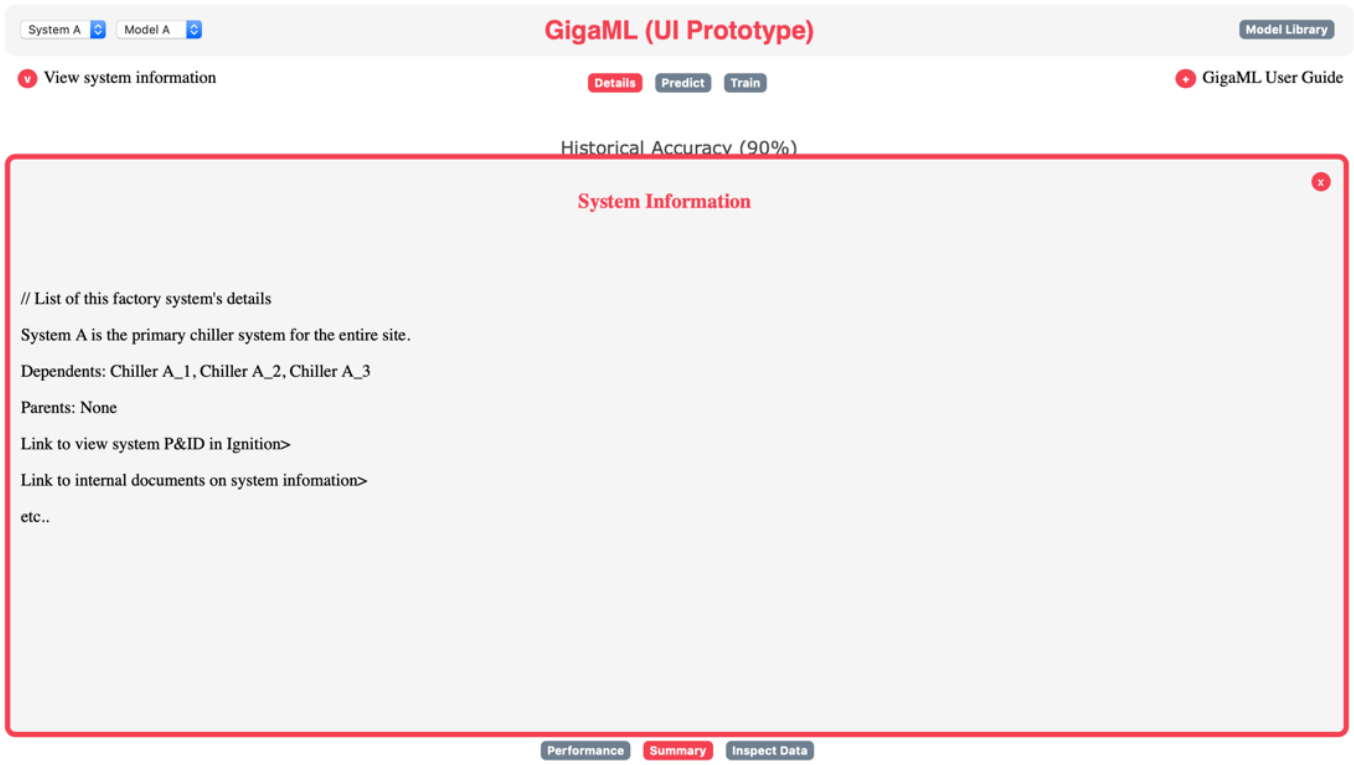


Figure 12: Screenshot showing the System Information page.

Figure 12 shows the "View system information" link from the top left of the default page. Here, the user can see a list of details about the system. This has yet to be fully visualized pending more information about the systems in place at the Gigafactory.

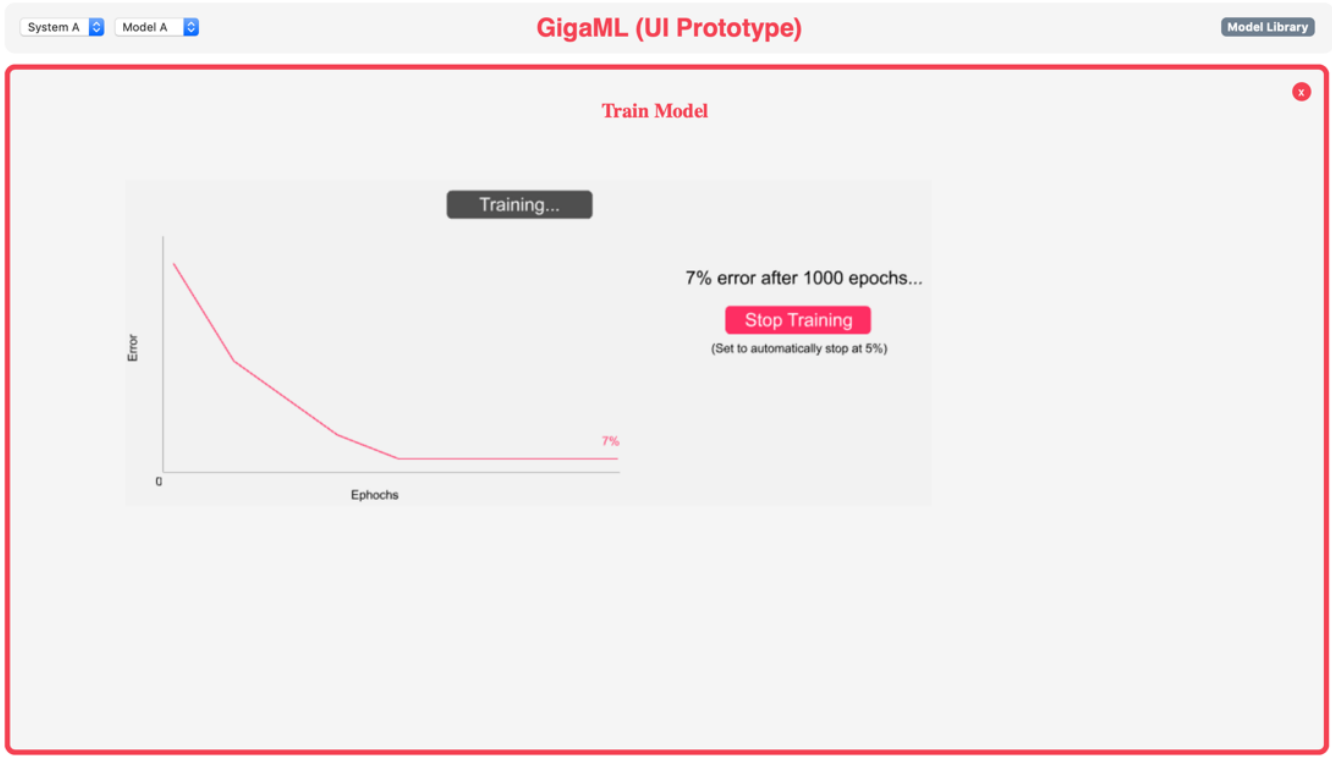


Figure 13: Screenshot showing the Train Model page.

Figure 13 shows the Train Model page. Here, the user is presented with a real time plot of training error that they can monitor. The user can also stop training from the button adjacent to the plot, once they are satisfied with the training progress.

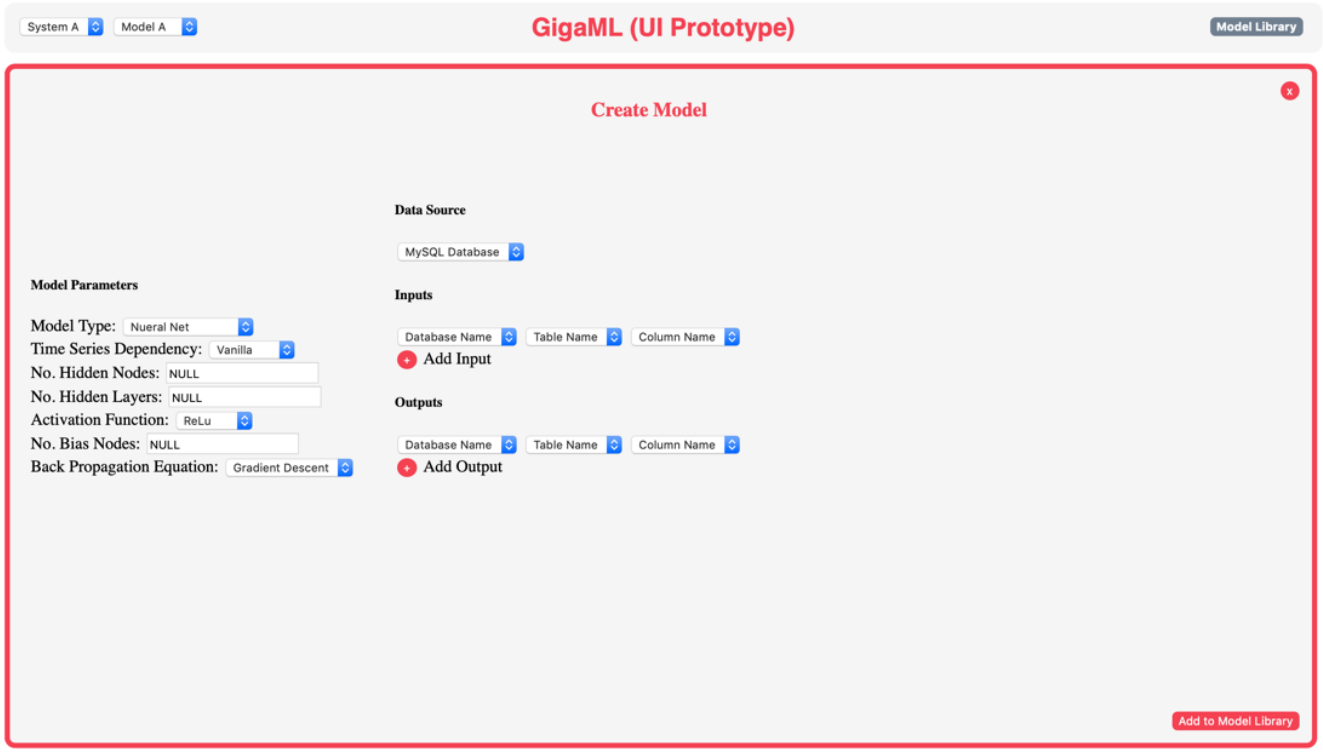


Figure 14: Screenshot showing the Create Model page.

Figure 14 shows the “Create Model” page. Here, the user can specify model specifics and variables for inputs and outputs. The user can also add inputs and outputs here if more are needed. This is a precursor to training the model.

Updated Glossary of Terms

1. **Active learning:** a form of semi-supervised machine learning where the algorithm can choose the data it wants to learn from; the program queries a programmer or labeled data set to learn the correct prediction for a given problem [1]
2. **Anomaly detection:** the identification of statistical outliers in a set of data; these items may result from contamination or errors in the data [2]
3. **Artificial intelligence:** the application of rapid data processing, machine learning, predictive analysis, and automation to simulate intelligent behavior and problem-solving abilities with machines; the intelligence of machines versus that of humans and animals [1]
4. **Backpropagation:** algorithm used to adjust each weight in a network in proportion to how much it contributes to overall error until a series of weights produce good predictions [3]
5. **Bayesian statistics:** a theory that states that the probability of something occurring in the future can be inferred by past conditions related to the event as opposed to by relative frequency in past samples [1]
6. **Cluster analysis:** a supervised learning technique that groups a set of unlabeled objects into clusters that are more similar to each other than the data in other clusters [1]
7. **Confusion matrix:** a "n-by-n" matrix where each row represents the true classification of a piece of data and each column represents the predicted classification or vice versa; it can be used to assess how accurately a model is classifying data and where it has problems [1]
8. **Deep learning:** a machine learning technique that constructs ANNs to mimic the structure and function of the human brain; uses multiple layers of nonlinear processing to extract features from data into different levels of abstraction [1]
9. **Dimensionality reduction:** mapping original high-dimensional data to a representation with less dimensions that captures the content of the original data [2]
10. **Genetic algorithm:** a technique in which a population of candidate solutions is mutated and selected from over the course of many cycles to find solutions to complex optimization problems [2]
11. **Gradient descent:** an optimization algorithm based on a convex function that tweaks parameters iteratively to minimize a function to its local minimum; used to find the values of a functions parameters that minimize a cost function as far as possible [3]
12. **Hopfield Networks:** a neural network that contains one or more recurrent nodes used for auto association and optimization tasks [4]
13. **Loss function:** functions used to determine the error, also known as loss, between the output of algorithms and the given target value; used to compute how accurate the output of a model is [1]
14. **Machine learning:** a field of computer science that aims to teach computers how to learn and act without being explicitly programmed; implemented by building models which allow programs to learn through experience [1]
15. **Markov chain:** used to model stochastic processes where each state has a certain probability of transitioning to the other states; predicts the next state based on the current state [1]
16. **Neural network:** also known as an artificial neural network (ANN); a computational learning system that uses a network of functions to understand and translate a data input from one form into a desired output [1]
17. **Neuron:** the basic building block of an ANN; inspired by biological neurons; consists of a piece of data and a collection of weights between itself and its connected neurons [1]

18. **Non-deterministic polynomial time:** also known as NP; a complexity class in theoretical computer science; a problem is NP if it is solvable in polynomial time by a non-deterministic Turing machine or if a proof can be verified in polynomial time by such a machine [1]
19. **Rectified Linear Units (ReLU):** a simple activation function used in deep learning models that returns zero if a negative input is received, and the actual value for any positive value received [1]
20. **Regularization:** a technique in machine learning that discourages learning a more complex or flexible model to avoid the risk of overfitting by constraining the parameter estimates towards zero [3]
21. **Reinforcement learning:** a type of unsupervised learning that seeks to incentivize computational agents to naturally learn correct decisions by trial and error and by using rewards [1]
22. **Semi-supervised learning:** a deep learning technique that labels some of the data in an AI's database but not all; with this method an ANN can infer what unlabeled data represents with better accuracy that in unsupervised learning but with less cost than in supervised learning [1]
23. **Supervised learning:** a class of systems and algorithms that extrapolate a function from known input and output data; by evaluating many samples of input and the corresponding output, the systems form models to evaluate the input [1]
24. **Unsupervised learning:** a technique where all input is unlabeled, and the algorithm must structure the input on its own [1]
25. **Weight:** in an ANN, a parameter associated with a connection between two neurons; corresponds to a synapse in a biological neuron; determines the extent to which the output from the first neuron factors into the output of the second neuron [1]

Engineering Standards and Technologies

Name/Abbreviation: Python3.6
Brief Description: A scripting language with a well-supported community. Python has many available libraries for creating machine learning models, preprocessing data, and creating backends for web apps.
S**tandard or Technology:** <Technology>
Use in Project: Python3.6 will be used to create the backend for GigaML. Any processing of data will be done using Python scripts as will any machine learning model creation.

Name/Abbreviation: Flask
Brief Description: A microframework for Python used to create web applications while allowing for large amounts of customization since Flask only provides the basic necessities.
Standard or Technology: <Technology>
Use in Project: Flask will be used to interact with the frontend webpage to get user input and send data information back to the frontend.

Name/Abbreviation: TensorFlow
Brief Description: Open-source Python3 and JavaScript library for dataflow programming as well as a symbolic math library. TensorFlow is commonly used for machine learning applications.
Standard or Technology: <Technology>
Use in Project: TensorFlow will be used for some data preprocessing (such as normalization of data) as well as for the dynamic creation and testing of neural networks.

Name/Abbreviation: SciKit Learn
Brief Description: SciKit Learn is a machine learning library for Python that has efficient data preprocessing.
Standard or Technology: <Technology>
Use in Project: SciKit Learn will be used for preprocessing of data on the backend of GigaML.

Name/Abbreviation: Pandas
Brief Description: A Python library for easy to use data structures and data analysis tools.
Standard or Technology: <Technology>
Use in Project: Pandas will be used to organize the data obtained from .xlsx files, csv files, and data from either Ignition or the MySQL server. Pandas will also be used to gather general information about that data, such as indexes, means, etc.

Name/Abbreviation: HTML
Brief Description: The standard markup language for creating webpages and web applications. Will be used in conjunction with CSS, JavaScript, and .NET.
Standard or Technology: <Technology>
Use in Project: Will be used to create the frontend of the GigaML application in conjunction with CSS, JavaScript, and .NET.

Name/Abbreviation: CSS
Brief Description: Used to describe how to format HTML elements and how they should be displayed.
Standard or Technology: <Technology>

Use in Project: Will be used to create the frontend of the GigaML application in conjunction with HTML, JavaScript, and .NET.

Name/Abbreviation: JavaScript
Brief Description: High level interpreted programming language that is dynamic, weakly typed, and prototype-based. Mainly used in web applications.
Standard or Technology: <Technology>
Use in Project: Will be used to create the frontend of the GigaML application in conjunction with HTML, CSS, and .NET. Will also be used with TensorFlowJS to create a real-time ML application within GigaML.

Name/Abbreviation: .NET
Brief Description: Framework developed my Microsoft that provides language interoperability across several programming languages.
Standard or Technology: <Technology>
Use in Project: Will be used to create the frontend of the GigaML application in conjunction with HTML, CSS, and JavaScript.

Name/Abbreviation: MySQL
Brief Description: A database management system that is in use at the Tesla Gigafactory holding data from the factory.
Standard or Technology: <Technology>
Use in Project: MySQL will be used to store and retrieve data from Tesla's servers to both the frontend and backend of GigaML.

Name/Abbreviation: UML
Brief Description: A general purpose, developmental, modeling language that provides a standard way to visualize the design of a system.
Standard or Technology: <Standard>
Use in Project: UML was used to visualize the design of the frontend, backend, and server-to-application piece.

Name/Abbreviation: JSON
Brief Description: An open-standard file format that uses text to store data objects consisting of attribute-value pairs and array data types.
Standard or Technology: <Standard>
Use in Project: JSON will be used as the format data will be stored in when sent between the frontend and backend of GigaML.

Updated List of References

Project Domain Books

A Brief Introduction to Neural Networks
Kriesel, David. A Brief Introduction to Neural Networks. No Publisher, 2007. Available at http://www.dkriesel.com.

This textbook serves as a concise introduction to neural networks. As opposed to the other text, it focuses exclusively on neural networks and goes into great detail on the history, conceptual underpinnings, applications, and construction of this type of system.

Pattern Recognition and Machine Learning
Bishop, Christopher M. Pattern Recognition and Machine Learning. Springer Science + Business Media, LLC, 2006.

This textbook is a general introduction to machine learning and pattern recognition aimed at advanced undergraduates and first year graduate students. It covers the mathematical and statistical concepts underlying machine learning as well as several different models that can be employed.

Reference Articles

Beginner's guide to neural networks
Illingworth, W.T. Beginner's guide to neural networks. Proceedings of the IEEE National Aerospace and Electronics Conference, 1989. DOI: 10.1109/NAECON.1989.40352

This conference proceeding discusses neural networks and their use as a solution to several different classes of problems. It gives a history of this type of model and illustrations of the basic parts and processes that make it up. Finally, it discusses several major paradigms for the construction of neural networks along with strengths and limitations of each type.

Deep convolutional neural networks for LVCSR
Sainath, Tara N.; Mohamed, Abdelrahman; Kingsbury, Brian; Ramabhadran, Bhuvana. Deep convolutional neural networks for LVCSR. IEEE International Conference on Acoustics, Speech and Signal Processing, 2013. DOI: 10.1109/ICASSP.2013.6639347

This conference proceeding discusses convolutional neural networks (CNNs) as an alternative to deep neural networks (DNNs) in the problem of large vocabulary continuous speech recognition (LVCSR). This problem is analogous to the one that our group will be facing because at the Tesla factory there will be a large number of states which are constantly changing, and which need continuous analysis.

Enhancing Spindle Power Data Application with Neural Network for Real-Time Tool Wear/Breakage Prediction during Inconel Drilling
Corne, Raphael; Mohamed El Mansori, Chandra Nath; Kurfess, Thomas. Enhancing Spindle Power Data Application with Neural Network for Real-Time Tool Wear/Breakage Prediction during Inconel Drilling. Procedia Manufacturing, Volume 5. 2016. DOI: 10.1016/j.promfg.2016.08.004

The authors of this journal article attempt to use a neural network to process real time data during industrial drilling to predict tool wear and breakage before it happens. The method employed by these researchers was highly successful. After training, the model was able to predict measured data highly accurately in a significant majority of samples.

Websites

https://www.deepai.org
[1] “Deep AI.” Website. Retrieved October 27, 2018 from https://www.deepai.org.

This website has a large amount of information on artificial intelligence and machine learning, in large part focusing on neural networks, the topic of our project. In addition to the definitions provided in the website’s glossary, the history and applications of a majority of the concepts are also explained.

https://colab.research.google.com
[2] “Colaboratory.” Website. Retrieved October 27, 2018 from https://colab.research.google.com.

Colaboratory is a Jupyter Notebook environment that runs in the cloud. Normally, Jupyter needs to be installed, but with Colaboratory, it can be run entirely from a browser. An additional benefit is that Colaboratory is integrated with Google Drive so that the members of our group can all collaborate on the project together.

https://www.tensorflow.org
[3] “Tensor Flow.” Website. Retrieved October 26, 2018 from https://www.tensorflow.org.

TensorFlow is an open-source machine learning library. This website contains the resources to download this library and its APIs. In addition, there are more than twenty different tutorials and guides which explain how to use the library and show it in action.

http://flask.pocoo.org
“Flask” Website. Retrieved February 10, 2019 from http://flask.pocoo.org.

Flask is a microframework based on Jinja 2 and Werkzeug for creating python-based web applications. This website contains the basic setup, download of the latest release, and the documentation.

Team Contributions

Team 15 worked together to create the Revised Specification and Design document with the understanding of group effort. A one-hour long meeting was held between the team members prior to creating the document, so all members understood each section goal and outline project development. Table 4 is a representation of the time worked on each section:

Team Member	Sections	Time Worked
Adam Cassell	Abstract, Updated Design	5 hours
Ashlee Ladouceur	Updated Specification, Updated Glossary of Terms, Team Contributions	5 hours
Braeden Richards	Recent Project Changes, Engineering Standards and Technologies, Updated List of References	5 hours

Table 4: Detailed breakdown of time per section by each member of Team 15 for the Revised Specification and Design document.