NEURAL NETWORK BASED DIGITAL TWIN FOR A HALF-BRIDGE LLC RESONANT CONVERTER

by

Nicholas Green

A Thesis

Submitted to the University at Albany, State University of New York

in Partial Fulfillment of

the Requirements for the Degree of

Masters of Science

College of Nanotechnology, Science, and Engineering

Department of Electrical and Computer Engineering

Spring 2024

ABSTRACT

This paper introduces a Neural Network based approach for creating a Digital Twin of a Half-Bridge LLC Resonant Converter. By using a set of measured inputs, the Digital Twin system can characterize the component values, and hard to measure switch characteristics such as the Junction Temperature or Gate-Source Capacitance. This capability is useful for reliability or health applications, and may also be useful for adaptive control. A case study is done for the LLC Resonant converter designed around an operating point. Data sweeps around that operating point are simulated to form a dataset. An introduction to Neural Networks is given before determining the input vector for each output parameter. The results and final best design of each Neural Network will be discussed with experiments demonstrating their performance. The range of the Mean Absolute Errors for all parameters was from 0.3% to 7%, with an average over all parameters of 2.3%. Predictions on the boundaries of the dataset proved difficult, while the midpoints were fit without issue.

ACKNOWLEDGMENT

I would first and foremost, like to thank my advisor, Professor Mohammed Agamy for his guidance during my thesis. It was him who first taught me the basics of Power Electronics but now goes beyond as my thesis advisor. His classes were challenging and filled with power engineering knowledge beyond just the textbook. Thank you for your guidance over the semesters. I also want to thank Professor Hany Elgala, and Professor Nathan Dahlin for their aid as part of the thesis committee. Finally, a few extra thanks for Professors Jonathan Muckell who encouraged my graduate studies and was am amazing professor to have and work with, Professor James Moulic who taught me everything about the digital side of electrical engineering from logic gates to caching policies, and Professor Vivek Jain who encouraged my interest in physics during my minor and guided my independent study of radiation detectors. Thank you all for making my time at Albany that much better.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Digital Twins

The idea of a Digital Twin (DT) is to have a digital mirror of a physical system. Ideally, the system can update in real-time, with high resolution and accuracy. When applied to power electronic converters, a DT reveals information about a converter's parameters and its operation, while minimizing the need for sensor hardware. In general, previous work has focused on using DTs to monitor converter health, characterize converter operation, or tune control parameters adaptively. Having this digital reflection of a physical system is perhaps most useful in reliability prediction and lifetime estimation given how difficult those processes are to model.

1.2 Previous Work

Previous literature has employed a variety of techniques to create DTs ranging from standard modeling, to Machine Learning, or real-time swarm optimization. Both [1, 2] apply a classical Averaged Modeling technique for its simplicity as well as adding a real-time FPGA implementation. A simple condition monitoring technique is even demonstrated in [2] by comparing the real and DT systems to determine if abnormal operation is occurring. A focus on adaptive control can be seen in [3] that uses a digital twin in conjunction with a Proportional-Integral (PI) gain tuner to optimize operation. This paper uses linear transfer functions and state-space equations for the DT modeling.

The next class of DT models are based on statistical modeling techniques. A Bayesian Optimization approach is used in [4] to characterize the resistance, inductance, capacitance,

parasitic resistances, and MOSFET On-State resistance of a Buck and Boost in an online application. By characterizing the circuit in real-time, the drift of components can be estimated which could be applied for control or repair of converters. In [5], Polynomial Chaos Expansion is used to transform state-space models into stochastic models. Components are assumed to be random variables defined by their tolerances. A numerical solver is implemented on an FPGA to make this system real-time. By modeling the converter and controller, basic condition monitoring can be done by flagging a deviation in DT and real behavior.

Another strategy used to characterize a DT model is optimization heuristic algorithms. These algorithms are used to search large solutions spaces but have no guarantees of convergence to a solution. A population-based technique called Particle Swarm Optimization (PSO) is used by [6, 7, 8, 9, 10]. PSO works by having a population of guess solutions that work together to search a solution space define by an objective function. An individual in this swarm balances between its own best solution, the group's best solution, and its current velocity in the search space. Generally these papers try to characterize component values of the converter for health monitoring or control. PSO can constantly search for solutions as the real converter data is gathered and compared to the DT. In contrast to PSO, one paper uses a technique called Arithmetic Optimization Algorithm (AOA)[11]. AOA makes use of simple arithmetic operations (Addition, Subtraction, Multiplication, and Division) to search over a space. It was shown in [11] that AOA was faster compared to PSO for a Buck Converter model.

Finally, a number of papers have implemented Machine Learning approaches, namely Neural Network variants. An Artificial Neural Network (ANN) is used in [12, 13]. The former paper inputs four PI parameters into the ANN to predict the error between reference and actual active and reactive power. This can be used to tune the control parameters to reduce control error. The latter paper uses an ANN to tune a DT model for accurate predictions. The method of Bayesian Regularized ANN (BR-ANN) and Random Forest

(RF) machine learning techniques were applied in [14]. The BR-ANN was trained to predict transient behavior such as settling time, overshoot, rise time, and steady state value. The RF was used to predict steady state ripple through a series of decision trees. One critique is that all of these techniques must look at a single snapshot of inputs and try to predict the outputs. The problem is converters are dynamic systems which might not fit well into the traditional ANN feed-forward approach. To try and add a dynamic view to the ANN model, methods such as Nonlinear AutoRegressive eXogenous (NARX-ANN) in [15] or a NARX with a Recurrent Neural Network (NARX-RNN) in [16] are used. These models take in inputs over a window of time to predict the converter's outputs building a sense of time into the machine learning model itself.

1.3 Approach

Many of the papers seen here use a buck or boost as opposed to more complex topologies. Moreover, the papers that try to predict converter parameters tend to simplify the switch greatly by treating it as ideal or only simulating the On-State Resistance. Just looking at the Machine Learning papers, none of them characterize the internal parameters but instead focus on control or output prediction applications. Something this thesis will add is the application of machine learning to characterize a converter's internal parameters, especially the switch's parameters, as well as testing the idea on a converter more complex than a buck or boost. Therefore, in this thesis, a series of Neural Networks are used to try and fully characterize the internals of a Half-Bridge LLC Resonant Converter. By measuring a set of key parameters, the converter's passives, parasitic resistances, switch capacitances, switch junction temperature, and Turn-On Zero Voltage Switching states can be estimated for use in condition monitoring or control. To do this, a converter design is established and simulations are run to collect data. Finally, the training and tuning of the neural networks are completed and tested.

1.4 Outline

The thesis will be organized as follows starting with Chapter 2. This chapter covers the selection, modeling, and design of the converter. Furthermore, basic converter behavior is explained and a frequency control rule is derived to properly operate the converter. Chapter 3 begins the discussion of data collection methodology required to power such a data driven solution. Details covering the simulations ranges, procedures, and data handling will be discussed. Moving on to Chapter 4, the core machine learning principles are introduced. The terminology and methods used to train and verify neural networks are discussed followed by the methods used to tune the networks. Chapter 5 goes over the DT input selection for different predicted parameters. The aim of this chapter is to produce a minimal set of inputs that are the least complicated to measure such that the converter can be characterized by the DT's outputs. Chapter 6 goes over the results of the best networks, and discusses what can be learned from the experiments. Finally, Chapter 7 concludes the thesis followed by a brief discussion on future directions for neural network based DTs.

CHAPTER 2

Converter Design

2.1 Converter Selection

A popular converter is the Solid State Transformer (SST) which performs the task of a regular transformer but can have advantages in size, weight, and power density [17]. Because of the solid state design, more can be done than with just a traditional transformer such as voltage or current regulation for compensation, or fault current limitation [17]. One example application uses a SST to take 7.2kV AC line voltage and transform it into 120/240V AC and a 400V DC output [18]. At the core of any SST is a DC-DC resonant converter, which will define the model circuit the proposed DT will emulate. In particular, a Half-Bridge LLC Resonant Converter shown in Fig. 2.1. This circuit provides a number of opportunities for internal predictions useful for monitoring surrounding the two switches and resonant tank. The converter itself is more complex than the typical Buck or Boost used in other papers. Combining this with a more detailed switch model enables experimentation with

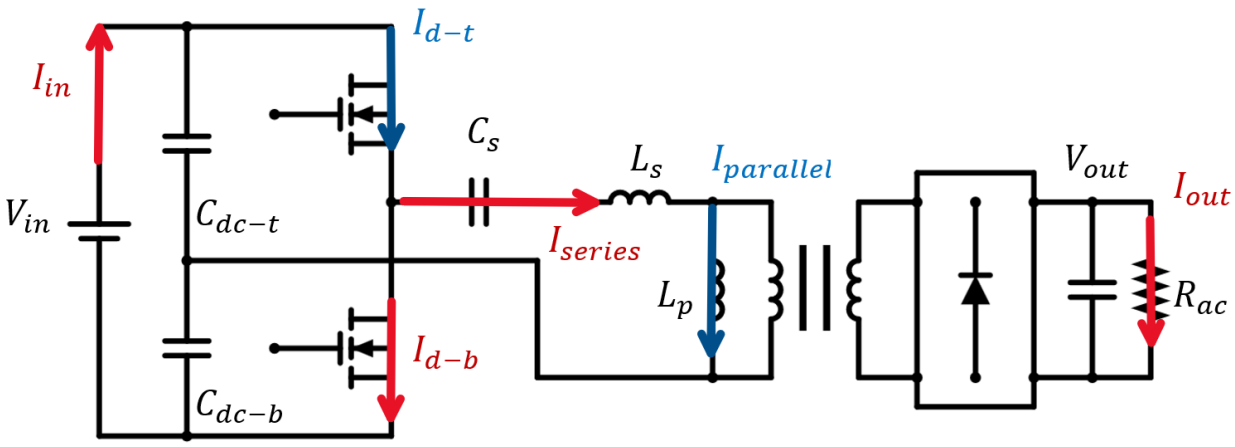

Figure 2.1: Half Bridge LLC Resonant Converter

more complex predictions about the switch beyond just On-State resistance. Moreover, experiments about the switch capacitances (Gate-Source and Drain-Source capacitors) and simple thermal models for the junction temperature are performed. By knowing more details about the switch, a mix of health monitoring and control can be used to extend the lifetime of a converter. This will be an important concern for SSTs if they see wide adoption as replacements for conventional transformers. The concepts here can always be extended to more complex circuits or even just the Full-Bridge or Dual-Active Bridge version. The model here is simply used as a case study to demonstrate the concept.

2.2 Converter Behavior and Modeling

First, the converter behavior must be understood to properly model it. The converter essentially has three stages. The first stage converts the DC input voltage into an AC square wave, this is the half-bridge's job. The second stage is the resonant tank which has a voltage gain controlled by the frequency of its input. Finally, the last stage will rectify and filter the AC back to a DC output voltage. The DC to AC stage consists to two DC-Link capacitors and two switches. The DC-Link capacitors will each store half the input voltage such that their intersection will also be at half the input voltage. The resonant tank is an LLC series-parallel design. It contains a series capacitor (C_s) and inductor (L_s), as well as a parallel inductor (L_p). This parallel inductor is generally designed into the transformer as the magnetizing inductance, while the series inductor is also designed into the transformer as leakage inductance. Figure 2.1 contains the symbol name schematic references.

The converter operations and behavior will be explored to begin. To produce an AC wave, the switches must operate 180° out of phase. In reality the switches have a small percentage of the period used for deadtime. This is when both switches are off which ensures no shorting of the input through the two switches occurs. This is also used for soft-switching discussed later on. Figure 2.2 displays example waveforms for the converter. The top two are

the switch voltage waveforms that align with the switch control signals. If the switch is on, its voltage is near zero. The out of phase switching behavior can be observed here as well. Figure 2.3 shows the current flow when the top switch is on. If only the top switch is on, then current flows through the top switch, into the resonant tank, and back to the switch through the DC-Link capacitor (C_{dc}). The two DC-Link capacitors are there to provide return paths from the resonant tank but as a result the capacitors split the input voltage in half. This means the resonant tank sees half the input voltage when the top switch is on. It is also true that the source will push current through the top switch but that isn't shown here. Figure 2.4 shows the behavior when only the bottom switch is on. In that case, the current flows through the bottom switch, through the resonant tank, and back through the bottom DC-Link capacitor. Note that when only the bottom switch is on, the resonant tank input sees the input side ground on the top wire, and the bottom DC-Link capacitor voltage at the bottom wire. As each DC-Link capacitor is charged to half the input voltage, the resonant tank sees negative half the input voltage. The third wave in Fig. 2.2 shows the full AC square wave aligned with the switches. This wave oscillates between plus and minus half the input voltage. The last two waveforms in the figure show the top and bottom switch currents respectively. When the switch is off, the current is a flat zero, and when the switch is on, it conducts the resonant current.

2.2.1 Converter Modeling

Fully modeling the resonant tank's gains for all the frequencies in the square wave, and then modeling the rectification and filtering of those signals will be cumbersome and likely contain negligible harmonics. Instead, the transformer, rectifier, and output load are modeled as a single resistor parallel to the magnetizing inductance. The methods developed for this modeling technique are found in [19]. This paper takes the point of view of looking into the rectifier stage given the assumption the output voltage is constant. This assumption is reinforced by the large output filter capacitor to dampen voltage swings. If the voltage on the

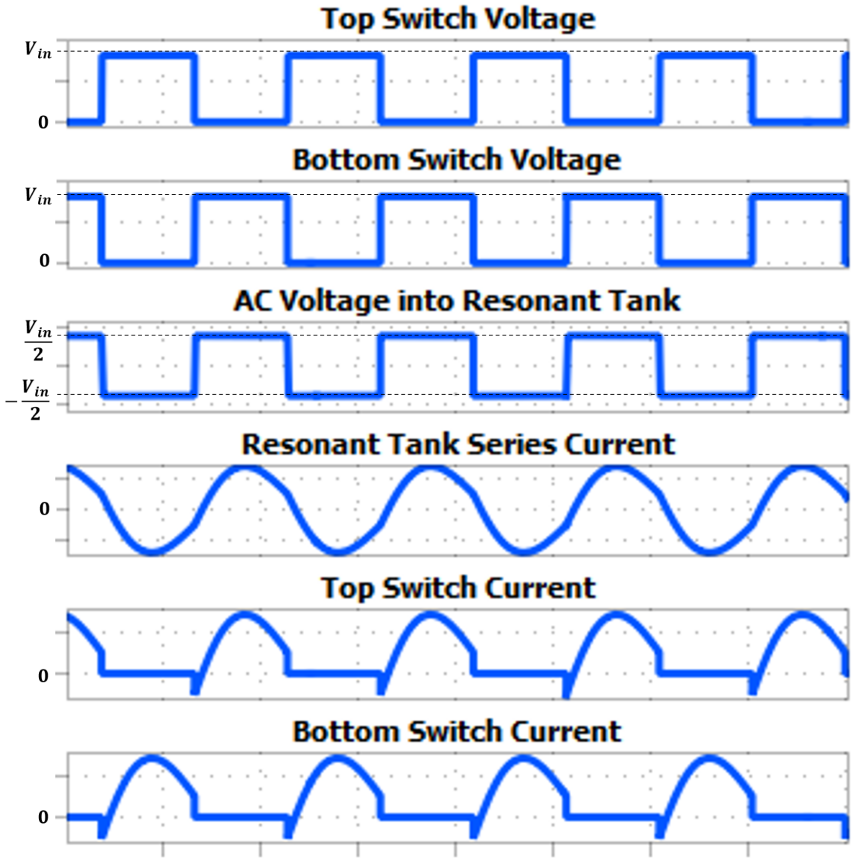

Figure 2.2: LLC Converter Waveforms

output side is constant, then the voltage before the rectifier is expected to be a square wave. This because going backwards through a rectifier "reverses" the rectification back into AC. As the output voltage is constant, this wave must be an AC wave with constant magnitude or simply a square wave. To make the analysis simpler, the paper approximates the square wave using only its fundamental frequency. Furthermore, the current seen before the rectifier will be a fundamental frequency approximation too. In resonant converters, the current waveform will generally take on a sinusoidal form. The AC fundamental approximation of the square wave is found using the first harmonic Fourier coefficient of $\frac{4}{\pi}$. The approximate AC voltage (V_{eq}) will have an RMS determined by (2.1) where V_{out} is the DC output voltage. Similarly, a rectifier acts to scale the RMS of a sinusoid by $\frac{2}{\pi}$. Using this fact, the RMS of the input current is described by (2.2) where I_{out} is the constant output current through the load.

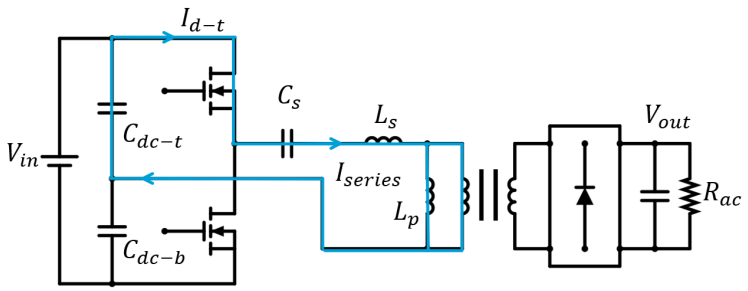

Figure 2.3: LLC Top Switch On Conduction Path

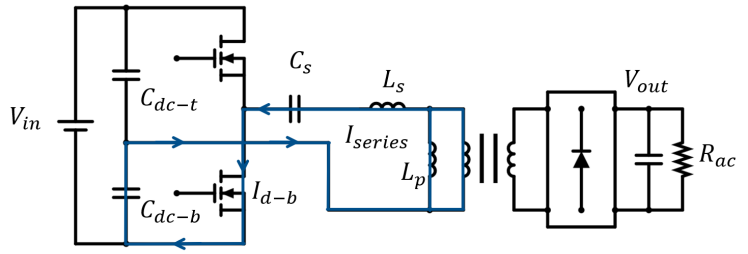

Figure 2.4: LLC Bottom Switch On Conduction Path

$$V_{eq} = V_{out} \cdot \frac{4}{\pi} \cdot \frac{1}{\sqrt{2}} \tag{2.1}$$

$$I_{eq} = I_{out} \cdot \frac{1}{\sqrt{2}} \cdot \frac{\pi}{2} \tag{2.2}$$

Where:

V_{out} is output voltage in Fig. 2.1

V_{eq} is the RMS AC fundamental of V_{out}

I_{out} is the output current in Fig. 2.1

I_{eq} is the RMS AC fundamental of I_{out}

Ohm's law is used to find the equivalent resistance of the rectifier and load combination. The final step is to scale the resistance by the turns ratio of the transformer (N) leading to

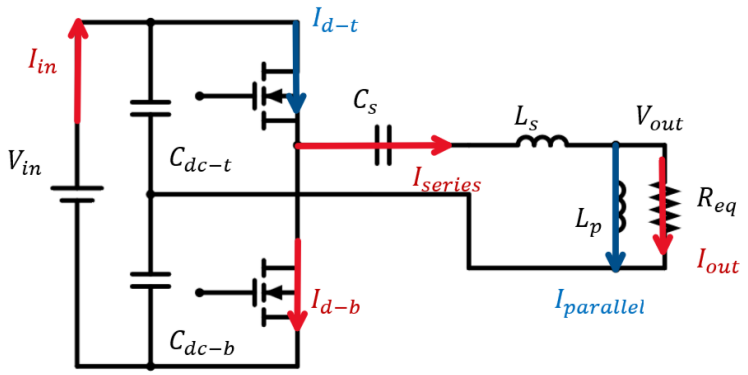

Figure 2.5: Simplified Converter with Equivalent Resistor

the final relationship of (2.3). Figure 2.5 shows the simplified converter schematic.

$$R_{eq} = \frac{V_{eq}}{I_{eq}} N^2 = \frac{8}{\pi^2} N^2 R_{ac} \tag{2.3}$$

Where:

N is transformer Turns ratio

R_{ac} is output resistor in in Fig. 2.1

R_{eq} is the equivalent resistor of the real load resistor transformed back through the rectifier and transformer

Continuing with the theme of AC fundamental approximations, the AC Square wave produced by the half-bridge can be approximated by its fundamental frequency such that simple AC analysis to be used for the whole converter. The gain of the resonant tank is found using a voltage divider equation to get the result of 2.4. This can be further expanded into the form of 2.5.

$$G = \frac{jX_{L_p} \parallel R_{eq}}{jX_{L_p} \parallel R_{ac} - jX_C + jX_{L_s}} \tag{2.4}$$

$$G = \frac{1}{\left(1 - \frac{1}{\omega_s^2 C L_p} + \frac{L_s}{L_p}\right) + j\left(\frac{\omega_s L_s}{R_{eq}} - \frac{1}{\omega_s C R_{eq}}\right)} \tag{2.5}$$

Where:

G is gain of the resonant tank's output voltage to input voltage

ω_s is angular switching frequency equal to $2\pi f_{sw}$

X_{L_p} is L_p's reactance defined by $\omega_s L_p$

X_{L_s} is L_s's reactance defined by $\omega_s L_s$

X_{C_s} is C_s's reactance defined by $\frac{1}{\omega_s C_s}$

One problem with using the form of (2.5) is that it directly dependents on the component values and chosen switching frequency. Often, it is useful to parameterize the gain equation with a proposed natural frequency (ω_0) and Quality-Factor (Q). The natural frequency is used to describe a system's "preferred" frequency, or in this case the resonant frequency of the resonant tank. The Quality-Factor describes how damped a system is through the ratio of stored to lost energy. Though these quantities are defined for simpler second-order systems, they can still be used to help parameterize the converter for design. The parameterization was chosen to be defined as in 2.6 and 2.7. These definitions are substituted in to get the final parameterized form of the gain equation in 2.8.

$$Q = \frac{X_c}{R_{eq}} \tag{2.6}$$

$$\omega_0 = \frac{1}{\sqrt{C L_p}} \tag{2.7}$$

$$G = \frac{1}{\left(1 - \left(\frac{\omega_0}{\omega_s}\right)^2 + \frac{L_s}{L_p}\right) + jQ\left(\left(\frac{\omega_0}{\omega_s}\right)^2 \frac{L_s}{L_p} - 1\right)} \tag{2.8}$$

2.2.2 Zero Voltage Switching

One final concept, useful especially for resonant converters, is soft-switching. The opposite concept is hard-switching which is when a switch turns on or off before the voltage (V_{ds}) or current (i_d) has reached zero. This will result in an overlapping non-zero current and voltage, or equivalently a power loss called switching loss. This loss occurs for every turn-on or turn-off event increasing with the switching frequency. For this converter, the turn-on soft-switching event will be the focus.

Switching Losses are avoided by forcing V_{ds} to zero before the current starts to ramp up as a switch is turning on. If the voltage is already zero, then no power will be lost regardless of the current. This is known as Turn-On Zero Voltage Switching (ZVS). The question then becomes how to force V_{ds} to zero? To do so, C_{ds} has to be discharged by a current, bringing V_{ds} to zero. In relation to the resonant tank, a backwards current forced through the top switch, or a forward current forced through the bottom switch will create this discharge effect when the switch is off. These directions are required by the direction of the switch's body diode. Using the deadtime when both switches are off, the natural resonance of the tank will discharge C_{ds} and drop the V_{ds} of a switch to zero just before turn-on. Figure 2.6 demonstrates the principal for the top switch. If the series resonant current (I_{series}) is moving backwards when both switches are off, it can only conduct through the top switch due to the body diode. This takes place in two parts. First, as denoted by I_{d1} in the figure, the current flows through C_{ds} and discharges it. At this point the switch has as V_{ds} of zero and could turn on with ZVS. Once C_{ds} is discharged, the series resonant current can continue flowing through the body diode where I_{d2} maintains zero V_{ds} (for an ideal diode). The same idea happens for the bottom switch in Fig. 2.7, but now the series current is revered so the

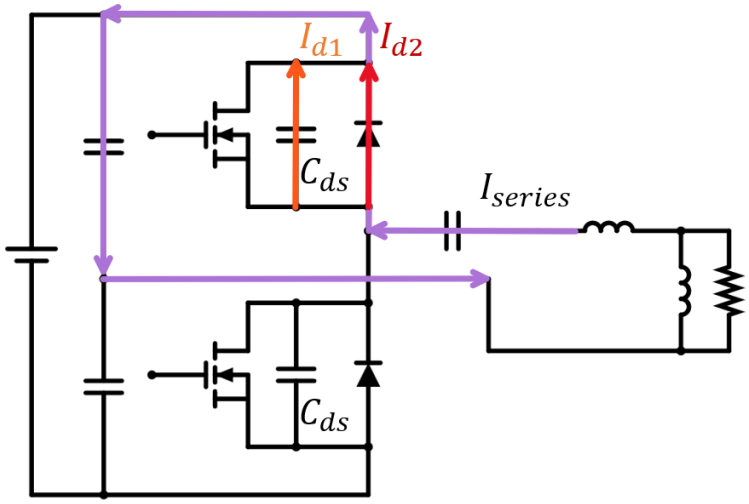

Figure 2.6: Turn-On Soft Switching for the Top Switch

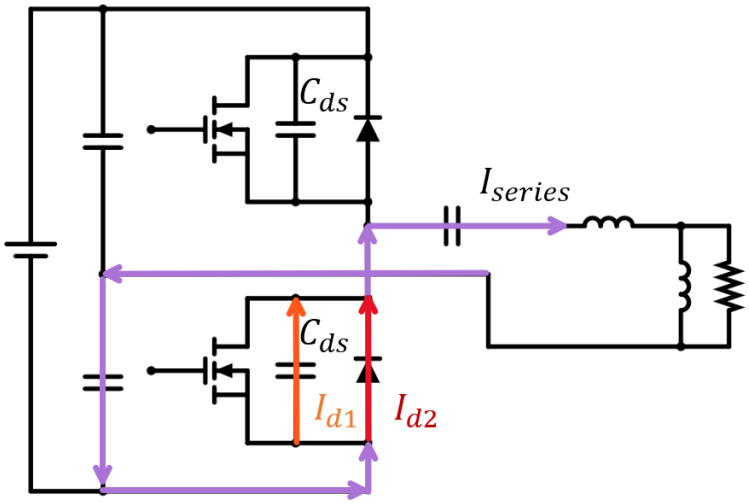

Figure 2.7: Turn-On Soft Switching for the Bottom Switch

bottom switch is the only path. This would lead to ZVS conditions for the bottom turn-on case. The challenge now is that the natural frequency of the tank may not align with the switching turn-on illustrated here. The solution is to control the switching frequency to be above the tank's natural resonance. This makes the circuit behavior more "inductive" and lags the current resulting in the described current directions.

2.3 Operating Point Selection and Design

Now that a complete understanding and model of the converter is formed, the component values can be chosen. The converter gain and resonant tank gain are chosen to be one, so the transformer must have a turns ratio of 1/2. On another note, to get more accurate simulations including switch dynamics, a spice model of a Qorvo 1200V SiC Cascode JFET was used [1]. From the datasheet[20], The switch has a max V_{ds} of 1200V, and a maximum DC I_d of around 114A for a case temperature of 125°C. To give some buffer, a voltage of 800V is chosen as the limit, along with a max DC current of 110A RMS or 156A Peak. This sets a theoretical max power of 88kW but due to the current limitations of the switch, 39kW will be the max power.

The parameterized gain equation (2.8) is used to design the converter. The gain equation's independent variable is the ratio of the natural-to-switching frequency $\left(\omega_{ratio} = \frac{\omega_0}{\omega_s}\right)$. Multiple curves of the gain equation will be plotted for different Q values showing the possible design space. The gain equation can be transformed into an equivalent form of (2.9) to align with the design choices described. To get the best solution, some tuning of the inductor ratio $\left(L_{ratio} = \frac{L_s}{L_p}\right)$ is necessary.

$$G = \frac{1}{(1 - {\omega_{ratio}}^2 + L_{ratio}) + jQ\left(\frac{L_{ratio}}{{\omega_{ratio}}^2} - 1\right)} \tag{2.9}$$

Additionally, the max drain current cannot exceed 156A Peak. This can be done by parameterizing the resonant current in terms of Q, ω_{ratio} and L_{ratio}. Using Ohm's law, the current is (2.10). Remember, the maximum voltage for the AC fundamental is half the input voltage and scaled by $\frac{4}{\pi}$.

$$I_{max} \geq \frac{V_{max}}{|Z_{tank}|} \tag{2.10}$$

[1] For an overview of the cascode structure see Appendix A

Where:

V_{max} is $\frac{V_{in}}{2} \cdot \frac{4}{\pi}$

$|Z_{tank}|$ is the magnitude of the equivalent series, resonant tank impedance at some frequency

I_{max} is the maximum current of the tank's series resonant current

Substituting in the parameters from (2.6, and 2.7) results in the final equation (2.11).

$$I_{max} \geq \frac{\frac{1}{R_{eq}} \left(\frac{\omega_{ratio}^2}{Q} + j \right)}{Q \left(1 - \frac{L_{ratio}}{\omega_{ratio}^2} \right) + j \left(1 - \omega_{ratio}^2 + L_{ratio} \right)} \tag{2.11}$$

Using equations (2.9 and 2.11) and substituting a chosen value for L_{ratio} of 0.1, the graphs in Fig. 2.8 and 2.9 are produced. The top graph contains the multiple gain curves for each Q over a range from 0.0 to 2.5. The curve is graphed over ω_{ratio} and the dotted line represents the required gain of one. The bottom graph represents the max drain current plotted in the same fashion. In this case, the dotted line represents the max drain current allowed by the switch model of 110A RMS or 156A Peak. There are many solutions to achieve the desired gain but most of these are invalid due to the max drain current limitations. This is why the theoretical maximum switch power of 88kW, and chosen power of 39kW, are so different. Within this solution space, a curve with a Q value of around 1.5 and up must be chosen to meet both gain and current specifications. As Q increases the curve becomes narrower allowing more "controllability". This means the gain is more sensitive to changes in frequency. This is generally a good thing as the converter doesn't have to dramatically change its switching frequency to get a desired gain. The tradeoff of a larger Q is less room to operate above the natural frequency to achieve ZVS. In these figures operating above the natural frequency means an $\omega_{ratio} < 1$. Zooming in on the curve with $Q = 2$ (Fig. 2.10, 2.11) shows this solution will meet the requirements. The final parameter values are: $Q = 2$,

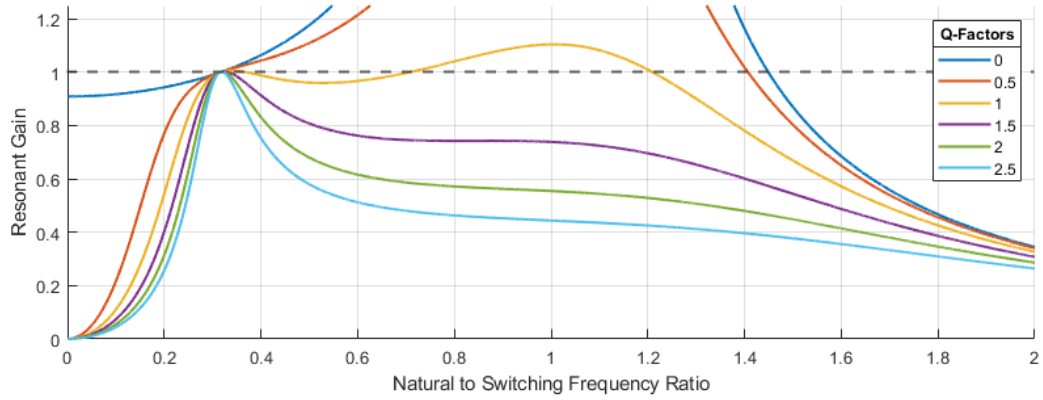

Figure 2.8: Parameterized Gain Design Curves

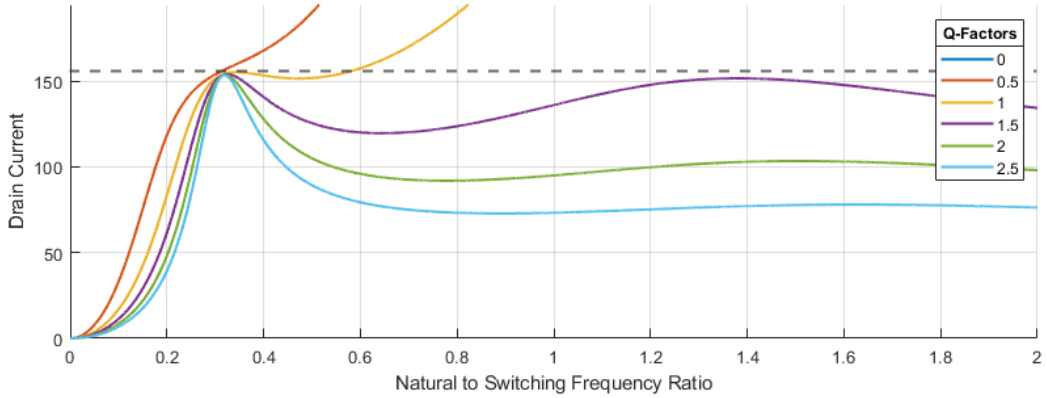

Figure 2.9: Parameterized Max Drain Current Design Curves

$L_{ratio} = 0.1$, and $\omega_{ratio} = 0.3163$.

Now that the design parameters are chosen, the real component values can be calculated. The load resistance is found using (2.3) along with the desired power and output voltage.

$$R_{eq} = \frac{8}{\pi^2} N^2 \frac{V_{out}^2}{P} = \frac{8}{\pi^2} \cdot 0.5^2 \cdot \frac{800V^2}{39kW} = 3.325\Omega$$

The passive reactances are found using (2.6, 2.7, and L_{ratio}):

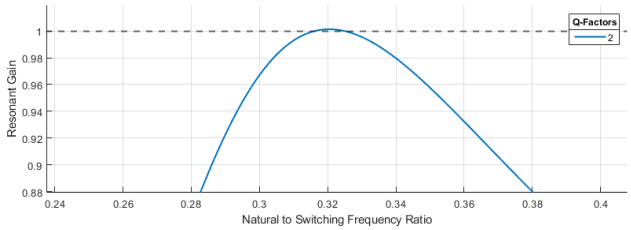

Figure 2.10: Chosen Gain Curve at $Q = 2$

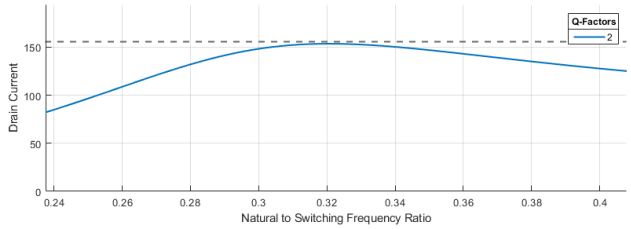

Figure 2.11: Chosen Max Drain Current Curve at $Q = 2$

$$X_c = QR_{eq} = 2 \cdot 3.325\Omega = 6.65\Omega$$

$$X_{L_p} = \frac{X_c}{\omega_{ratio}^2} = \frac{6.65}{0.3163^2} = 66.48\Omega$$

$$X_{L_s} = L_{ratio}L_p = 66.48 * 0.1 = 6.648\Omega$$

A switching frequency of 100kHz is chosen and the reactances are converted into component values. This yields final component values of $C_s = 239nF$, $L_s = 10.58\mu H$ and , $L_p = 105.8\mu H$. The full converter design and final components values are in Table 2.1.

2.4 Frequency Controller

As a final task, a frequency-controller is defined to find an operating frequency for any component value. Using the gain equation (2.8) and solving for ω_s, the high order polynomial of the form (2.12) is found. Since there is no simple general solution for this, a numerical solver is used to find a frequency for every simulated configuration. If no real solution is

Table 2.1: Converter Specifications

Parameter	Value	Unit
Voltage	800	V
Drain Current	110	A RMS
Converter Gain	1	V/V
Q	2	
L_{ratio}	0.1	
ω_{ratio}	0.3163	
Power	39	kW
Frequency	100	kHz
C_s	239	nF
L_p	105.8	μH
L_s	10.58	μH
R_{eq}	3.325	Ω

found, that simulation configuration is not used.

$$\omega_s^6 \left[\frac{L_s}{R_{eq}}\right]^2 + \omega_s^4 \left[\left(\frac{L_s}{L_p}\right)^2 + 2\frac{L_s}{L_p} + 1 - \frac{1}{G^2} - 2\frac{L_s}{C_s R_{eq}^2}\right] + \omega_s^2 \left[\frac{1}{C_s^2 R_{eq}^2} - 2\frac{L_s}{C_s L_p^2} - 2\frac{1}{C_s L_p}\right] + \left[\frac{1}{C_s L_p}\right]^2 = 0 \tag{2.12}$$

CHAPTER 3

Data Collection

3.1 Parameter Sweeps and Parameter Correlations

To properly train a neural network, a complete dataset containing different circuit configurations in the desired ranges must be constructed. The number of parameters desired to be predicted and the ranges these parameters may span leads to an explosion of required simulations to fully cover the simulation space. In order to reduce the number of simulations needed, parameters with weaker effects or significance can be eliminated. After this, the ranges for each parameter will be chosen.

3.1.1 Experimentation and Parameter Correlations

To truncate the number of unique simulations, the most vital parameters must first be determined. This was done by building a circuit model and manually running sweeps to test the sensitivity of parameter relationships. For example, all the resonant passives were swept in a range of $\pm 35\%$. Then the impact of passive component variations on a list of converter behavioral parameters were checked to see if there were any trends. Taking the change in C_s, the effect on the RMS/Max series resonant current is displayed in Fig. 3.1. This experiment produced a clear, reasonably measurable correlation that is almost linear. What this suggests is that predicting changes in C_s will require having data about the resonant current. That was a simple example but a more complex question may include how changes in the switch's internal JFET Threshold voltage affect the gate charge? This might be useful since a common degradation mechanism for SiC devices revolves around the gate oxide layer breakdown and changes in the threshold voltage [21, 22, 23, 24]. This thesis has a unique switch situation though, since a cascode structure is used such that the JFET's

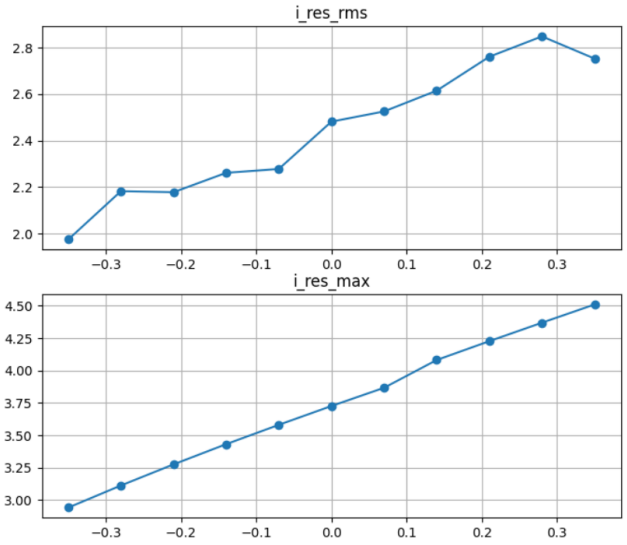

Figure 3.1: Resonant Current versus ±35% Sweep in C_s

gate is not being directly controlled. SiC devices tend to have their threshold voltages drift upward [21, 22], but for a Normally-On JFET, the threshold voltage is negative. Because of that, the sweep ranged from 0 to an additional −1V. The results are shown in Fig. 3.2 (Q_{gate} *results are all scaled by* 10^{-7}), where the top and bottom correspond to the top and bottom maximum gate charge during steady state. The gate charge only spans about 10nC so the effect would already be quite small. It seems no simple pattern occurs, though compression in the simulation data could lead to some noise in these results. In summary, this type of result would suggest the relationship is not strong and might be worth leaving out of the simulation pool. Looking at more experiments, it was found that the JFET Threshold Voltage was not a major factor due to its small expected deviation and the fact that the JFET is never directly controlled in the cascode structure. However, it was found that directly modifying the JFET Gate-Source capacitor leads to changes in the gate charge of 200nC, which would be more significant. This measurement was not included though as the switch should be observed at the level of its package instead of internal details. Similar experiments were repeated for L_s, L_p, R_{ds-on}, JFET C_{ds}, JFET C_{gs}, MOSFET C_{ds}, MOSFET C_{gs}, JFET V_{th}, and MOSFET V_{th}. Note that any parameters belonging to a switch must be ran such that the top and bottom sweeps occur alone, as well as at the same time.

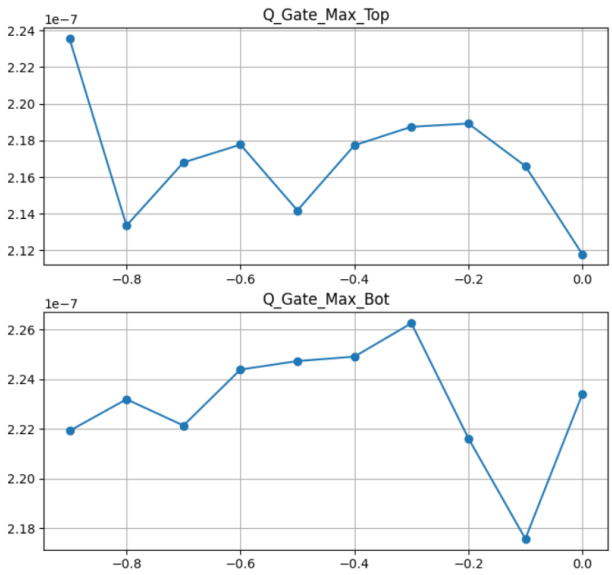

Figure 3.2: Top and Bottom Gate Charge vs Additional -1V Sweep in JFET Threshold Voltage

3.1.2 Sweep Ranges Determination

Using the experimental results and converter operating point, a series of parameters to sweep and their ranges were chosen. Table 3.1 shows the complete set of parameters and their ranges. The reasoning for these ranges is described below.

1. **Input Voltage** : This was chosen to simply sweep around the designed voltage

2. **Power**

   A converter may see are large range of power draw depending on the load. The max power is the 39kW already chosen, and the lower limit is 1kW. This is because a no load condition (0kW) corresponds to a Q of zero, which for this design has no solution. However, a power of 1kW was able to properly simulate and represent low power conditions.

3. **Passive Components**

   The passives were picked to span an area larger than just the component tolerances to

Table 3.1: Simulation Parameter Ranges

Parameter	Base Value	Range
C_s	239 nF	±50%
L_s	10.58 μH	±50%
L_p	105.8 μH	±50%
V_{in}	800 V	±25%
C_s ESR	20 mΩ	±25%
C_{DC} ESR	10 mΩ	±20%
Parameter	**Min**	**Max**
Power	1 kW	39 kW
L_{loop}	0.5 nH	20 nH
L_s ESR	10 mΩ	75 mΩ
L_p ESR	10 mΩ	75 mΩ
R_{ds-on}	0 mΩ	50 mΩ
C_{ds}	0 pF	105 pF
C_{gs}	0 pF	2550 pF
T_c	-25 °C	200 °C

also be useful for condition monitoring.

4. **Passive Series Resistance**

   To fully model the converter an Equivalent Series Resistance (ESR) for all passives was added. The capacitors' ESR were based on a plausible component and then operated in a range around that. In the case of C_s, a 220nF 1kV Film Capacitor by KEMET was measured using an LCR meter and found an ESR of about 20mΩ. The same processes was used for the DC-Link capacitors who had a real counterpart ESR around 10mΩ. The inductor ESR's were chosen to have a broader range that is associated with different implicit ESR's and an increase in resistance due to heating the copper.

5. **Additional On-State Resistance**

   The R_{ds-on} was chosen to have a range that only added additional resistance. Additional resistance here can represent junction temperature increases as well as device degradation. The maximum of 50mΩ was chosen as it exceeds the range of variation given by [20, Fig. 5].

6. **Switch Capacitance's**

   Switch capacitances include C_{gs}, C_{ds}, and C_{gd}. They are arranged around the switch as in Fig. 3.3. Similar to R_{ds-on}, only additional capacitances were added on. In this case 30% of the typical datasheet value ([20]) was taken to be the max additional.

7. **Loop Inductances**

   Loop inductances are small inductors added in series ahead of the top switch's drain, and after the bottom switch's source. Fig. 3.4 includes them on the top and bottom rails connected to the source. These inductors help to simulate inductance from board traces that can add non-ideal voltage spiking to V_{ds}. These inductances were kept very small only ranging from 0.5 to 20nH.

8. **Case Temperatures**

   Because thermal simulations are so slow a thermal approximation model was used. In this case the switch junction temperature, T_j, was fixed at an average value to compromise for all possible converter configurations. An estimated switch loss was used as the input to the switch's thermal model and the case temperature, T_c, was swept instead. This allows the computing of approximate final T_j's for different T_c's. T_c was allowed to sweep a wide range as the simulations took almost no time.

3.2 Simulation Model

3.2.1 Spice Model

The circuit was implemented in LTSpice which can be seen in Fig. 3.4. The circuit is as expected but there are a few details to be aware of. The ESR resistances are implicit in the passives, but they are being modified as parameters. The two loop inductances can be seen on the top and bottom wires connected to the source. A large resistor, not part of the model, was placed parallel to L_p just to help the simulation's numerical solver. The switches

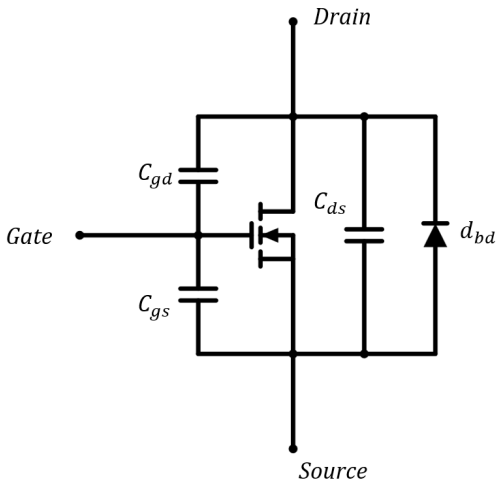

Figure 3.3: Switch model including Capacitance and Body Diode

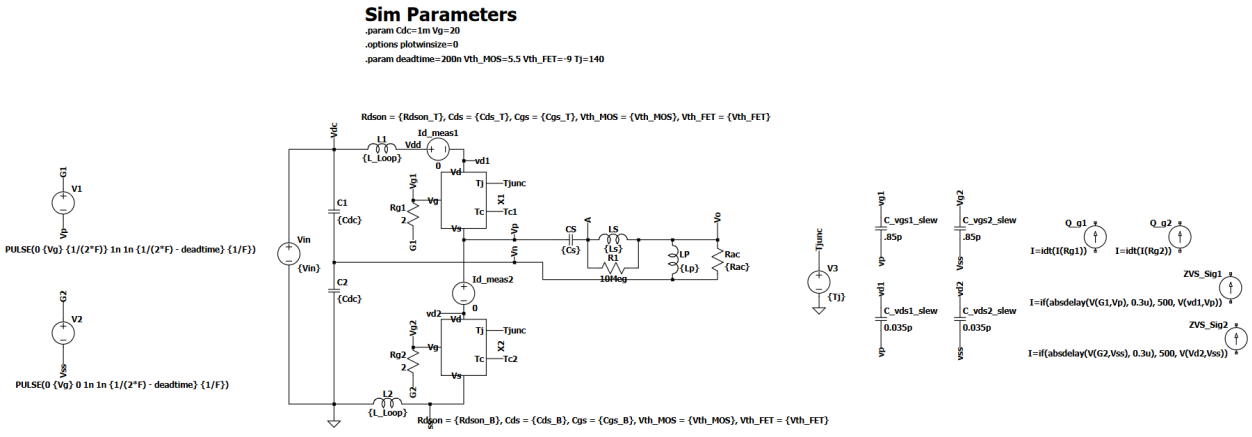

Figure 3.4: LTSpice Circuit Model

are sub-circuits that have a long list of parameters to add all the switch parameter sweeps discussed. On the left side, it can be seen that the two gate drivers are just dependent voltage sources configured to produce square waves. On the right, a single voltage source holds T_j constant which is necessary for the thermal approximation. Also on the left, four capacitors exist to find the slew rates of V_{ds} and V_{gs}. The only parameters kept constant throughout all the simulations are the DC-Link capacitors, switch gate control voltage, switch gate resistors, threshold voltages of the internal FET and MOSFET, dead-time and the switch junction temperature. Finally, data compression is turned off with a directive. This is so that the results do not have noise or have cut off peaks due to lossy compression.

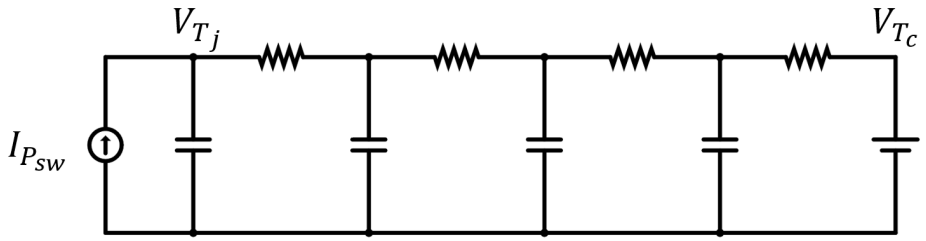

Figure 3.5: RC Network Thermal Model

3.2.2 Thermal Approximation Model

It will be useful to collect thermal data related to the switches. This has consequences for the health of the switches and opportunities for preventative control. The switches themselves do have thermal models but to see any significant thermal dynamics would take far too many switching cycles to practically simulate. Instead, to rapidly collect data for the numerous simulations needed, a simple approximate model is used. The main switch has an existing RC thermal model named an Extracted Cauer Network [25]. The models look like Fig. 3.5 where the switch power is injected as a current source going into the network. At the other end, a voltage source is used to represent the case temperature. For a given switch loss and case temperature, the network will reach a steady state where the final junction temperature can be measured as left most capacitor voltage. In practice, the average switch losses from the main simulations are measured and used as the current input to the thermal circuit. The approximation comes from having to fix the junction temperature at a steady state that is not true for every simulation. This approach leaves out self-heating effects that would practically increase the losses and therefore junction temperature, but it allows rapid estimations. To try and even out the effects of self-heating and the multitude of circuit configurations that might change the temperature, the main simulation will force the junction temperatures on each switch to be 140°C. This will overestimate the switch losses for some circuit configurations and underestimate the switch losses for other circuit configurations. As a final note, the thermal approximations are ran for both switches and each case temperature is swept according to the range in Table 3.1.

3.2.3 Initial Conditions

To further reduce simulation time, the simulation is not ran to steady state but instead preset to a calculated AC-approximate steady state condition. These calculated initial conditions are not perfect, so the converter is given another cycle to adjust and then three more cycles used for data measurements. The initial conditions that are set are the voltages and currents in the resonant tank. They are found using the fundamental AC analysis with the following terms:

$$Z_1 = X_c + X_{Ls} \tag{3.1}$$

$$Z_2 = X_{Lp} \parallel R_{eq} \tag{3.2}$$

Using those impedance terms the equations for the initial conditions are defined. The first equation (3.3) uses Ohm's law with V_{max}, the AC fundamental approximation of the half-bridge square wave. The next two equations (3.4, 3.5) are again Ohm's law applied to the reactance of that respective component. Next, the output AC voltage is found using the previously defined terms or alternatively, the gain equation (2.8). Finally, using the output voltage, and again, Ohm's law, the magnetizing inductance current is calculated (3.7). These values will be set at before the simulation begins.

$$I_s = \frac{V_{max}}{Z_1 + Z_2}, \quad \text{where } V_{max} = \frac{V_{in}}{2} \cdot \frac{4}{\pi} \tag{3.3}$$

$$V_{Cs} = I_s X_c \tag{3.4}$$

$$V_{Ls} = I_s X_{L_s} \tag{3.5}$$

$$V_{out} = V_{max} \frac{Z_2}{Z_1 + Z_2} = V_{max} \cdot G \tag{3.6}$$

$$I_p = \frac{V_{out}}{X_{L_p}} \tag{3.7}$$

3.3 Simulation Procedure

To gather all this data, a custom script is used to manage all of the configuration sweeps and only uses LTSpice for one simulation at a time. This is because LTSpice has limited customization and if a simulation fails or stalls during a sweep, this could lead to the loss of simulation data and time. Managing all the simulations in this custom manner also makes it easy to back files up for later use and reference. This section will go through how this is done. For a flowchart of this procedure see Fig. 3.6.

To begin, a random operating point is chosen from the ranges in Table 3.1. This is done by subdividing the ranges by 25 and choosing a random point in the range for all parameterized values. To help keep track of the multitude of simulation files, a hash function is used on the configuration text string as a name for all related simulation files. Once a configuration is picked the operating frequency is calculated using equation (2.12). Finally, using the configuration and frequency equations the initial conditions are found.

At this point the simulation netlist file is created through LTSpice's command line commands. Subsequently, the netlist is modified to include the desired configuration and initial conditions. This involves writing Spice commands such as ".param" or ".ic", to the netlist file. All the measurement statements are also written as configured in the script by a list of variable names, ranges, and style of measurement. These measurements are limited to max, min, RMS, or average over the three cycles used. In general, measuring is cheap to do during simulation. After a simulation, it requires manual scripting to automate

measurements, so getting the measurements done right is helpful. Despite trying to avoid this, some measurements are done after the simulations by directly using the data points from the raw files along with custom scripts. This was done for V_{ds-on}, which consists of finding the on-time period and taking the max, min, average, and RMS of the voltages to stay consistent with LTSpice. V_{ds-on} is useful for R_{ds-on} measurements as examined later. Another set of post-processing measurements done were for drain and gate slew rates (both current and voltages). The slew rates from LTSpice were found to be susceptible to large spikes due to their differentiation method. So a custom differentiation method over a larger window of several samples is used to avoid these spikes. Using the resulting derivative waveform the max, min, RMS, and average measurements are taken again.

Sometimes when running a simulation it could crash and halt. In that case, the netlist and log files are still saved, but marked as failed in their file's name. This will be used to avoid repeat simulations and also help debug failed simulations. Other times, simulations can run for much too long. A timeout feature was added to fail a simulation if it went on too long. A timeout of 10 minutes was used for the main simulation, while a timeout of 2.5 minutes was used for the faster thermal simulations. Alongside all of these checks, log messages are being written to a custom log file that could be referenced later. These messages contain information on what simulation is currently running, if it timed out, if it succeeded, if a valid operating frequency could be found, or the total time to run a simulation batch.

After the main simulation is done, two more thermal approximation simulations are ran for each switch. The methods used are described in Section 3.2.2 resulting in the approximate steady state final junction temperatures. Before running those simulations, the average switch power has to be read out from the main simulation log file. Even though the thermal simulations are much less complicated or prone to failing, they are also checked for success and could timeout if stalled. That is the last step and concludes the general flow of running a single simulation configuration. The process of simulating is repeated as much as needed

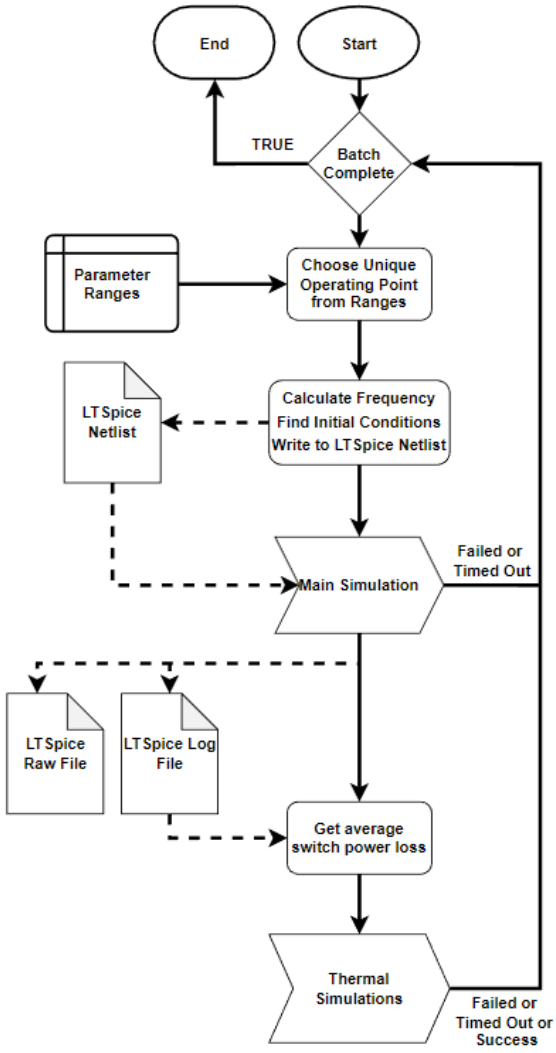

Figure 3.6: Simulation Procedure Flowchart

and then the data must be gathered and post-processed.

3.4 Data Handling

Once many configurations are simulated, there will be a mix of log files, raw files, and thermal log files. Fetching that data involves manually going in and reading the measurement statements. In the next sections, the processes of going from simulation files to a complete dataset is outlined. For an overview, see the flowchart in Fig. 3.9.

The first step is to get the list of Simulation ID's (SID) and check to make sure each

file with that SID has two thermal log files, and isn't marked as failed. This guarantees the simulation ran to completion. In general, the failure rates for simulations were always below 5%. Next, each log file and netlist is scraped for the relevant data by looking for spice commands such as ".param", "Measurement:", or ".step". One thing to keep in mind is that for each SID, there are a range of case temperatures simulated. This effectively repeats the simulation configuration for every case temperature and paired final junction temperature. This increases the number of effective simulations by 10 (due to the range size of T_c). All of this data is packed into a dictionary structure as discussed in the Data Formatting section below (3.4.2). For each parameter, the data will be normalized between zero and one to help the neural network keep the gradients at the same scale so no one parameter is preferred because it is larger.

3.4.1 Post Processing

Unfortunately, collecting measurements in LTSpice after a simulation has ran must be done manually. The solution to this is to directly read the ".raw" files, gather the data and time points, and do custom operations on those. For this project, both V_{ds-on} and a series of slew rates are measured using this process. The raw files themselves contain a header with information on the simulation, and then all the data stored in raw binary. This can be decoded manually into floating point numbers though an existing Python LTSpice library is used to do just that [26].

The first calculations done were the switch slews rates including V_{gs}, I_{gs}, V_{ds}, and I_{ds}. It was found that LTSpice's derivatives resulted in waveforms with large erroneous spikes that would pollute the data. To fix this, a custom derivative over a larger window of a few samples (3.8) is used. This is done by indexing the data and time arrays by the window size, then dividing the data difference by time differences. Each adjacent index in the time array doesn't correspond to the same time step so it's important to get the time difference directly. Doing this over the waveform results in the complete slew rate waveform. Then,

like before, the min, max, average, and RMS are taken. The min and max are easy to get by simply searching the slew data list. To get the average and RMS, trapezoidal integration is used (3.9) for consistency with LTSpice. Figure 3.7 shows the graphical representation of integration where one discrete area is equal to a rectangle and triangle defined by the two samples. The only other thing calculated after simulating was V_{ds-on}. In Fig. 3.8, both the gate control signal (top) and V_{ds-on} waveform (bottom) are displayed. The gate control edges can be calculated from the switching frequency and a small buffer is added on to mask out the ringing, in this case 5% of the switching period. The result is a window containing only V_{ds-on}. The data within this mask can be used to get the max, min, average and RMS like before.

$$\frac{\Delta X}{\Delta T} = \frac{X[n+W] - X[n-W]}{T[n+W] - T[n-W]} \tag{3.8}$$

Where:

X is the waveform array

T is the time array

n is the current index

W is the window size

$$Y = \sum_{n=t_0}^{t_1} \left(X_{min}[n] + \frac{1}{2}\Delta X[n] \right) \Delta T[n] \tag{3.9}$$

Where:

ΔX is the absolute difference in two adjacent waveform samples

ΔT is the difference in two adjacent time samples

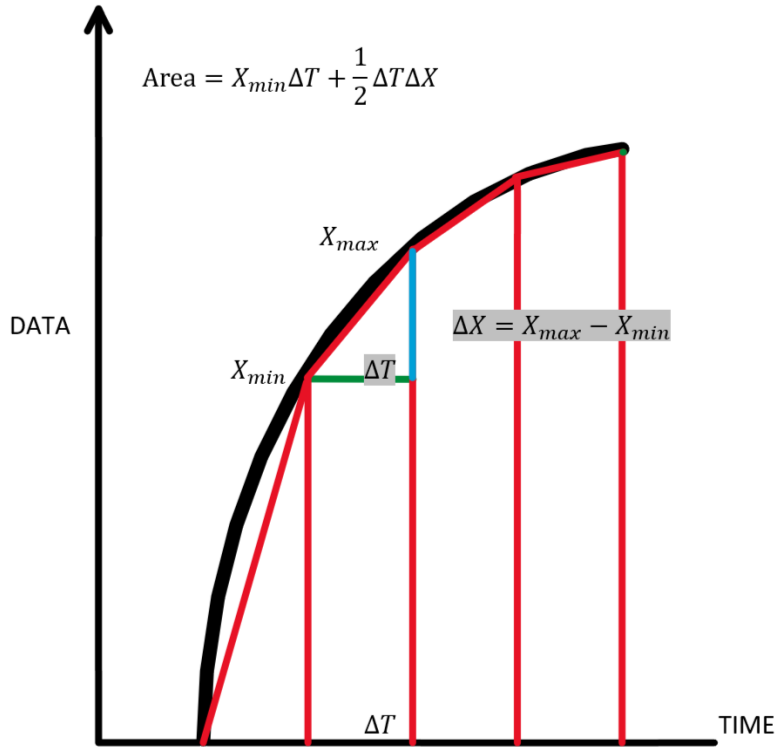

Figure 3.7: Trapezoidal Integration Diagram

X_{min} is the smaller waveform sample for two adjacent waveform samples

3.4.2 Data Formatting

To best interface with the Pytorch (a framework used for machine learning) [27], a custom dataset object is created. There is some freedom in how to store the data so a dictionary data structure was chosen. A dictionary is simply a list of key/value pairs. This is perfect for the data because each measurement or parameter can be referenced by its name as a key, and the values can be stored in a long list (See table 3.2). In order to differentiate the values in the lists, the SID is stored which has the form of a number. For each SID, all the measurements and parameters will be stored in the lists, aligned to the SID index.

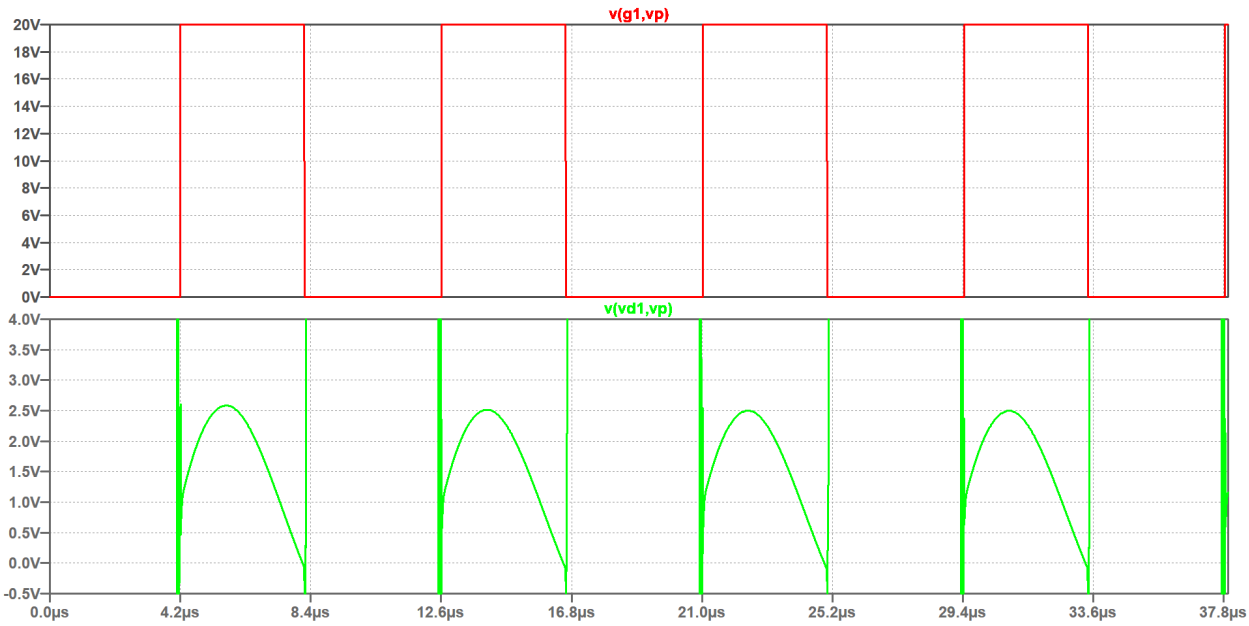

Figure 3.8: V_{ds-on} **(Bottom) and Gate Control (Top) Waveforms Demonstrating** V_{ds-on} **Measurements**

By doing this in order, every index of a list will be the data for one simulation. Table 3.2 has red values in all the lists for each key. Those red values correspond to one simulation configuration as an informal column through each key's list. Something to keep in mind is that accessing data from a dictionary will not be fast because a part of each list must be loaded into memory just to retrieve a single value. For large datasets, that results in a lot of wasted loading during training. Therefore, a pre-packing method is used to help speed up training. All this does it pre-fetch the input/output pairs and store them in an array. This way, only one list needs to be searched to get all the relevant data for that simulation.

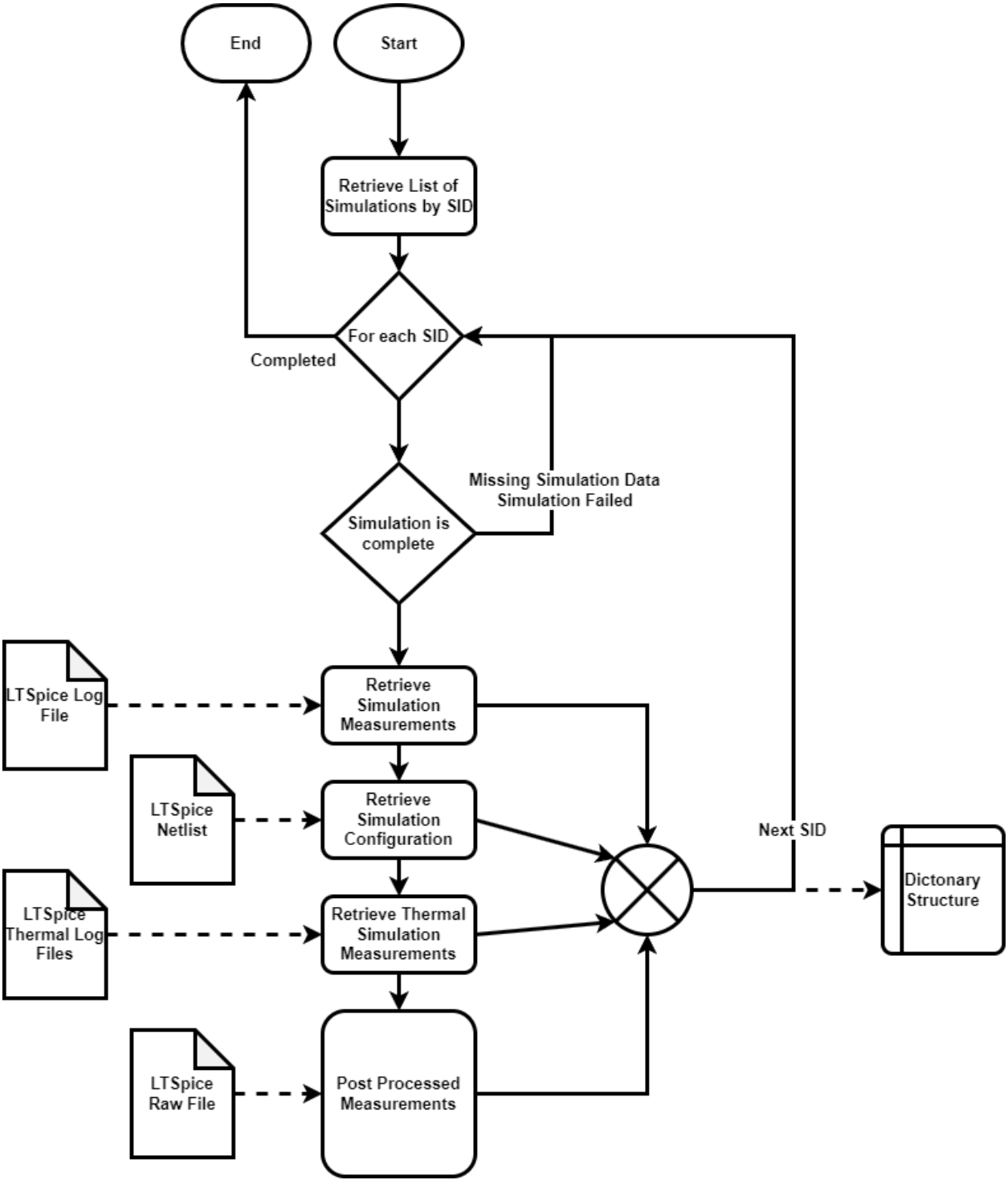

Figure 3.9: Data Collection, Processing, and Storage Flowchart

Table 3.2: Dictionary Data Structure

Keys	Measurement Lists
SID	[0, 1, 2, 3, 4, ...]
C_s	[0.23, 0.65, 0.82, 0.12, 0.05, ...]
L_s	[0.97, .73, 0.12, 0.60, 0.41, ...]
⋮	⋮
C_{ds-t}	[0.56, 0.63, 0.79, 0.51, 0.40, ...]
C_{ds-b}	[0.94, 0.97, 0.84, 0.42, 0.83, ...]
⋮	⋮
$I_{series-max}$	[0.53, 0.00, 0.72, 0.65, 0.10, ...]
$I_{series-min}$	[0.69, 0.08, 0.59, 0.92, 0.50, ...]
⋮	⋮
$v_{ds-on-avg}$	[0.83, 0.84, 0.00, 0.33, 0.17, ...]
$q_{gate-rms}$	[0.67, 0.75, 0.13, 0.31, 0.42, ...]

CHAPTER 4

Machine Learning Architecture

4.1 Artificial Neural Networks

4.1.1 Feed Forward Neural Networks

Artificial Neural Networks (ANN) propagate inputs through a series of interconnecting functions. There are many different kinds of architectures for different uses, but the simplest one is called a Feed Forward Neural Network (FFNN). In a FFNN, there are a series of layers that contain "Perceptrons", which act like mathematical neurons to gather inputs and produce a new output. Each layer has some number of these Perceptrons such that the inputs to each perceptron include all the outputs from the previous layer, making it fully connected. A generic structure for a FFNN is shown in Fig. 4.1. It has the inputs, represented by boxes, feeding into every perceptron in the first "input layer". The information is propagated through in a fully connected fashion until the output layer is reached. This example network only has an input and output layer, which are both necessary interfaces. Larger networks will contain "hidden layers" in between, adding more depth for more sophisticated calculations. Next, the inner-workings of the perceptron are explained. The breakdown of a perceptron is displayed in Fig. 4.2. The perceptron takes all of its inputs as a vector and performs the dot product with its internal weight vector. After this, a bias parameter is added to the scalar output from the dot product. Both the weight vector and the bias are both trainable parameters to adjust for a better fit to the data. The final step is to push the previous result through some non-linear activation function. In summary, a perceptron can be written in the form of (4.1) where $\vec{W}$ is the weight vector, $\vec{X}$ is the input vector, B is the bias value, and f is the non-linear activation function.

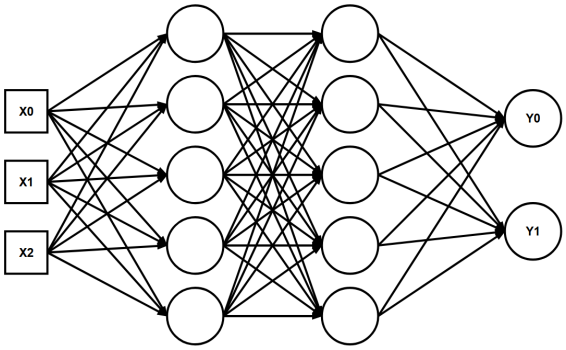

Figure 4.1: Feed Forward Artificial Neural Network Architecture

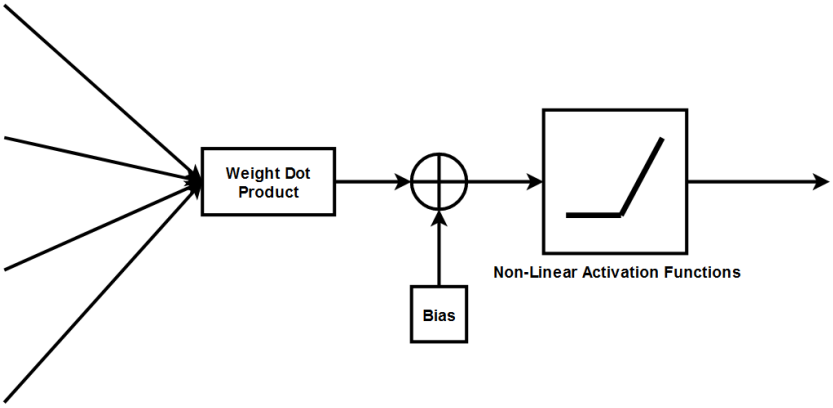

Figure 4.2: Perceptron Block Diagram

$$y = f\left((\vec{W} \cdot \vec{X}) + B\right) \tag{4.1}$$

4.1.2 Activation Functions

Each perceptron has a non-linear activation function on its output which allows the FFNN to fit complex non-linear curves to the data. There are a few of choices for this activation function to discuss. In general, this is considered a hyperparameter that must be tuned by the designer. A selection of activation functions are shown in Figures 4.3, 4.4, 4.5, and 4.6. The first activation function in Fig. 4.3 is called a Sigmoid curve. Notice that it smoothly transitions between zero and one, but saturates at those values. The next curve is the Hyperbolic Tangent or Tanh (Fig. 4.4). This curve also has an "S" like transition

between two asymptotes except tanh has a range of negative one to one. Both of these functions saturate which is good and bad. Saturation is useful because very large outputs cannot occur which overpower the network [28]. On the other hand, the derivatives of these functions will be very small near the asymptotes. The processes of training a network with Gradient Descent requires calculating the gradients back through the network in a large chain. If these activation functions can produce small gradients, they will multiply to even smaller gradients, shrinking the magnitude of changes that can be made to a trainable parameter, resulting in slower training. This problem is named the Vanishing Gradients problem[28].

Beyond the smooth S-shaped functions, there is an entirely different kind of activation function called the Rectified Linear Unit or ReLU for short[28]. Seen in Fig. 4.5, this is a piece-wise function which only passes the output when it is positive, hence the rectification. The range is from zero to infinity, meaning unbounded for positive inputs. This function is quite fast given its simplicity and linearity, so fast than it can perform six times as fast as other activation functions [28]. There are a few downsides to its nature, for example when fitting smooth curves, a piece-wise linear function is not as easily adaptable as the previous smoother functions. Another problem is the possibility for neurons with ReLU to stop learning referred to as "Dying ReLU" [28]. If a neuron with ReLU happens to learn a large negative bias or negative weights such that the input to ReLU is a negative number, ReLU will output zero. During the gradient chain calculation, the derivative of ReLU for negative inputs is zero resulting in the neuron having no change in its weights. This will halt the learning process and effectively remove that neuron from the network. To fix the possibility of Dying ReLU, a function called Leaky-ReLU was introduced (Fig. 4.6). This function has a slight downward slope for negative inputs so the neuron doesn't have that zero derivative and weights will always be updated. The slope of the negative line introduces another hyperparameter choice.

In general, something like the vanishing gradient problem is worse for networks with

more layers, so this may not be a problem here. It is not recommended to use the Sigmoid due to speed, though it is good for shallow networks or classification [28]. Tanh is preferred over Sigmoid because the gradients on this function tend to be higher reducing, the vanishing gradient problem. Sigmoid also suffers from zig-zag weight updates associated with sigmoid only having positive outputs and gradients[28]. This is slows training down as the weights cannot traverse the solution space freely in the direction of optimal values. In general the ReLU series of functions are generally recommended to start with, and are faster to compute [28]. With these recommendations, the ReLU and Leaky ReLU will be tried first, but tanh can be experimented with for its smoothness which may aid in fitting the non-linearities associated with switch parameters especially.

4.1.3 Training FFNN

Training a neural network involves a few steps. Assuming the network is initialized with random weights, it can calculate outputs if fed inputs. Of course, the FFNN will likely make a bad prediction. To be more exact on what a "bad" prediction really is, a Loss Function is used to. The loss function finds a measure of difference between predicted outputs and the known ground-truth outputs. In the case of regressions of continuous parameters, a loss function like the Mean-Squared-Error or MSE (4.2) is useful. The MSE takes the difference between the true outputs (y_i) and the predictions ($\hat{y}_i$), and accumulates their average squares. For binary classification, the Binary Cross Entropy Loss (BCE) 4.3 is a typical choice. Since there is only have one class such that the output represents the probability of ZVS, the BCE loss only has one log term. Once a loss has been computed, the trainable parameters are modified slightly in proportion to their sensitivity to the loss function. If a parameter is important in determining a bad prediction, it will be modified more for that training example. Likewise, parameters which have little effect for a training example won't be changed by much. This method of training is called Gradient Descent. The method follows the gradient in the solution space as it tries to find minima of the loss function through the

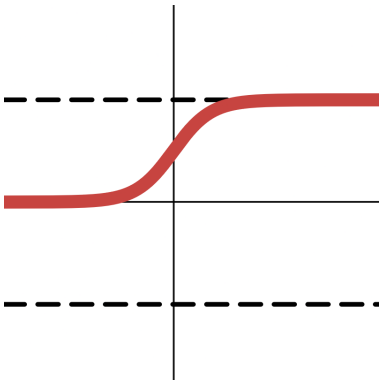

Figure 4.3: Sigmoid Activation Function

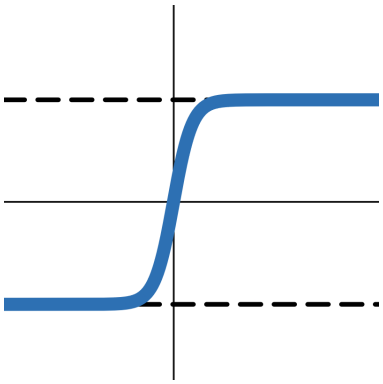

Figure 4.4: Hyperbolic Tangent Activation Function

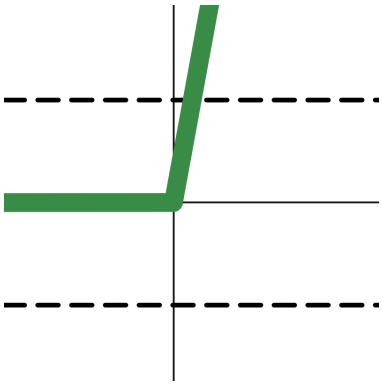

Figure 4.5: ReLU Activation Function

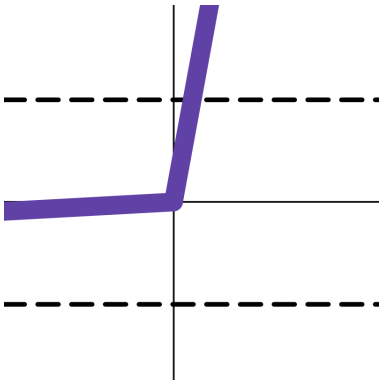

Figure 4.6: Leaky ReLU Activation Function

weights of the network.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4.2}$$

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}\log(\hat{y}_i) \tag{4.3}$$

The process of quantifying how much a parameter effects the predictions, and therefore the loss function, is done by finding the gradient of the loss with respect to any given weight

or bias. The formula for modifying one weight is written in (4.4). The α term is called the learning rate which modifies the "nudge" amount. The learning rate adds to the list of hyperparameters needed to be chosen. Since the gradient is difficult to compute symbolically, libraries like Pytorch make use of an Auto-Gradient system to do this. The process of using the gradients to actually modify the weights and biases is called back propagation. In Pytorch, this is done by an optimizer of which the Adam Optimizer is a popular choice [29]. This optimizer adds two main features that have proven useful. Adam expands regular gradient descent by adding a feature called "momentum", giving the nudges a sort of inertia. It does this by adding the previous nudge, times a constant (β) to the current nudge (4.5). Adam combines this idea of momentum with a learning rate scheduling algorithm that takes into account the previous gradients and current gradient to dynamically change the learning rate. If the learning rate is too large, minima will be skipped over by taking large strides. If the learning rate is too small, the network will take longer to learn and can get stuck in local minima by taking small strides. These methods help to take a more direct path to an optimal solution and avoid getting stuck in local minima.

$$w_{t+1} = w_t - \alpha \frac{\partial}{\partial w_t} L(y_t, \hat{y}_t) \tag{4.4}$$

$$w_{t+1} = w_t - \alpha \frac{\partial}{\partial w_t} L(y_t, \hat{y}_t) + \beta(w_t - w_{t-1}) \tag{4.5}$$

Where:

ω_t is a trainable weight at some time t

α is the learning rate

$L(y_t, \hat{y}_t)$ is the loss function evaluated at some time t using the ground truth and predicted outputs

β is the momentum constant

4.2 Testing Scores

Once a model is trained it should be verified on data it has not trained on. This will confirm if the model has overfit and determine how it performed. In this sense, testing is similar to the loss function just without any gradient descent steps. The testing phase uses a few different scores to help quantify performance. For regression tasks, the first score is the Mean Absolute Error (MAE) (4.6) which finds the average absolute difference between the predicted and true values. This gives a rough idea of model accuracy. The next score is the Root Mean Squared Error (RMSE) (4.7) which is just the root of the MSE (4.2) such that it ends up on the same scale as the MAE for easy comparison. Unlike the MEA, the RMSE will weight larger errors much more due to the squaring. If a model is mostly accurate but has a few really bad predictions, the RMSE will be relatively larger than the MSE indicating instances of bad predictions. Finally, something called the Coefficient of determination or R-Squared (R^2) is used. Imagine a graph where one axis represents the true output magnitude, and the other axis for the predicted output magnitude. Plotting the test data would results in a scatter plot where each ground-truth would have a predicted value point in this space. The ideal case would be that the predicted and true output are equal which would create a $y = x$ line of points. If the prediction was constant, for example, a horizontal line equal to the predictions would form spanning the different ground-truths. Using this setup, the R^2 quantifies the "goodness of fit" to the ground truth outputs. A better score (closer to 1) occurs when the scatter plot data fits closer to the $y = x$ case. The equation for R^2 is (4.8) where $\bar{y}$ is the mean of the true outputs.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{4.6}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{4.7}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N} (y_i - \bar{y})^2} \tag{4.8}$$

Scoring is done differently for classification tasks. The simplest score is Accuracy or the ratio of correct predictions to all predictions. There are other scores to help differentiate model behavior and their importance is related to the goals of the model. Precision is a score defined by the ratio of true positives to all predicted positives (true and false positives). This score gives insight to how often a model thinks something is positive when it isn't. Another score is named Recall defined by the ratio of true positives to all positives (true positives and false negatives). Recall differentiates how well a network will miss a positive for a false negative. In the case of a DT, the classifier will be used to predict ZVS occurrences in both switches. In this context Precision scores how well the DT can avoid false positives or false predictions of ZVS occurring when it isn't. If the DT predicts ZVS is occurring but it is not, the missed absence of ZVS will lead to higher switching losses, resulting in greater thermal stress, leading to changes in switch characteristics and degradation. For health monitoring or lifetime extension, false positives for ZVS should be avoided. In the case of Recall, this scores the DT's ability to avoid false negatives, or thinking ZVS is lost when it is still working. If your control algorithms are using this to extend the lifetime by reducing the converter's power or switching frequency in absence of ZVS, this could lead to a loss in performance. That being said, the upside is that the converter will not be subjected to increased thermal stress and degradation if controlled conservatively. Depending on the application, either Precision or Recall could be more useful. For now, they help benchmark the DT agnostic of specific application scenarios.

4.3 Hyperparameter Tuning

Most of ANN's parameters are trainable but a few must be chosen by the designer and cannot be changed by gradient descent. These are referred to as hyperparameters.

Examples of hyperparameters include the number of hidden layers, the sizes of those layers, the activation function, the batching of data, the loss function, the optimization algorithm used, and the number of rounds trained on the dataset (named an "epoch"). There is no generalized method for optimizing these so manual "tuning" through trial and error is required. It is the task of searching over numerous combinations of hyperparameters that can make training performant networks a lengthy process. To simplify, the loss function (MSE or BCE) and optimizer (Adam) will be kept constant. Instead, tuning will focus on trying different network sizes, number of epochs, and some experimentation with the activation functions. Part of the training will also involve testing different inputs, but this is different than hyperparameter tuning which is used to optimize any given network's performance.

CHAPTER 5

Digital Twin Design

This chapter uses the knowledge gained from the converter modeling, parameter correlation experiments, determined sweep ranges, and FFNNs to determine the starting point input and output selections for the Digital Twin. A review of sensing limitations is done to avoid the more difficult measurements and then the input and outputs are selected.

5.1 Sensing Limitations

The ideal DT is capable of emulating the converter completely with only the known inputs. In this sense a DT eliminates the traditional burden of trying to sense many indirect internal parameters and using that information to characterize the actual parameters. To achieve this ideal DT is difficult and internal measurements still must be made. The desired result being a minimal set of simple measurements that is simpler than traditional sensing techniques. In this section a discussion of what measurements are most feasible is conducted and will aid the processes of choosing inputs for the DT later on.

Starting at the resonant tank, there are different voltages and currents that may be useful. From the experiments done, any prediction of a passive component in the resonant tank strongly correlates to the series resonant current, and voltages across the components. A problem that presents itself immediately is the accessibility of the resonant inductors. The transformer inside the resonant converter is often designed to implicitly contain these inductances as the magnetizing inductance (L_p) and leakage inductance (L_s), therefore a voltage cannot directly be measured on these devices. Furthermore, all the passive components have large high frequency signals requiring galvanic-isolation. Exposing sensing hardware to electromagnetic interference (EMI) could distort the measurements and excite

parasitics causing ringing. These challenges make these measurements undesirable and are better avoided. However, to get information on the resonant tank, the series resonant current is measured. If a shunt resistor is used, even at the 110A RMS, the voltage across is small. The downside here is the extra resistance changes the resonant tank and adds to conduction losses. Because there is no DC component in the series current, isolated magnetic sensing methods can also be used [30, 31]. Measurements that are normally already taken for converters are the input and output currents and voltages. These are useful to define the input and outputs of the resonant tank, and general information about power losses.

Moving on, the switch measurements presents challenges due to large slew rates in current and voltage, high frequencies and EMI. Fitting a small sensor near the switch to accurately measure large, fast changing parameters requires more specialized sampling and processing hardware. In some cases, these measurements are unavoidable. Properly estimating the switch capacitances requires knowledge of the slew rates on V_{ds} or V_{gs}. Another useful, but challenging parameter to measure is the On-State voltage (V_{ds-on}). Due to the huge difference between the Off-State voltage and On-State voltage, a sensor would have to differentiate small voltage changes throughout the whole range. Again, measuring R_{ds-on} is unavoidable, especially for degradation cases where R_{ds-on} will drift upward ([21, 22] leading to increases losses, and thermal stress. Generally, any correlations found between V_{ds} or V_{gs} ringing during switching events were avoided. While there may be useful correlations between ringing peak amplitudes and changes to internal switch capacitances, the peak voltages are relatively small in amplitude and occur in nanoseconds time spans. Alternatively, simple measurements surrounding the switch will include the drain current, or V_{ds-off} values. In conclusion difficulties with certain switch measurements encourage only sensing them when either necessary or exceedingly helpful. When possible, the neural network design should try and eliminate or implicitly estimate difficult measurements.

5.2 Input-Output Selection

This section presents a discussion on the network outputs and reasoning behind each chosen input. These aren't the final networks, but the reasoning of where to start and what parameters matter is important to avoid randomly trying inputs until something works. For a summary of the inputs and outputs see Table 5.1 below.

1. **Passive Components (C_s, L_s, L_p):**

   For the passives, it was found that any measurement in the tank will correlate well with changes in the passive values. As just discussed, many resonant tank voltage measurements should be avoided. Instead, the series resonant current will be relied upon for resonant tank information. Other useful measurements to include for this parameter will be the input and output voltages, currents, as well as the control frequency which will contain information from feedback control loops.

2. **Tank Losses:**

   Instead of predicting each and every passive component resistance, the lumped tank loss could be predicted. Differentiating between the series passives resistances would be difficult and may not be a useful regression regardless. For the model to find the tank loss, it must still separate out the switch losses from the total converter power. To do this, the inputs will include the input and output currents and voltages to get the total power. Next, V_{ds-on} measurements and tank series current are used to help define the switch losses. By having the switch losses and total power, the difference will result in the tank lumped loss.

3. **Switch Junction Temperature:**

   To predict the switches final junction temperature, it must learn its average switch power loss, and constants relating to the RC thermal model. For inputs, the case temperature is used along with V_{ds-on} and drain current measurements to define the

switch power loss. The thermal model is constant and can be learned through the data.

4. **On State Resistance:**

   Much like the previous parameter, to estimate R_{ds-on} requires V_{ds-on} and the drain currents. Predicting R_{ds-on} isn't just Ohm's law, but it can get most of the way there with some learned non-linearity. Possibly adding information about the series resonant current will help define the V_{ds-on} shape since it is the resonant current that passes through R_{ds-on}.

5. **Gate-Source Capacitance:**

   The gate source capacitance is mainly defined during the turn-on period as that can be isolated from C_{ds}. Inputs regarding the gate are important here, so to start all gate related measurements are used. These include the gate charge (Q_{gate}), V_{gs}, and the voltage across the gate resistor (V_{rg}). To try and separate the V_{gd} effects during turn-on, V_{ds} is used as well.

6. **Drain-Source Capacitance:**

   Predicting the Drain-Source Capacitance C_{ds} is more difficult. Use of the slew rates could predict it during turn-off, but that would require knowing C_{ds}'s current at that very specific time. Alternatively, finding the charge and voltage associated with C_{ds} when the switch is off could work, but it is hard to separate out the effects of the other switch capacitances. There isn't technically a single switch with a single C_{ds}. Instead, a JFET/MOSFET cascode exists making this even more complex. The best can still be done to include the most relevant information such as V_{ds} Slew rates, V_{ds}, or the series tank current which passes through C_{ds}. From the experiments in Section 3.1.1, a correlation between changes in C_{ds} and Q_{gate} was found to be useful. Including some gate related variables, similar to C_{gs} may prove useful.

Table 5.1: Model Input/Output Summary

Output Parameter	**Inputs**
C_s, L_s, L_p	V_{in}, V_{out}, I_{in}, I_{out}, I_{series}, f_{sw}
P_{tank}	V_{in}, V_{out}, I_{in}, I_{out}, V_{ds-on}, I_d
T_j	V_{in}, V_{out}, I_{in}, I_{out}, I_{series}, V_{ds-on}, f_{sw}
R_{ds-on}	V_{ds-on}, I_d
C_{gs}	Q_{gate}, V_{gs}, $\frac{\partial}{\partial t}V_{gs}$,$V_{rg}$, V_{ds}
C_{ds}	V_{ds}, $\frac{\partial}{\partial t}V_{ds}$, I_{series}, Q_{gate}, V_{gs}
ZVS	V_{ds}, $\frac{\partial}{\partial t}V_{ds}$, I_{series}, f_{sw}, Q_{gate}

7. **Zero-Voltage Switching:**

   ZVS is an important parameter to predict as it can have consequences for efficiency, heating, and degradation of the switch from thermal cycling. Predicting this involves understanding if V_{ds} of a switch will reach zero before the gate signal rises, and the current starts to flow. To calculate this would involve knowing how much charge can be dissipated in C_{ds} by the resonant current. This becomes a complicated calculation to learn. The network should benefit most from information regarding V_{ds}, V_{ds} Slew, I_{series}, and f_{sw} which gives an idea of the phase of the resonant current compared to the switch control phase. Q_{gate} will also be included to help differentiate from the other capacitances since the network won't know what C_{ds}.

CHAPTER 6

Results and Verification

6.1 Results

This sections presents the best results after training the networks. The scores used to quantify a networks performance will be the MAE, RMSE, and R-Squared. The main visual to display the network's performance will be done with a scatter plot of ground-truth vs predicted outputs. If both are closely matching, the network is performing well and the graph should show points closely following a $y = x$ line. For the following results, a dataset of 6394 random points were simulated. This dataset is an important limitation of these results but doesn't invalidate them. The dataset is split in a 70:30 ratio for training and testing. Every regression network used the MSE loss function, every classification network used the BCE loss function, and both used a learning rate of 0.01. The structure of each network has one input layer, one output layer, and a chosen number of hidden layers in between. Each hidden layer has the same size (number of perceptrons). For the complete list of results see the tables in Appendix B. For the table containing of all required measurements see Table B.2.

Starting with C_s in Fig. 6.1 which has an R-Squared of 99.33% and a MAE of 1.327%. The inputs include the max and RMS of the series resonant, input, and output currents. Input and output voltages were also included, but only the output voltage had a max and RMS. The parameter V_{in} is the DC input voltage so it is left constant. Finally, f_{sw} is added. In this case, the Hyperbolic Tangent activation function performed a few percentage points better though this is not always true. The biggest factors in improving the network were to increase the size of the hidden layers and to include both the max and RMS versions of the inputs. The figure referenced shows a good fit except near the boundaries. The network

has trouble at the boundary layer which suggests a buffer of extra training data could help smooth that out. Using the same inputs, L_s was able to an R-Squared of 99.14% and a MAE of 0.5729%. Figure 6.2 shows the scatter plot fit. The Tanh activation function also worked the best here. It does a great job, but again near the edges the model starts to break down. Finally, using the same inputs, L_p was able to an R-Squared of 90.87% and a MAE of 4.076%. Figure 6.3 shows the scatter plot fit. L_p is much more difficult to get working well compared to the other passives and still suffers from the boundary errors even worse than the others. Despite that, points in the middle follow the trend well but with more error than before.

The next parameter is the lumped resonant tank loss caused by the ESR's of the components. The inputs for this were top and bottom V_{ds-on} min and max, the series current RMS, both input and output currents' RMS, $V_{out-max}$, V_{in} and f_{sw}. The network was able to be kept small and used the ReLU activation function. The best case resulted in an R-Squared of 97.99% and a MAE of 0.2984%. Figure 6.4 is the scatter plot fit. Notice that most of the data for the loss was concentrated at the bottom end of the plot. Plotting a histogram of the true outputs and predictions can help visualize how the data is distributed. Figure 6.5 makes it clear that the majority of the losses are around one spike with a minority around its base. This makes it hard to conclude whether the network has found a general solution to tank loss without a better dataset range for the tank losses. This could be done by deliberately choosing circuit configurations with higher losses, but this may also deviate from real converter behavior.

Next up are the top and bottom final junction temperatures. They both used the same inputs, with the respective top and bottom versions. Here V_{ds-on} and the drain currents were used along with the normal input and output currents, and input and output voltages, with f_{sw} and T_c. Again the network was kept small with 1000 epochs and used ReLU. The best case for the top resulted in an R-Squared of 98.51% and a MAE of 0.5039%. The best case for the bottom resulted in an R-Squared of 98.29% and a MAE of 0.3693%. The scatter plot

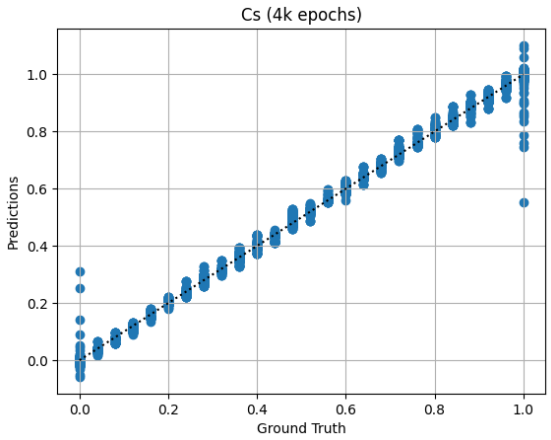

Figure 6.1: Scatter plot of best performance for C_s

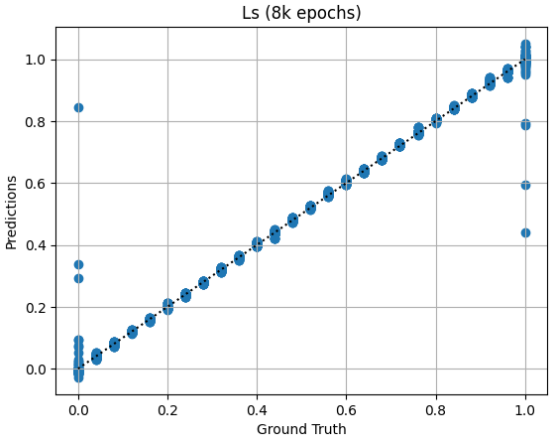

Figure 6.2: Scatter plot of best performance for L_s

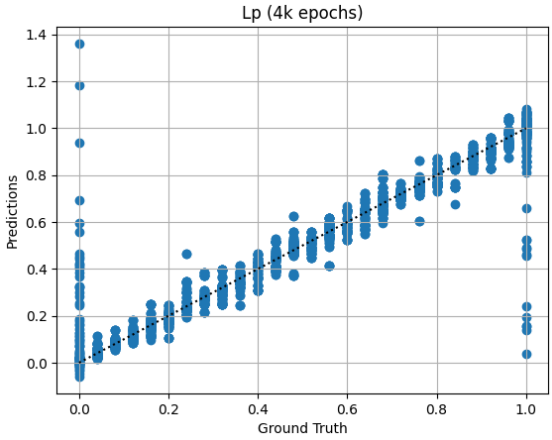

Figure 6.3: Scatter plot of best performance for C_s

results are in Fig. 6.6 and 6.7. These plots appear to suggest that the top temperatures are more evenly distributed while the bottoms ones are not. Looking at Fig. 6.8 and 6.9 shows that again, one main spike accounts for the majority of the data in both. This converter with this operating point doesn't often produce wide temperature ranges. This is something to keep in mind as either the dataset must be added to with more temperature variation, or temperatures outside this range must be tested before applying to health monitoring or control.

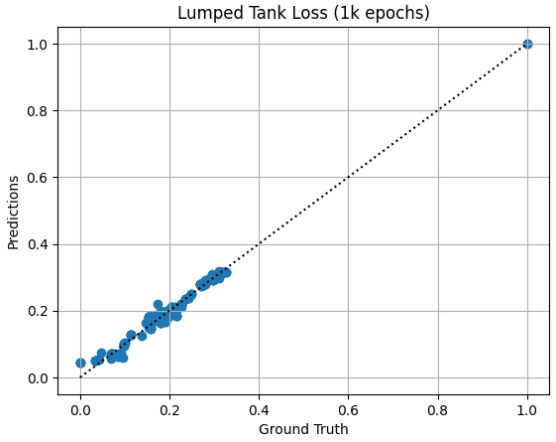

Figure 6.4: Scatter plot of best performance for P_{tank}

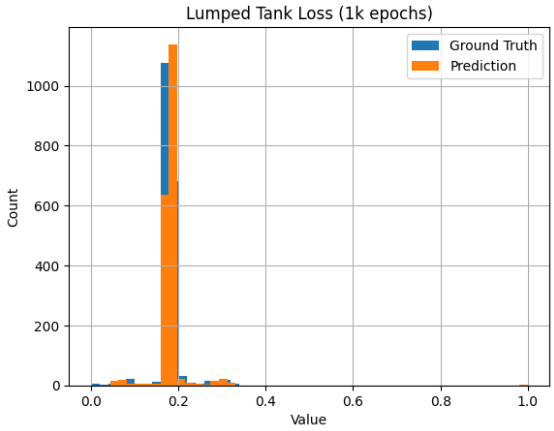

Figure 6.5: Histogram for the best performance for bottom final P_{tank}

Moving onto the top and bottom R_{ds-on}. They both used the same inputs, with the respective top and bottom versions. Inputs only include the max and min V_{ds-on}, and the max, min and average I_d, with f_{sw} added. The network improved from increasing the hidden layer size to 30 and using Leaky ReLU. The best case for the top resulted in an R-Squared of 98.72% and a MAE of 1.547%. The best case for the bottom resulted in an R-Squared of 98.68% and a MAE of 1.473%. The scatter plot results are in Fig. 6.10 and 6.11. The fit is quite good with only a few bad predictions either on the boundary or right in the middle.

Following up are the C_{gs} results. At first the inputs consisted of Q_{gate}, V_{gs}, V_{rg}, and V_{ds} along with a small network using ReLU. The results were not great with a MEA of 11.5%. Experimentation with the network was not helping so the inputs were reconsidered. An important measurement missing was the slew rate ($\frac{\partial}{\partial t}V_{gs}$). Upon adding this, results improve but not until the network was expanded to have hidden layers of 40 perceptrons wide. The best results for the top are an R-Squared of 98.26% and a MAE of 2.291%. The best case for the bottom had an R-Squared of 97.58% and a MAE of 1.961%. Figures 6.12 and 6.13 show a good fit with the boundaries still having trouble fitting.

Moving onto the C_{ds} measurements which were the most trouble. The best cases used the max, min, average, and RMS of V_{ds} slew. Also important were measurements like Q_{gate},

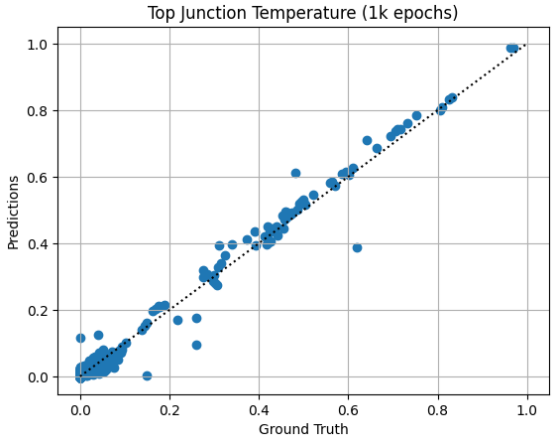

Figure 6.6: Scatter plot of best performance for the top final T_J

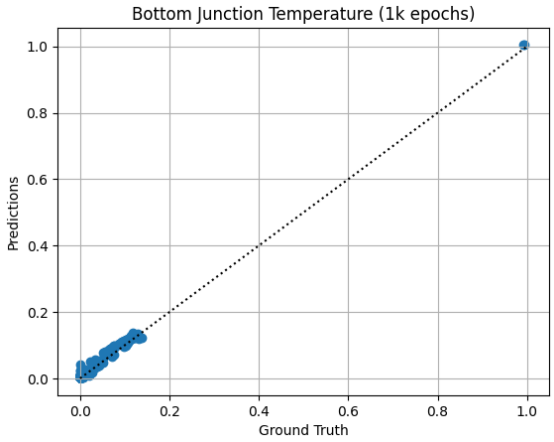

Figure 6.7: Scatter plot of best performance for bottom final T_J

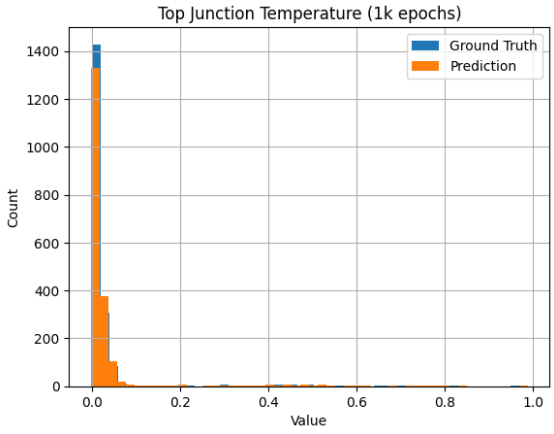

Figure 6.8: Histogram plot of best performance for the top final T_J

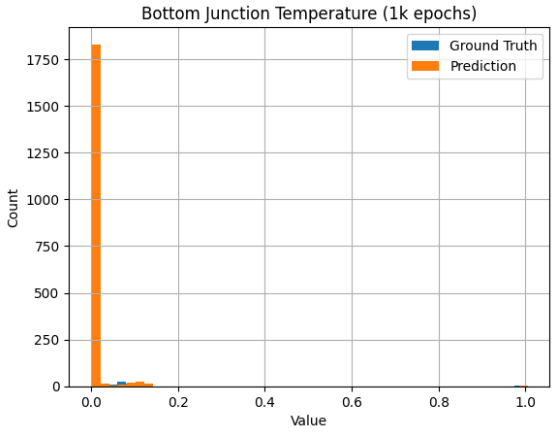

Figure 6.9: Histogram plot of best performance for bottom final T_J

V_{gs}, $V_{ds-on-max}$, and $I_{series-rms}$. The network proved better with a hidden layer size of 40 and Tanh as the activation function. In the case of the bottom C_{ds}, Leaky ReLU was slightly better. The best case for the top has an R-Squared of 76.09% and a MAE of 6.483%. The best case for the bottom has an R-Squared of 77.37% and a MAE of 6.388%. The scatter plots are shown in Fig. 6.14 and 6.15. The overall fit it good, but the edges are even worse. This is reflected in the RMSE's of 15.37% and 14.63% for the top and bottom. RMSE will weigh the large deviations greatly reflecting the errors on the boundaries.

Finally, the last prediction is the binary classification of ZVS occurring for the top

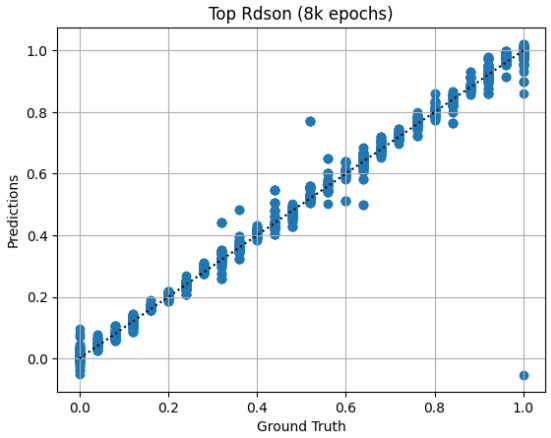

Figure 6.10: Scatter plot of best performance for top R_{dson}

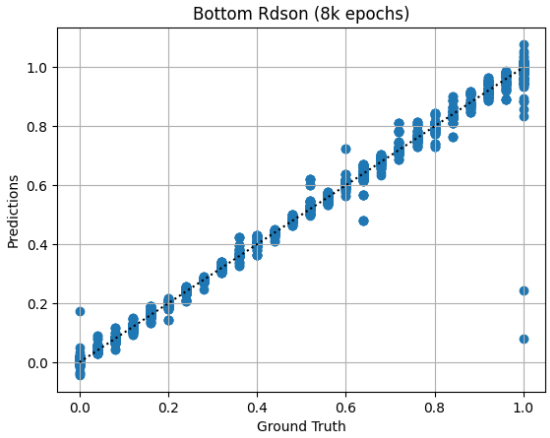

Figure 6.11: Scatter plot of best performance for bottom R_{dson}

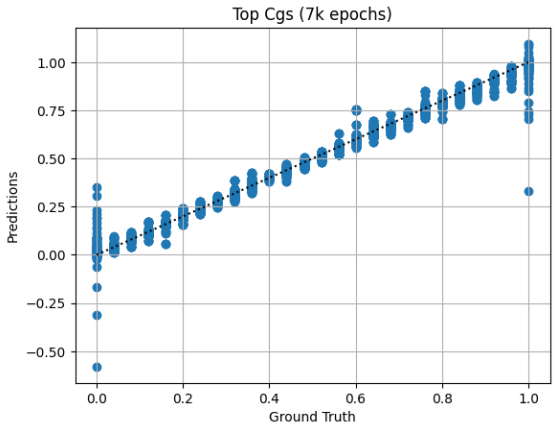

Figure 6.12: Scatter plot of best performance for top C_{gs}

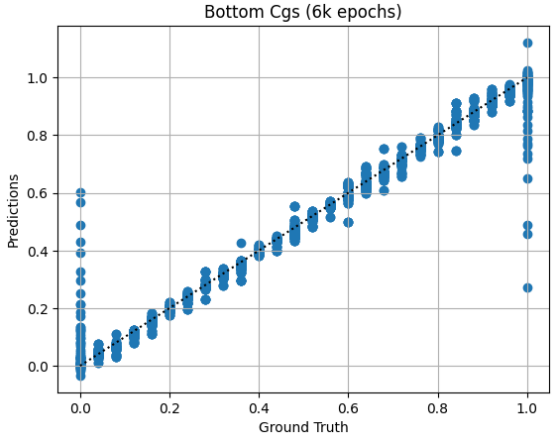

Figure 6.13: Scatter plot of best performance for bottom C_{gs}

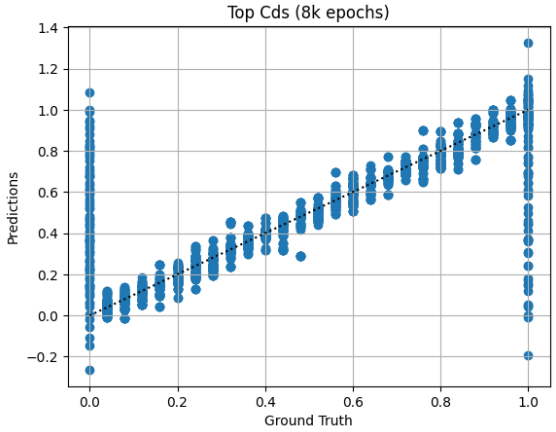

Figure 6.14: Scatter plot of best performance for top C_{ds}

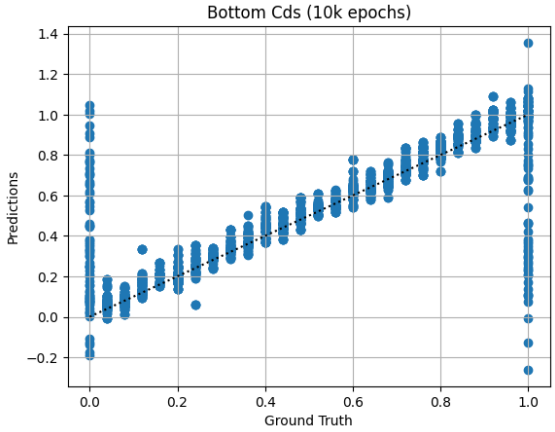

Figure 6.15: Scatter plot of best performance for bottom C_{ds}

and bottom switches. Due to the frequency control and operating point of the converter, the dataset is overwhelmingly made up of ZVS occurrences being true which will certainly make it hard to tell if the network is learning general principals or just guessing positives. For context, 98% of the top ZVS data is true, and 98.5% of the bottom ZVS data is true. The testing dataset itself may vary as the test samples are randomly chosen, but these numbers give a baseline for performance. If the network only learns to output a positive top ZVS, then an accuracy of around 98% will be achieved, and the precision score will be the exact same, while recall will be a perfect 100% due to never guessing a negative ZVS. The inputs used didn't change from the input/output discussion in Section 5.2. There was some experimentation with the network size and variations of those base input parameters. For all classifications, the activation function used was sigmoid. For results, the highest accuracy achieved for the top was 99.32% with a precision of 99.89% and recall of 99.42%. In this case, the network did not simply always output positives based on the criteria described above.

For the bottom ZVS scenario at network widths of 10+, the scores all reached 100% with only 500 epochs. The perfect scores with minimal training suggest the network may not have learned a true generalized rule. To avoid this, a smaller network of 2 layers with width 8 was found to work well. The highest accuracy achieved was 99.74% with a precision of 99.89% and recall of 99.84%. Figures 6.16 and 6.17 show the top and bottom histograms where any bar > 0.5 is considered a positive output. Given the high accuracy, the predictions almost completely occlude the ground truth data. Because the ZVS data is already so skewed, it is hard to tell how well this network can generalize to cases where ZVS is off. In the case of control, assuming ZVS is true when it isn't could lead to higher losses, temperatures and therefore depredation. The precision score is perhaps more important since it outlines how often ZVS was predicted to be true when it was false.

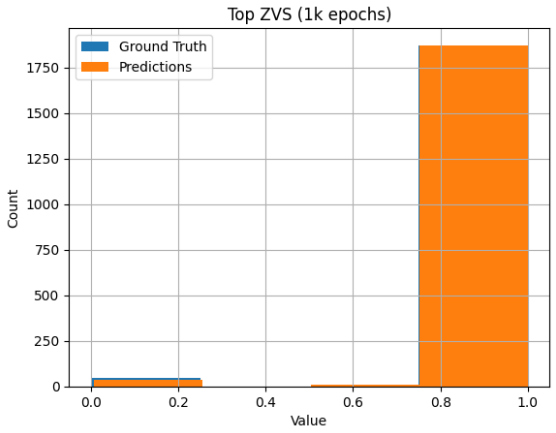

Figure 6.16: Histogram of best performance for top ZVS classification

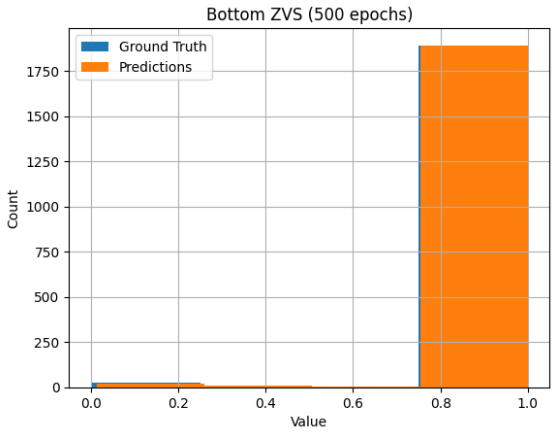

Figure 6.17: Histogram of best performance for bottom ZVS classification

6.2 Discussion

From this data three lessons can be learned about the work. First, many networks have trouble properly fitting the boundaries of the dataset. This may be a consequence of warping the neural network to fit most of the data, but failing to align this fit at the edges. A few different strategies could be used to help fix this. First, the dataset could be expanded just beyond the parameter range boundaries so the boundary errors happen outside the range. The next solution is to specifically add more of the minimum and maximum boundary values such that the dataset histogram will have two pillars at its edges. This may help to reinforce the boundaries on increasing the pure number of examples. Finally, modifying the cost function to add a penalty term based on a measure of that boundary error could help to encourage a better fit. Perhaps using RMSE or the ratio of RSME to MAE will help identify those rough boundary errors.

The second lesson, also regarding the dataset, is that a few parameters have tight distributions that do not give the network a wide range of training examples. ZVS was probably the worst example of this, but the Junction Temperatures, and Lumped Tank Loss also suffered from this. Having a dataset with these variables more evenly distributed would help train a more robust network, capable of accurately predicting a larger range and avoid

overfitting to a peak. On the other hand, the converter operation of interest may never realistically produce values in a uniform range and this seemingly biased dataset is actually desired. It may be enough to use a skewed dataset if the converter won't leave certain ranges, and the network can do a good job of discriminating rare outputs that deviate from the main peak. Looking at the scatter plots for the Tank Loss, and both Junctions Temperatures (Fig. 6.4, 6.6, and 6.7) do show a decent fit suggesting the network can successfully follow points outside the main peak even if rare.

Finally, a pattern found to occur with the networks was that expanding the hidden layer size was more important than increasing the depth. In other words, width was more important than depth. One concern of wider networks is that the network is over-parameterized and is memorizing or overfitting the training data. Looking at Fig. 6.18 shows the training set and predictions together. This shows that the network did not just memorize the training data which is backed up by the fact the training loss never reached 0%. Figures 6.19, 6.20, and 6.21 all had wide networks of 30 or 40. These figures show that the training set was not memorized and interpolation is being performed in between the dataset points.

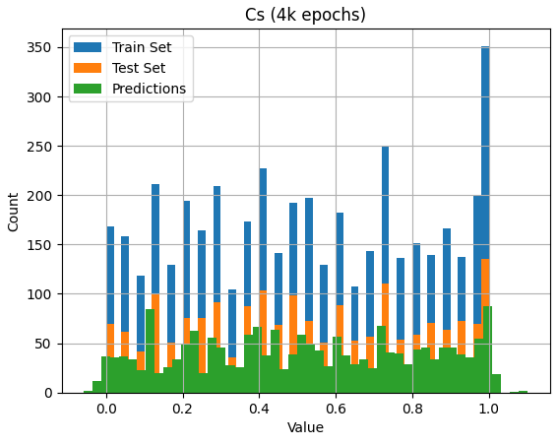

Figure 6.18: Histogram of the Train, Test and Prediction Data for C_s

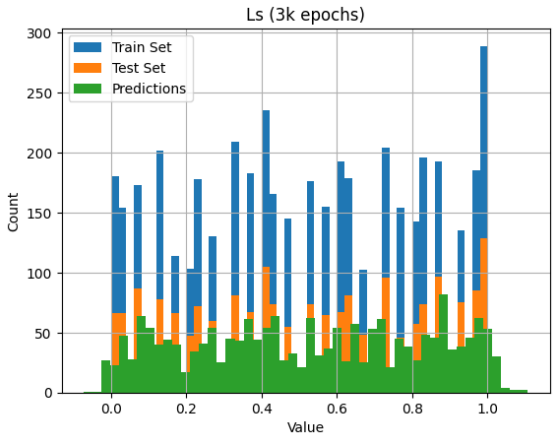

Figure 6.19: Histogram of the Train, Test and Prediction Data for L_s

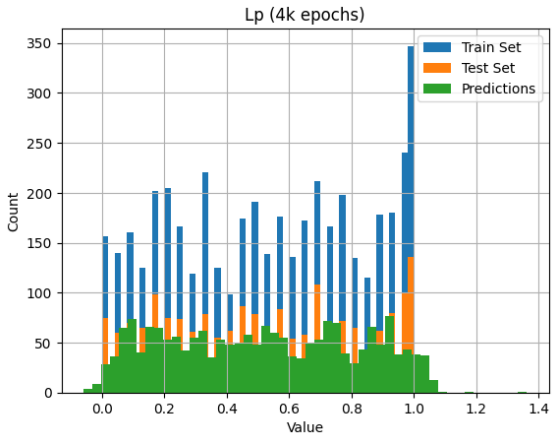

Figure 6.20: Histogram of the Train, Test and Prediction Data for L_p

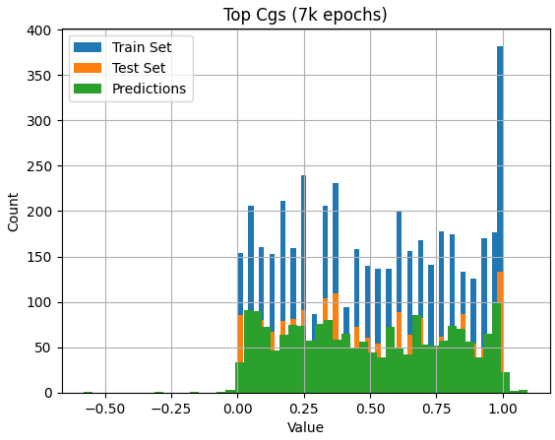

Figure 6.21: Histogram of the Train, Test and Prediction Data for Top C_{gs}

CHAPTER 7

Conclusion

This thesis applied simple Feed Forward Neural Networks to a Half-Bridge LLC Resonant converter to create a Digital Twin capable of predicting internal component values such as the resonant tank passives, or switch parameters including the Junction Temperature, Gate-Source capacitance or the On-State Resistance. First, the converter was modeled and designed before going through a simulation procedure capable of collecting and producing a dataset. The concepts of Neural Networks and their training was discussed, completed by proposing the input vectors for each desired output to be predicted. Finally, the networks were tuned and the best of each parameter predicted was presented and discussed. The resonant passive component values were all able to achieve an MAE below 4.08%, C_{gs} MAEs at or below 2.29%, C_{ds} MAEs at or below 6.48%, R_{ds-on} MAEs below 1.55%, the Final T_J MAEs below 0.504%, and finally, a Lump Tank Power Loss with an MAE of 0.298%. For Turn-On ZVS classifications, the accuracies were at or above 99.3%, both with a precision of 99.89%. Regression results showed that the boundaries of the dataset presented large errors. Those specific predictions could have errors much larger than the Mean Absolute Error. Changes in the dataset or training may alleviate this problem. Another limitation found is that some of the parameters did not have wide distributions to train on due to the physics of the converter operation. This only effected the Junction Temperatures, Lumped Tank Loss, and ZVS parameters. The biggest limitation was the size of the dataset being only just over 6000 samples. Despite this, the results were favorable with small enough errors to be useful for condition monitoring or control.

7.0.1 Future Work

There are a few directions this work could be expanded upon to become more useful. First, a series of real converter models could be used to verify the DT at different operating points. If the DT and real converter tend to be in good agreement, this method of modeling may be preferred to other approaches. Second, the converter model itself can be expanded beyond the AC fundamental approximation taken in the modeling stage. A transformer model with distributed parasitics, and a complete rectifier are the first expansions for the LLC Resonant Converter. This will extend the simulation time but would offer data to more accurately characterize the whole converter behavior. On a different note, an important step forward towards real application would be to get the DT models running online with an FPGA. Ideally, this will run as fast as the control loops frequency for use in cycle-by-cycle control. Some of the papers from Section 1.2 defined ways to get models running on an FPGA using High-Level Synthesis of C-Code, though not all were ANN based. Initial tests show that on a GTX 750ti GPU having just under 1.4 PFLOPS for 32 bit float point numbers, the inference tasks took about 2.7μs per sample. This test demonstrates that it may be possible to use an FPGA to run the networks faster than the switching frequency. This is still dependent on many factors relating to the neural networks, FPGA hardware, logic implementation, and sensing capabilities which is why it must be attempted. Finally, a couple different machine learning models should be explored. The two main models are time-series architectures, and Physics-Informed Neural Networks (PINN). The power of time-series forecasting is that converters are dynamics systems and the ability to properly incorporate a sense of time is not only more accurate but may lead to better predictions with less training data. These time-series could be on time scales useful for health monitoring or cycle-by-cycle updates for control. Architectures such as NARX [15, 16] would be a good place to start. Equally important is trying use the established physics knowledge of converters to help shrink the search space for ANN solutions, and reduce the amount of data needed to train. PINNs help do this by incorporating known dynamics into the ANN and learning

process [32, 33]. By forcing or rewarding the compliance of physically correct behavior, the network should adhere to more physically accurate models, as well as better interpolate and even extrapolate results. One example architecture is SINDy or Sparse Identification of of nonlinear dynamical systems, which learns the form of a non-linear differential equation [34]. SINDy can be used along with auto-encoders to isolate the most important physical variables in low dimensional space, and directly learn the dynamics using those [33]. In summary, PINNs such as these are useful in reducing simulation time, increasing accuracy, while also improving interpolation and extrapolation.

APPENDIX A

JFET Cascode Structure

This appendix gives a brief overview of the Cascode JFET structure model used in this thesis. The device works by cascoding a depletion-mode N-Channel JFET with a small N-Channel MOSFET. The JFET being normally open would require negative voltages to turn it off. Instead of direct control, a small cascode MOSFET is used to open or close the circuit allowing current flow through the switch. This structures offers the advantages of easy switching, and reverse conduction through the MOSFET body-diode while taking advantage of SiC's better density and efficiency.

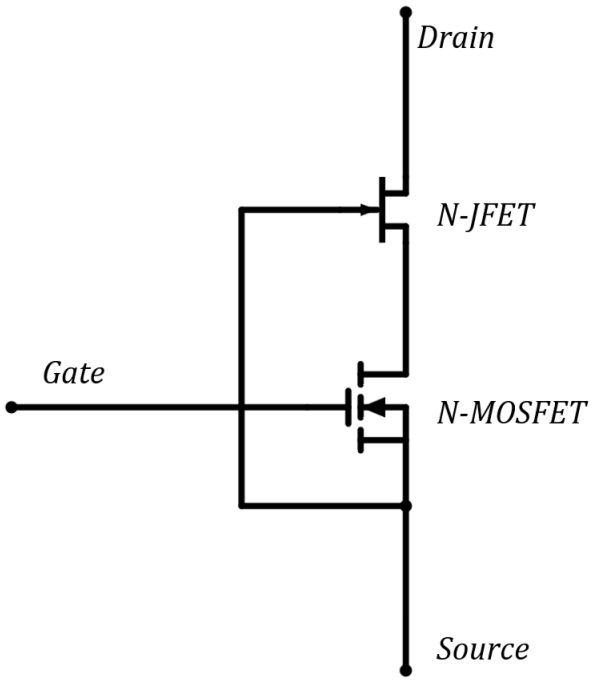

Figure A.1: JFET Cascode Switch Structure

APPENDIX B

Model Results Tables

Presented below is a table of all the final, best parameter models with their inputs, activation functions, number of epochs trained, number of hidden layers, size of the hidden layers, and the performance scores RMSE, MAE, and R-Squared, or Accuracy, Precision and Recall for classification.

Table B.1: Final Model Input/Output Selection

Output Parameter	**Inputs**	**Activation Function**	**Epochs**	**Number of Hidden Layers**	**Size of Hidden Layers**	**RMSE**	**MAE**	R^2
C_s	$I_{series-rms}$ $I_{series-max}$ $I_{out-rms}$ $I_{out-max}$ I_{in-rms} I_{in-max} $V_{out-rms}$ $V_{out-max}$ V_{in} f_{sw}	Tanh	4000	4	40	2.476%	1.327%	99.33%
L_s	$I_{series-rms}$ $I_{series-max}$ $I_{out-rms}$ $I_{out-max}$ I_{in-rms} I_{in-max} $V_{out-rms}$ $V_{out-max}$ V_{in} f_{sw}	Tanh	8000	4	30	2.873%	0.5729%	99.14%
L_p	$I_{series-rms}$ $I_{series-max}$ $I_{out-rms}$ $I_{out-max}$ I_{in-rms} I_{in-max} $V_{out-rms}$ $V_{out-max}$ V_{in} f_{sw}	Leaky ReLU	4000	4	30	9.382%	4.076%	90.87%
P_{tank}	$V_{ds-on-max}$ top $V_{ds-on-min}$ top $V_{ds-on-max}$ bottom $V_{ds-on-min}$ bottom $I_{series-rms}$ $I_{out-rms}$ I_{in-rms} $V_{out-max}$ V_{in} f_{sw}	ReLU	1000	4	12	0.5379%	0.2984%	97.99%

Output Parameter	**Inputs**	**Activation Function**	**Epochs**	**Number of Hidden Layers**	**Size of Hidden Layers**	**RMSE**	**MAE**	R^2
T_J Final Top	$V_{ds-on-max}$ top $V_{ds-on-min}$ top I_{d-max} top I_{d-min} top $I_{out-max}$ I_{in-max} $V_{out-max}$ V_{in} f_{sw} T_c	ReLU	1000	4	12	1.193%	0.5039%	98.51%
T_J Final Bottom	$V_{ds-on-max}$ bottom $V_{ds-on-min}$ bottom I_{d-max} bottom I_{d-min} bottom $I_{out-max}$ I_{in-max} $V_{out-max}$ V_{in} f_{sw} T_c	ReLU	1000	4	12	0.4771%	0.3693%	98.29%
R_{ds-on} Top	$V_{ds-on-max}$ top $V_{ds-on-min}$ top I_{d-max} top I_{d-min} top I_{d-avg} top f_{sw}	Leaky ReLU	8000	4	30	3.463%	1.547%	98.72%
R_{ds-on} Bottom	$V_{ds-on-max}$ bottom $V_{ds-on-min}$ bottom I_{d-max} bottom I_{d-min} bottom I_{d-avg} bottom f_{sw}	Leaky ReLU	8000	4	30	3.538%	1.473%	98.68%

Output Parameter	**Inputs**	**Activation Function**	**Epochs**	**Number of Hidden Layers**	**Size of Hidden Layers**	**RMSE**	**MAE**	R^2
C_{gs} Top	$\frac{\partial}{\partial t}V_{gs}$ Max top $\frac{\partial}{\partial t}V_{gs}$ Min top $Q_{gate-max}$ top $Q_{gate-rms}$ top V_{gs-max} top V_{rg-max} top V_{rg-rms} top $V_{ds-on-min}$ top	Leaky ReLU	7000	4	40	4.138%	2.291%	98.26%
C_{gs} Bottom	$\frac{\partial}{\partial t}V_{gs}$ Max bottom $\frac{\partial}{\partial t}V_{gs}$ Min bottom $Q_{gate-max}$ bottom $Q_{gate-rms}$ bottom V_{gs-max} bottom V_{rg-max} bottom V_{rg-rms} bottom $V_{ds-on-min}$ bottom	Leaky ReLU	6000	4	40	4.753%	1.961%	97.58%
C_{ds} Top	$\frac{\partial}{\partial t}V_{ds}$ Max top $\frac{\partial}{\partial t}V_{ds}$ Avg top $\frac{\partial}{\partial t}V_{ds}$ RMS top $\frac{\partial}{\partial t}V_{ds}$ Min top $Q_{gate-max}$ top V_{gs-max} top $V_{ds-on-max}$ top $I_{series-rms}$	Tanh	8000	4	40	15.37%	6.483%	76.09%
C_{ds} Top	$\frac{\partial}{\partial t}V_{ds}$ Max bottom $\frac{\partial}{\partial t}V_{ds}$ Avg bottom $\frac{\partial}{\partial t}V_{ds}$ RMS bottom $\frac{\partial}{\partial t}V_{ds}$ Min bottom $Q_{gate-max}$ bottom V_{gs-max} bottom $V_{ds-on-max}$ bottom $I_{series-rms}$	Leaky ReLU	10000	4	40	14.63%	6.388%	77.37%

Output Parameter	**Inputs**	**Activation Function**	**Epochs**	**Number of Hidden Layers**	**Size of Hidden Layers**	**Accuracy**	**Precision**	**Recall**
ZVS Top	$V_{ds-on-max}$ Top $\frac{\partial}{\partial t}V_{ds}$ Min top $\frac{\partial}{\partial t}V_{ds}$ RMS top $\frac{\partial}{\partial t}V_{ds}$ Avg top $\frac{\partial}{\partial t}V_{ds}$ Max top $I_{series-max}$ top $Q_{gate-max}$ top	Sigmoid	1000	8	30	99.32%	99.89%	99.42%
ZVS Bottom	$\frac{\partial}{\partial t}V_{ds}$ Min bottom $\frac{\partial}{\partial t}V_{ds}$ RMS bottom $\frac{\partial}{\partial t}V_{ds}$ Avg bottom $\frac{\partial}{\partial t}V_{ds}$ Max bottom $I_{series-max}$ bottom $Q_{gate-max}$ bottom	Sigmoid	500	2	8	99.74%	99.89%	99.84%

Table B.2: All Required Measurements

Measurement	**Variations**
I_{series}	Max, RMS
I_{out}	Max, RMS
I_{in}	Max, RMS
V_{out}	Max, RMS
V_{ds-on}	Max, Min
I_d	Max, Min, Average
V_{gs}	Max
V_{gs} Slew	Max
$^*Q_{gate}$ / V_{rg}	Max, RMS
V_{ds} Slew	Max, Min, Average, RMS
V_{in}	DC value
f_{sw}	Controller value
T_c	Single Sample

*The gate charge is the integration of current through the gate resistor. This relates to V_{rg} by Ohm's law and the gate resistance. This is why they are on the same line.
All the Min, Max, Average, and RMS measurements are taken over three switching cycles.

BIBLIOGRAPHY

[1] K. Sado, J. Hannum, and K. Booth, "Digital Twin Modeling of Power Electronic Converters," in *2023 IEEE Electric Ship Technologies Symposium (ESTS)*, (Alexandria, VA, USA), pp. 86–90, IEEE, Aug. 2023.

[2] J. Xiong, H. Ye, W. Pei, K. Li, and Y. Han, "Real-time FPGA-digital twin monitoring and diagnostics for PET applications," in *2021 6th Asia Conference on Power and Electrical Engineering (ACPEE)*, (Chongqing, China), pp. 531–536, IEEE, Apr. 2021.

[3] A. Wong, J. Cronin, and E. Santi, "Digital Twin Approach Enables Switching Converter Adaptive Control for All-Electric Ship Power Distribution System," in *2023 IEEE Electric Ship Technologies Symposium (ESTS)*, (Alexandria, VA, USA), pp. 154–160, IEEE, Aug. 2023.

[4] S. Chen, S. Wang, P. Wen, and S. Zhao, "Digital Twin for Degradation Parameters Identification of DC-DC Converters Based on Bayesian Optimization," in *2021 IEEE International Conference on Prognostics and Health Management (ICPHM)*, (Detroit (Romulus), MI, USA), pp. 1–9, IEEE, June 2021.

[5] M. Milton, C. De La O, H. L. Ginn, and A. Benigni, "Controller-Embeddable Probabilistic Real-Time Digital Twins for Power Electronic Converter Diagnostics," *IEEE Transactions on Power Electronics*, vol. 35, pp. 9850–9864, Sept. 2020.

[6] G. Di Nezio, M. Di Benedetto, A. Lidozzi, and L. Solero, "Digital Twin based Real-Time Analysis of DC-DC Boost Converters," in *2022 IEEE Energy Conversion Congress and Exposition (ECCE)*, (Detroit, MI, USA), pp. 1–7, IEEE, Oct. 2022.

[7] M. T. Fard and J. He, "Digital Twin Health Monitoring of Five-Level ANPC Power Converter based on Estimation of Semiconductor On-State Resistance," in *2023 IEEE*

Industry Applications Society Annual Meeting (IAS), (Nashville, TN, USA), pp. 1–7, IEEE, Oct. 2023.

[8] Y. Gong, Y. Tian, C. Wen, H. Luo, C. Li, and W. Li, "Digital Twin Based Condition Monitoring for High Power Converters," in *2022 IEEE International Power Electronics and Application Conference and Exposition (PEAC)*, (Guangzhou,Guangdong, China), pp. 892–897, IEEE, Nov. 2022.

[9] Y. Peng, S. Zhao, and H. Wang, "A Digital Twin Based Estimation Method for Health Indicators of DC–DC Converters," *IEEE Transactions on Power Electronics*, vol. 36, pp. 2105–2118, Feb. 2021.

[10] Q. Wu, W. Wang, Q. Wang, L. Xiao, and B. Hu, "Digital Twin Approach for Degradation Parameters Identification of a Single-Phase DC-AC Inverter," in *2022 IEEE Applied Power Electronics Conference and Exposition (APEC)*, (Houston, TX, USA), pp. 1725–1730, IEEE, Mar. 2022.

[11] S. Rajendran, V. S. K. Devi, and M. Diaz, "Digital twin based identification of degradation parameters of DC-DC converters using an Arithmetic Optimization Algorithm," in *2022 3rd International Conference for Emerging Technology (INCET)*, (Belgaum, India), pp. 1–5, IEEE, May 2022.

[12] X. Cheng, J. Chen, Y. Chen, X. Zhao, Z. Zhou, Q. Yang, and N. Y. Dai, "Digital-Twin-Driven PI Parameter Evaluation Method for Grid-Connected Converters," in *2023 4th International Conference on Power Engineering (ICPE)*, (Macau, Macao), pp. 26–31, IEEE, Dec. 2023.

[13] X. Song, T. Jiang, S. Schlegel, and D. Westermann, "Parameter tuning for dynamic digital twins in inverter-dominated distribution grid," *IET Renewable Power Generation*, vol. 14, pp. 811–821, Apr. 2020.

[14] H. S. Krishnamoorthy and T. Narayanan Aayer, "Machine Learning based Modeling of Power Electronic Converters," in *2019 IEEE Energy Conversion Congress and Exposition (ECCE)*, (Baltimore, MD, USA), pp. 666–672, IEEE, Sept. 2019.

[15] A. Wunderlich and E. Santi, "Digital Twin Models of Power Electronic Converters Using Dynamic Neural Networks," in *2021 IEEE Applied Power Electronics Conference and Exposition (APEC)*, (Phoenix, AZ, USA), pp. 2369–2376, IEEE, June 2021.

[16] G. Rojas-Duenas, J.-R. Riba, K. Kahalerras, M. Moreno-Eguilaz, A. Kadechkar, and A. Gomez-Pau, "Black-Box Modelling of a DC-DC Buck Converter Based on a Recurrent Neural Network," in *2020 IEEE International Conference on Industrial Technology (ICIT)*, (Buenos Aires, Argentina), pp. 456–461, IEEE, Feb. 2020.

[17] Xu She and A. Huang, "Solid state transformer in the future smart electrical system," in *2013 IEEE Power & Energy Society General Meeting*, (Vancouver, BC), pp. 1–5, IEEE, 2013.

[18] G. Wang, S. Baek, J. Elliott, A. Kadavelugu, F. Wang, X. She, S. Dutta, Y. Liu, T. Zhao, W. Yao, R. Gould, S. Bhattacharya, and A. Q. Huang, "Design and hardware implementation of Gen-1 silicon based solid state transformer," in *2011 Twenty-Sixth Annual IEEE Applied Power Electronics Conference and Exposition (APEC)*, (Fort Worth, TX, USA), pp. 1344–1349, IEEE, Mar. 2011.

[19] R. Steigerwald, "A comparison of half-bridge resonant converter topologies," *IEEE Transactions on Power Electronics*, vol. 3, pp. 174–182, Apr. 1988.

[20] "1200V-8.6mW SiC FET," UF3SC120009K4S Datasheet, Qorvo, Dec. 2019.

[21] M. Farhadi, B. T. Vankayalapati, and B. Akin, "Reliability Evaluation of SiC MOSFETs Under Realistic Power Cycling Tests," *IEEE Power Electronics Magazine*, vol. 10, pp. 49–56, June 2023.

[22] M. Farhadi, F. Yang, S. Pu, B. T. Vankayalapati, and B. Akin, "Temperature-Independent Gate-Oxide Degradation Monitoring of SiC MOSFETs Based on Junction Capacitances," *IEEE Transactions on Power Electronics*, vol. 36, pp. 8308–8324, July 2021.

[23] H. Jiang, X. Qi, G. Qiu, X. Zhong, L. Tang, H. Mao, Z. Wu, H. Chen, and L. Ran, "A Physical Explanation of Threshold Voltage Drift of SiC MOSFET Induced by Gate Switching," *IEEE Transactions on Power Electronics*, vol. 37, pp. 8830–8834, Aug. 2022.

[24] K. Puschkarsky, T. Grasser, T. Aichinger, W. Gustin, and H. Reisinger, "Review on SiC MOSFETs High-Voltage Device Reliability Focusing on Threshold Voltage Instability," *IEEE Transactions on Electron Devices*, vol. 66, pp. 4604–4616, Nov. 2019.

[25] S. A. Aghdam, M. Agamy, Z. Li, and P. Losee, "Electro-thermal design of mv sic jfet based solid state circuit breakers," in *2023 IEEE 10th Workshop on Wide Bandgap Power Devices Applications (WiPDA)*, pp. 1–6, 2023.

[26] DongHoonPark, "ltspice pytool 1.0.6." `https://github.com/DongHoonPark/ltspice_pytool`. Accessed: April 2024.

[27] "Pytorch 2.0.1." `https://github.com/pytorch/pytorch`. Accessed: April 2024.

[28] T. Szandała, "Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks," in *Bio-inspired Neurocomputing* (A. K. Bhoi, P. K. Mallick, C.-M. Liu, and V. E. Balas, eds.), vol. 903, pp. 203–224, Singapore: Springer Singapore, 2021. Series Title: Studies in Computational Intelligence.

[29] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2014.

[30] M. Han and Y. Zhong, "Analysis and Measurement of Resonant Tank Current on LLC," Application Report SLUA690, Texas Instruments, July 2013.

[31] "Sensors for Power Electronics," in *Power Electronics Applied to Industrial Systems and Transports*, pp. 1–73, Elsevier, 2016.

[32] M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, Feb. 2019.

[33] K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of coordinates and governing equations," *Proceedings of the National Academy of Sciences*, vol. 116, pp. 22445–22451, Nov. 2019.

[34] S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 113, pp. 3932–3937, Apr. 2016.