2022

Establishing a Machine Learning Framework for Discovering Novel Phononic Crystal Designs

Drew Feltner
Wright State University

ESTABLISHING A MACHINE LEARNING FRAMEWORK FOR DISCOVERING NOVEL PHONONIC CRYSTAL DESIGNS

A thesis submitted in partial fulfillment of the
requirements for the degree of
Master of Science

By
DREW FELTNER
B.S.E.E., Wright State University, 2018

2022
Wright State University

WRIGHT STATE UNIVERSITY
GRADUATE SCHOOL

July 27th, 2022

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Drew Feltner ENTITLED "Establishing a Machine Learning Framework for Discovering Novel Phononic Crystal Designs" BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science.

_____

Amit Sharma, Ph. D.
Thesis co-Advisor

_____

Chandriker Dass, Ph. D.
Thesis co-Advisor

_____

Ivan Medvedev, Ph.D.
Chair, Department of Physics

Final Examination Committee

_____
Amit Sharma, Ph. D.

_____
Chandriker Dass, Ph. D.

_____
Ivan Medvedev, Ph. D.

_____
Barry Milligan, Ph. D.
Dean of the Graduate School

Abstract

Feltner, Drew. M.S. Department of Physics, Wright State University, 2022. Establishing a Machine Learning Framework for Discovering Novel Phononic Crystal Designs.

A phonon is a discrete unit of vibrational motion that occurs in a crystal lattice. Phonons and the frequency at which they propagate play a significant role in the thermal, optical, and electronic properties of a material. A phononic material/device is similar to a photonic material/device, except that it is fabricated to manipulate certain bands of acoustic waves instead of electromagnetic waves. Phononic materials and devices have been studied much less than their photonic analogues and as such current materials exhibit control over a smaller range of frequencies. This study aims to test the viability of machine learning, specifically neural networks in aiding in phononic crystal design. Multiple combinations of training datasets, neural network configuration, and data formatting methods are attempted with performance metrics recorded. A novel inverse design scheme is proposed that utilizes phonon density of states to perform prediction of phononic crystal parameters given a desired band gap and center frequency.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

I would like to thank everyone in the AFRL RXAN phononics group and my committee for supporting me throughout this research. More specifically, I'd like to thank my advisor Dr. Chandriker Dass (Kavir) for his guidance and making this opportunity possible, my co-advisor Dr. Sharma for helping to keep me on track, Dr. Robert Bedford for all the very useful brainstorming sessions, and Debanik Das for generating the training data and helping out whenever I had questions.

I would also like to thank all parties involved in my funding throughout this project. This includes RXAN for providing funding, SOCHE for employing me, and Booz Allen Hamilton who funded my tuition. I'd like to especially thank all my coworkers and management chain at Booz Allen Hamilton for being more supportive than I could ever ask for throughout my time in grad school.

Finally, I'd like to give a huge thank you to my family and friends for all your support. I want to thank David Zilz for all his advice and great ideas, Bassirou Seck for advice and defense practice, Hannah Benston for helping to keep me sane in our classes over the pandemic, and Destiny Rogers for always being there to provide crucial moral support.

CHAPTER 1. INTRODUCTION

A phononic crystal (PnC) is a device designed to manipulate phonons (quantum unit of acoustic waves) within a material or larger system. These devices are engineered to block or amplify propagating phonons within a specific frequency range. Though PnCs can be fabricated into a large variety of shapes and sizes, and can be made of many different materials, they typically consist of a slab (substrate) of a particular material embedded with periodic defects. The figure below shows a diagram of a simple PnC along with the accompanying band structure diagram.

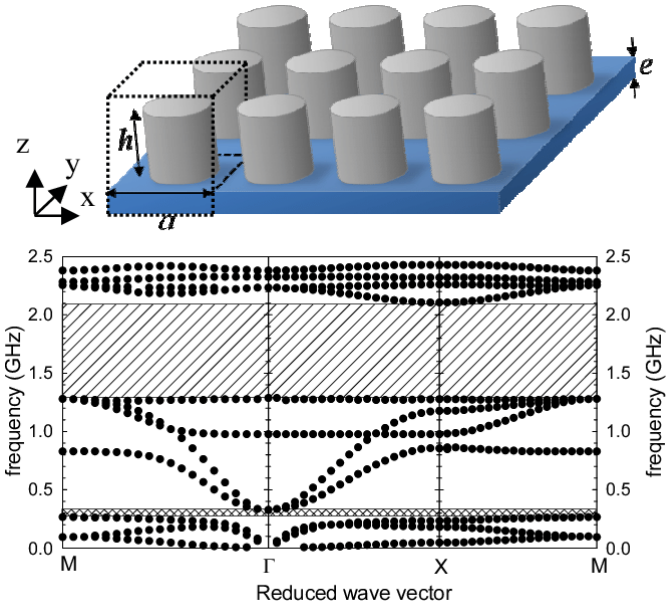


Figure 1: (above) An example diagram of a simple phononic crystal consisting of a material embedded with a periodic array of cylindrical pillars. (below) The associated frequency band structure diagram. The dashed lines indicate the region of frequency in which no phonons can propagate. [1]

1.1 Applications

As there are many systems in which it is desirable to manipulate the mechanical/acoustic waves in the system, PnCs have a wide array of applications. For example, on the larger scale, PnCs can be tuned to block resonant vibrations in aircraft, increasing part durability and reliability. PnCs can also be used to sound-treat large structures such as concert halls.

As heat consists of high frequency oscillations, PnCs can be designed to manipulate thermal transport within a system. The resulting applications include the ability to achieve more efficient cooling, harvesting of thermal energy that would otherwise go to waste, and achieving room temperature operating conditions for systems that normally need to be cooled to extremely low temperatures. For example, quantum computers currently require operating temperatures on the order of 1's of Kelvin. PnCs could one day allow for the ability to operate quantum computers or other quantum technologies with similar restraints at room temperature.

PnC applications extend to several other areas, such as the medical and biochemical fields. PnCs can be used to create biometric sensors tuned to detect molecules that vibrate at specific frequencies.

1.2 The Large PnC Parameter Space

Phononic crystals are fabricated with specific features in order to target bandgaps of specific frequency ranges. The center frequencies and bandgap sizes are determined by multiple factors. Some of these include substrate material, density, thickness, and height, among many others.

Along with intrinsic material properties, artificial defects/inclusions are engineered to manipulate the phononic band structure. These inclusions are typically periodic and can come in

the form of holes or structures protruding from the top of the material. Inclusion designs can range from simple structures such as circular holes/pillars to more complex designs such as the one shown in the figure below. The number of possible designs is only limited to the geometrical constraints of the fabrication process used to make the crystal.

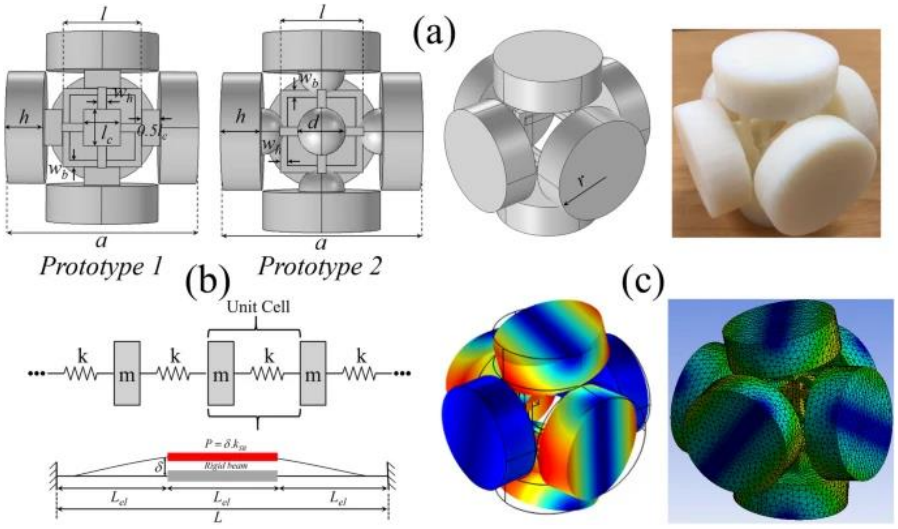


Figure 2: Example of an ultrawide bandgap 3-D phononic crystal design. Note in (a) there are multiple lengths (*h*, *w*, *lc*, etc.) that can be varied in the design. The high number of variable parameters results in a substantially large design space. [2]

Since there are many potential varying parameters when designing a phononic crystal, the potential design space can expand rapidly. Consider a design consisting of only 3 design parameters (for example, pillar height, pillar radius, and substrate height): if 10 different values for each parameter are considered then the size of the design space is 10 x 10 x 10 = 1000 combinations. Because the number of possible designs is essentially limitless, and number of combinations of varying parameters within a particular design is so large, the phononic crystal design space is largely under-explored.

1.3 Machine Learning Applied to PnC Design

Machine learning (ML) is a subset of artificial intelligence in which a computer is trained to analyze and predict patterns in a dataset. As access and storage capability for large amounts of data increases, so does the usefulness of machine learning in a variety of applications.

As mentioned in the previous section, the phononic crystal design space is large and under explored. This is in part because generating band structure information for many PnC designs is a very slow process, making it difficult to analyze the large number of configurations in a design space in quick succession.

ML helps speed up the exploration of the PnC design space by providing practically on-demand PnC parameter predictions. Once training data is generated, and an ML algorithm is trained using that data, the algorithm is able to interpolate information for any design located in parameter space between any two training data samples, provided the training data was sampled sufficiently.

1.4 Thesis Overview

Chapter 1 is an introduction to the various topics discussed in the paper. This includes a discussion on phononic crystals and their applications.

Chapter 2 reviews relevant background information. The basic theory behind phonons and phononic bandgaps is discussed, followed by an overview of machine learning with a focus on neural networks.

Chapter 3 discusses the methodology used in performing the neural network experiments. This includes the data generation, file input/output, data formatting, and neural network training/performance evaluation.

Chapter 4 contains descriptions of the datasets used as training data for the neural network experiments. For each dataset, the range of parameters used to generate the data are given. Along with the generation parameters, plots are shown to help visualize the various characteristics of each dataset.

Chapter 5 contains the neural network performance results. Various network configurations are attempted and performance metrics (described in Chapter 3) for those configurations are plotted. The results are laid out in the chapter in the following branching layout: (RPN/DPN/Other) → (Training data formatting method) → (Training dataset).

Chapter 6 will discuss conclusions and key takeaways from the machine learning experiments. There is also a section for future work that contains a number of ways the work done in this study could be expanded in the future.

The purpose of this study is to explore the potential in using machine learning to predict phononic band structure behavior of designs whose properties are not readily available. In addition, the information in this paper can act as a starting point for others who wish to conduct ML experiments using PnC training data. As specific techniques and code are not commonly documented in existing PnC ML studies, it can be difficult for others to replicate existing results. Because of this, code snippets, along with ML parameters used to achieve results are laid out in the Appendix section of this paper.

CHAPTER 2. BACKGROUND AND THEORY

2.1 Phonons: Monoatomic Linear Chain

A crystal is a highly ordered substance that consists of periodic arrays of atoms arranged in a geometric lattice. The atoms within the lattice constantly vibrate. These vibrations can be thought of as being mediated by springs connecting the atoms.

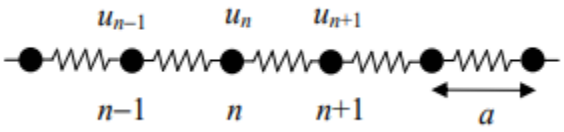


Figure 3: The physics governing the movement of atoms in a crystal lattice can be modeled as if the atoms were attached together with springs. [3]

In the case of a one-dimensional lattice, we can express the force on the nth atom as the sum of the forces contributed onto it by the adjacent atoms in the lattice:

$$M\frac{d^2u_n}{dt^2} = F_n = C(u_{n+1} - u_n) + C(u_{n-1} - u_n)$$

where M is the mass of the atom and C is the spring constant. Using the solution

$$u_n = Ae^{i(qx_n - \omega t)}$$

where ω and A are the frequency and amplitude of the wave respectively, and q is the wavevector (k is also commonly used for wavevector). Inserting this into the previous equation leads to:

$$M(-\omega^2)e^{iqna} = -C\left[2e^{iqna} - e^{iq(n+1)a} - e^{iq(n-1)a}\right]$$

$$M\omega^2 = C\left[2 - e^{iqa} - e^{-iqa}\right] = 2C(q - \cos(qa)) = 4C\sin^2\left(\frac{qa}{2}\right)$$

From this the dispersion relation, the relationship between wavevector and vibration frequency emerges:

$$\omega = \sqrt{\left(\frac{4C}{M}\right)}\left|\sin\left(\frac{qa}{2}\right)\right|$$

Plotting this relation, it becomes clear that q is periodic every 2pi/a. The region of independent q values between -pi/a and pi/a (length of 2pi) is known as the first Brillouin zone.

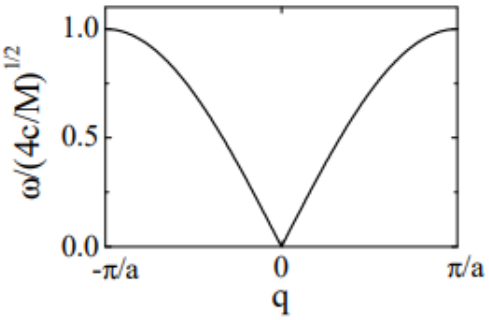


Figure 4: Phonon dispersion relation. [3]

Tysymbal [3] explains in a breakdown of this theory that when expanding this concept to a diatomic chain (containing two different types of atoms each with different mass), a new mode emerges in the dispersion relationship, an optical mode:

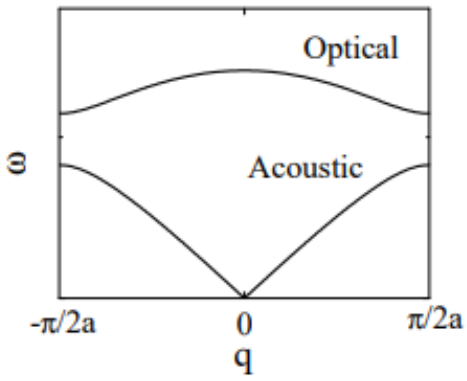


Figure 5: An optical mode arises in a diatomic chain of atoms. [3]

As the complexity and degrees of freedom available in the unit cell increase, so do the number of modes in the dispersion relationship. This forms the basis for the band structure diagrams seen throughout this study. For more information and a full derivation of this information, see [2].

2.2 Phononic Bandgaps

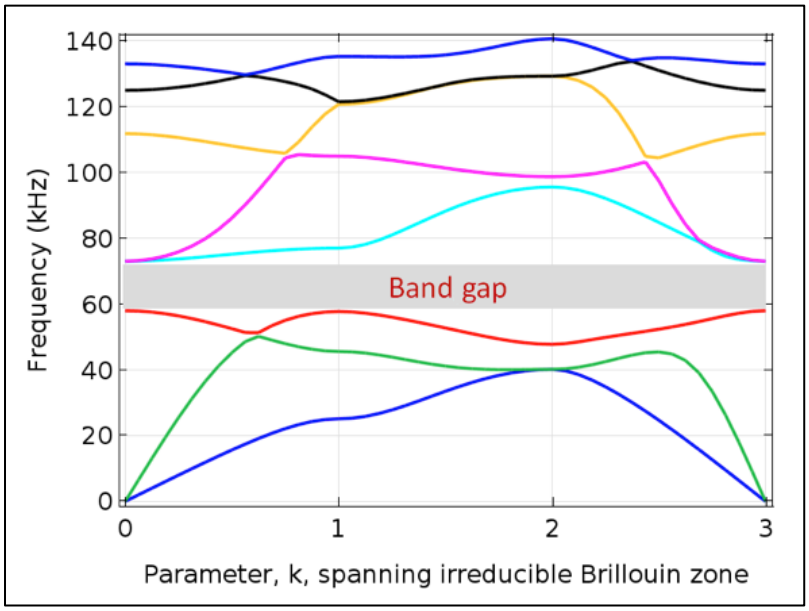


Figure 6: The type of diagram used to visualize what frequencies are allowed for a particular material is a phononic band diagram. The x-axis value of a band diagram is the value in k-space, and the y-axis value is the allowed frequency. [4]

k-space (k is the same as q from the previous section), also known as reciprocal or momentum space, is a transformation of a crystal lattice in which each point in the reciprocal lattice represents a different plane in the original lattice. The figure below shows an example transformation between real and k-space. Since every straight line in which a particle could travel through a material can be represented by plane, analyzing the lattice in k-space gives information about what particle properties are allowed in the various directions of propagation inside a

material. Since phonon dispersion is periodic in reciprocal space, only a section reciprocal space needs to be sampled in order to capture every possible direction in which a phonon can flow in a material. As mentioned in the previous section, this area of reciprocal space is known as the first Bernoulli zone.

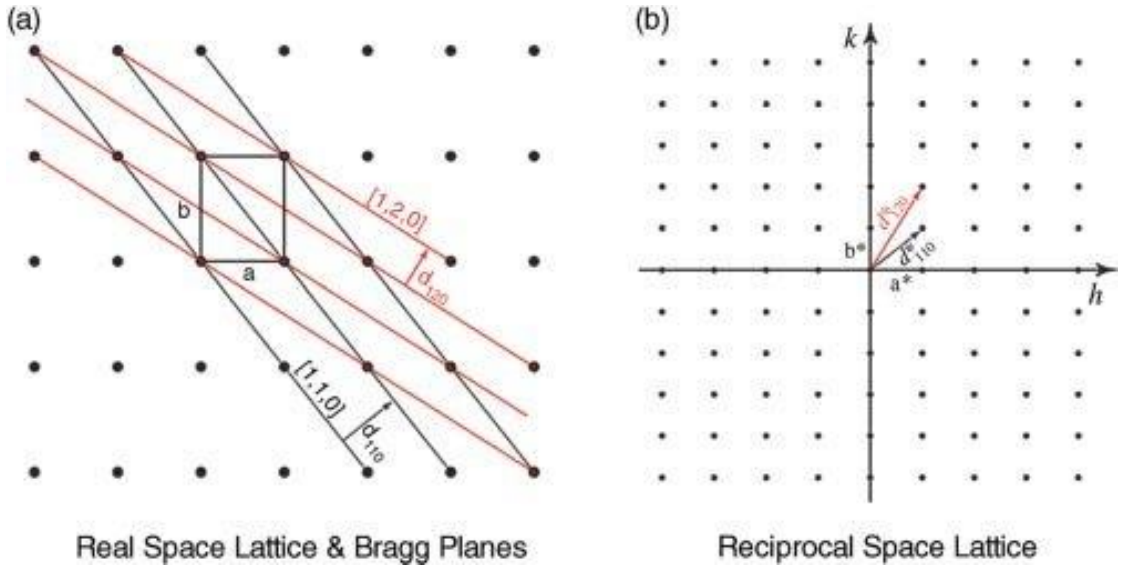


Figure 7: Diagram of real vs reciprocal space lattices. [5]

If a range of frequency values is not present in any eigenmode for any point in k-space, then a phononic bandgap is present within a material. This means that phonons or vibrations at that frequency range can propagate in any direction inside the material. This happens due to an effect called Bragg scattering, which produces destructively interfering waves within the material that cause complete bandgaps across k-space. Figure 6 shows a band structure representing a crystal in which no phonons can propagate within the approximately 60 kHz to 75 kHz range.

2.3 Elastic Continuum Theory

Elastic continuum theory (ECT) is the modeling of discrete structures as a continuous medium. Though crystals are discrete in nature (made of atoms), the phononic or other properties can be approximated through ECT.

We can write the equation of motion in tensor form under ECT:

$$\rho \ddot{u}_i = \sigma_{ij}$$

where ρ is density, u is the elastic displacement vector, and σ is stress. We define stress and strain (Hooke's Law) as such:

$$\sigma_{ij} = C_{ijkl} e_{kl}$$

$$e_{ij} = S_{ijkl}\, \sigma_{kl}$$

where C is stiffness and S is compliance. Stiffness is defined as the ability of a material to resist deformation while compliance is its inverse. Taking silicon as an example, its cubic arrangement allows the stiffness matrix to be reduced to terms, c_{11}, c_{12}, and c_{44}:

$$\begin{bmatrix} c_{11} & c_{12} & c_{12} & 0 & 0 & 0 \\ c_{12} & c_{11} & c_{12} & 0 & 0 & 0 \\ c_{12} & c_{12} & c_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{44} \end{bmatrix}$$

A solution of the equation of motion can for a wave moving in the x-axis can be written as:

$$u = A \exp\left[i\,(kx - \omega t)\right]$$

where A is amplitude and ω is angular frequency. The angular frequencies for a longitudinal and transverse wave are as such:

$$\omega_L^2 = \frac{c_{11}k^2}{\rho}$$

$$\omega_T^2 = \frac{c_{44}k^2}{\rho}$$

ECT is an approximation for large scale effects, and as such does not scale to all situations. ECT should be applied for elastic waves of low frequency or events taking place over long time scales. For problems relating to small lengths or small time scales, atomistic effects cannot be neglected and ECT begins to fail [6]. More information and examples of ECT can be found in [7] and [8].

2.4 Machine Learning

As we look to computers to solve increasingly complex problems, the interest in finding methods for computers to learn and solve problems on their own has increased too. Machine learning (ML) is one answer to this problem. ML is a subset of artificial intelligence in which a computer is trained so solve a specific problem on its own. Through a machine learning algorithm, a computer is able to generate extremely complex models that would be near-impossible for a human to explicitly program.

ML algorithms typically benefit from having large amounts of data to train on. As the computational power of computers grows over time, the ability to quickly generate large datasets increases. Easy access to data is one reason that ML algorithms have become popular in the modern era.

There are many types of ML algorithms, including “random forest,” “support vector machines,” and “k-means.” These and other ML algorithm types fall into one of two categories: supervised learning (the algorithm is trained with data in which a known answer is given) or unsupervised learning (the algorithm finds patterns in the data without being given the answer). Though many different types of ML algorithms could be applied to the topic of this study, the focus here was on a type of supervised learning known as a neural network.

2.4.1 Neural Networks

A neural network is an ML algorithm that mimics the biological methods that brains use to learn. Like a brain, it is composed of layers of varying numbers of nodes (analogous to neurons). Through a training process, the neural network finds patterns in data, and the weights and biases associated with these nodes are manipulated. Through training a neural network, we can arrive at complex input/output models which would have been extremely difficult to derive otherwise.

The figure below shows a diagram of a typical neural network. The architecture consists of an input layer, a number of hidden layers, followed by an output layer. Information flows in this order through the multiple connected nodes found in each layer.

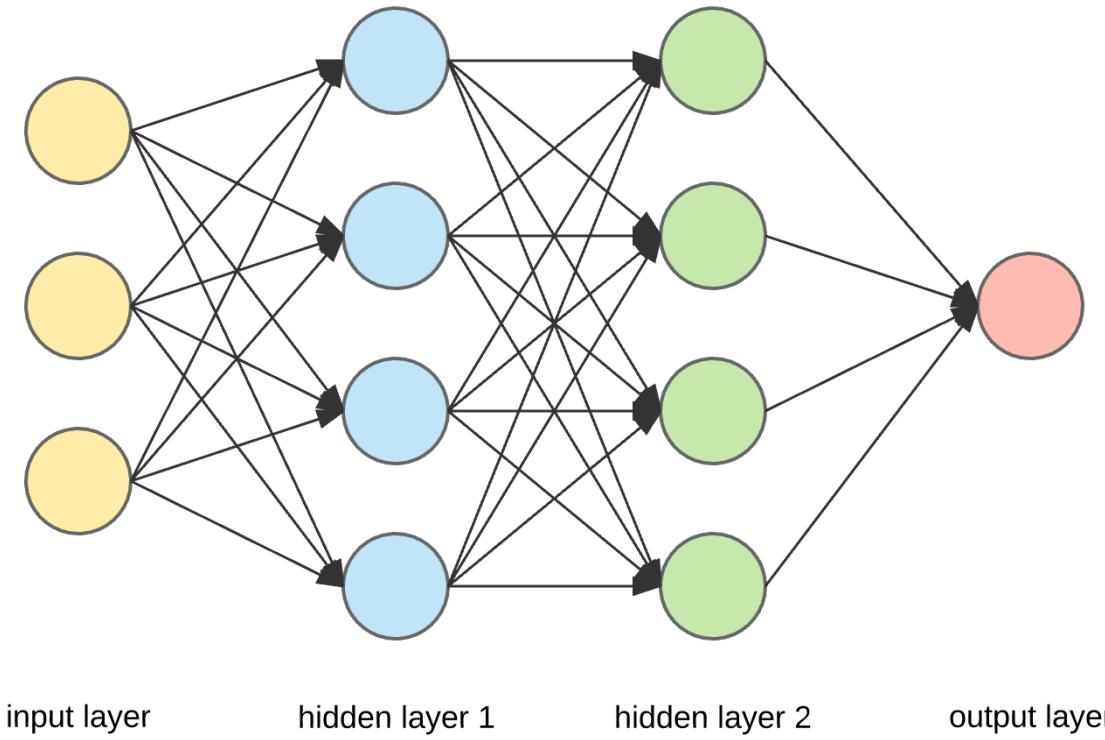


Figure 8: Example neural network architecture composed of an input layer, hidden layers, and an output layer. [9]

The input and output layers hold the input and output of the system, as the names suggest. The input layer contains the values that the user has supplied. For example, one common input to a neural network is image data for image classification. In this case, each node in the input layer corresponds to the intensity value of one pixel in an image. The number of nodes in the input layer is as many as are needed to encompass the amount of possible input values; if an image contains 256 pixels, then 256 input layer nodes are needed. The output layer would then contain an output node corresponding to each of the possible classification decisions.

The hidden layer nodes take as input values from the previous layer (input layer or other hidden layer) and manipulates them through a series of weights, biases, and activation functions. The output value of a hidden layer node is as follows:

$$\sigma(\sum_{i=1}^{n} w_i a_i + b)$$

which is a sum of the values from each of the n nodes from the previous layer a_i, multiplied by some weight w_i. A bias b is added to the sum before the sum is passed to an activation function σ. The figure below illustrates an example of a single hidden layer node with three inputs from the previous layer in the neural network.

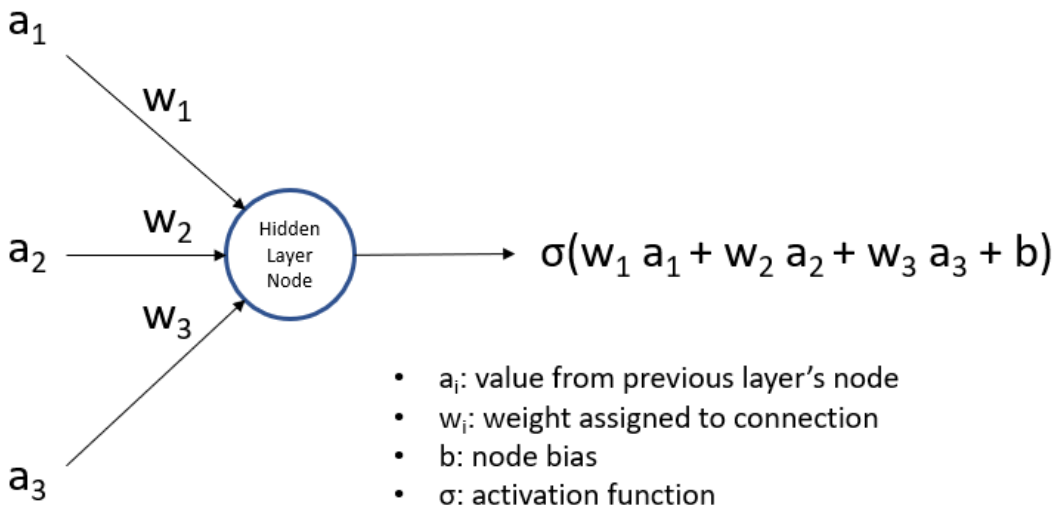


Figure 9: Input and output parameters for a single hidden layer node.

The main purpose of the activation function is to introduce non-linearity into the model, as many problems are complex and require non-linear solutions. Each activation function has strengths in different applications. Some activation functions may be a better choice for use in classification problems, or problems in which an input is categorized into one of a set of discrete options (e.g., deciding if an image is that of a cat or a dog), while some activation functions may be a better choice for regression problems, or problems in which a continuous value is desired.

(e.g., predicting the value of a stock a week from today). The figure below shows examples of commonly used activation functions.

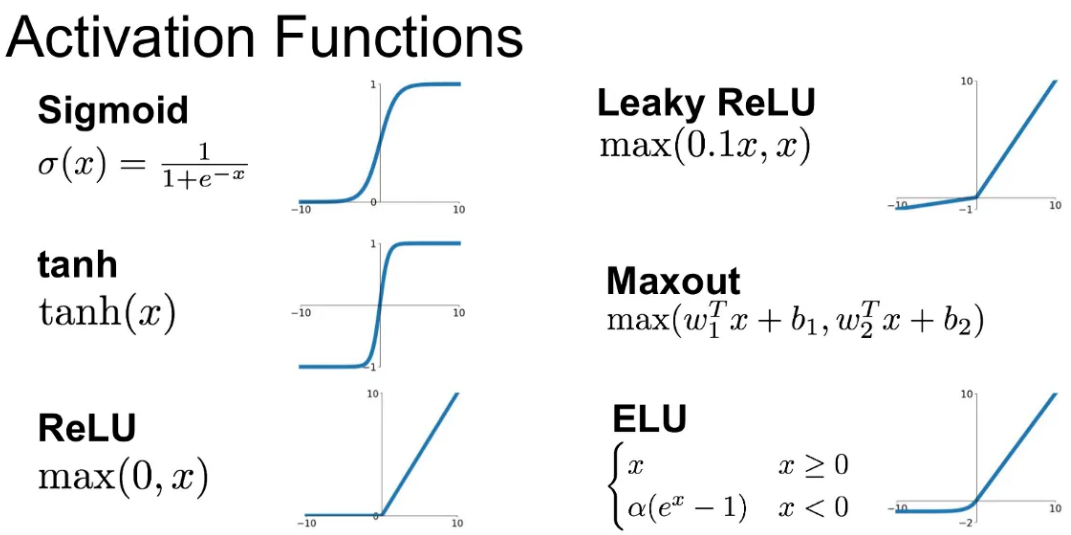


Figure 10: Commonly used activation functions. [10]

2.4.2 Training and Back-propagation

The values for the weights and biases the neural network initially uses are picked randomly. During the training process, the neural network uses a pre-generated dataset (training data) containing correct inputs and outputs to update these values to be more and more optimal for the given problem. This is done through a process called back-propagation.

The neural network's goal when training is to minimize a cost function which represents how well the neural network is performing (i.e., how well the chosen weights and biases work to fit the desired solution). One way the cost function is commonly calculated is through the mean squared error (MSE) between the output of the neural network and the true values.

The process in which the neural network minimizes its cost function is called gradient descent. The cost function has some global minimum that the neural network attempts to reach by

adjusting its weights and biases. Starting at the output layer and working backwards, the neural network will calculate the partial derivatives of the cost function with respect to the parameters of the previous layer and update the parameters in a way minimizes the cost function. This is repeated for each layer. This process is visualized in the figure below using a plotted cost function.

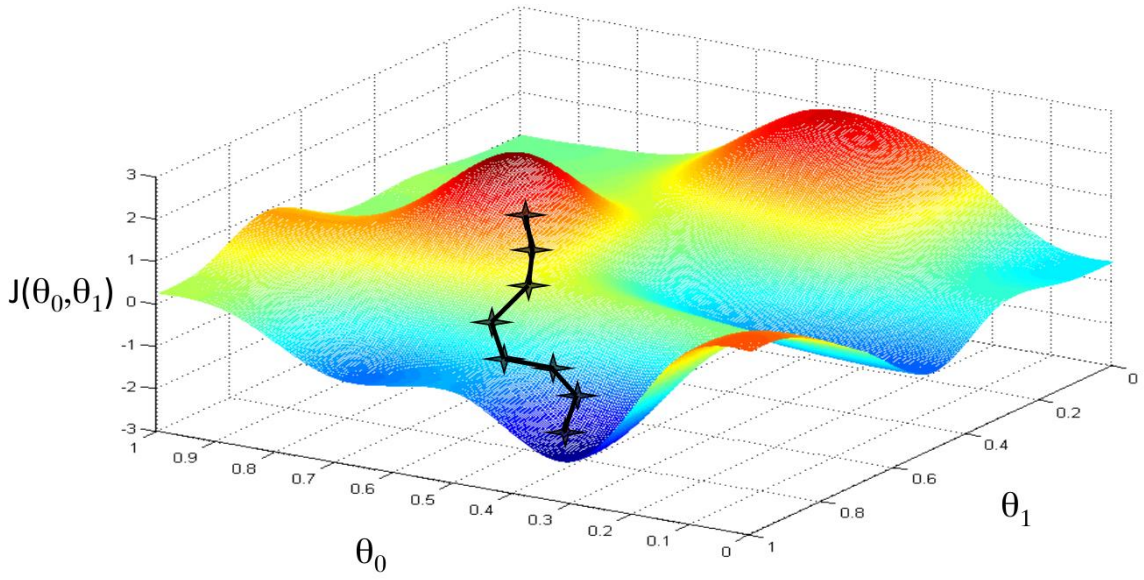


Figure 11: The process of gradient descent works by following the path of the cost function that leads to its minimum values. The value of the cost function starts high but works its way to low values. [11]

Once back-propagation is complete, the process starts again using new samples from the training dataset. If the training process was successful, the neural network will have minimized the cost function to some global minimum. Sometimes though, neural networks will get stuck in local minima of the cost function (see dark blue areas that aren't the global minimum in the previous figure). There are however a variety of parameters that can be adjusted to help prevent this.

2.4.3 Other Neural Network Categories

There are many types of neural networks, including “recurrent neural networks” (RNN), “convolutional neural networks” (CNN), and “generative adversarial networks” (GAN). Each of these network types have strengths in particular applications. RNN’s for example are commonly used for speech recognition applications. Another interesting application is found in a GAN’s ability to produce fake images by having two neural networks compete. See https://thesecatsdonotexist.com/ for a gallery of fake images of cats generated by a neural network.

CHAPTER 3. METHODS

This chapter describes the various methods used for conducting the experiments listed in Chapter 5. This includes the software used and a discussion of the various representations used for training data formatting.

As libraries made specifically for processing phononics data using machine learning tools aren't widely available, a custom package was made for this purpose. The package acts as a training data formatter and test harness for conducting different types of machine learning experiments using the custom phononics data that was generated specifically for this project.

3.1 Libraries and Software Used

The ability to use ML to predict novel crystal designs has already been demonstrated in AFRL (Air Force Research Laboratories), who funded this study. An ML platform, MANTIS (Machine Accelerated Nanoscale Targeted Inhomogenous Structures), had been developed to conduct machine learning studies on photonic crystals. The MANTIS package contains examples of multiple different neural network types and tutorials. As such, that package was heavily referenced when writing neural networks for the phononics studies in this paper. Many of the decisions for which libraries/APIs to use were made because MANTIS was already leveraging them and expertise from the team that developed it was available.

The programming language used for the experiments in this study was Python. Python is an interpreted language that contains a large set of open-source libraries for a variety of purposes, many of them well-suited for scientific or data-analytical studies. Most of the other packages used were included in Anaconda, which is a collection containing Python and other commonly used

packages. When starting with Python for the first time or trying to recreate the studies done in this paper, installing Anaconda is a good starting point.

Python also has multiple machine learning packages available. The library used for constructing neural networks in this study was Keras. Keras is an interface for another Python package, TensorFlow, which is a Google-owned machine learning API. The purpose of Keras is to create a simple user interface for accessing TensorFlow capabilities. Programming a neural network from scratch is an involved process and these libraries abstract much of the low-level math/processes while still allowing for enough freedom to create custom neural networks for specific applications.

3.2 Data Generation

In order to train the neural networks, phononic band structure data needed to be generated. This was done using a C++ package built by a *Worcester Polytechnic Institute* PhD student, Debanik Das, collaborating with AFRL. This package is able simulate phononic structures and produce band structure information. Given a parameter range and step size, the package can sweep over the parameter space and generate band structure information for multiple designs.

This package has been verified, using COMSOL simulations, to produce accurate results. The major achievement demonstrated by this package is that its data generation method has shown to produce results as much as twice as fast as COMSOL. This is convenient because generating phononics data is generally a slow process. While the training data could have been generated with COMSOL to nearly the same results, the software was chosen as a more time-efficient alternative.

The method used to generate the training data is the reduced Bloch mode expansion. The reduced Bloch mode expansion (RBME), proposed by Hussein in [12], is a technique for

generating band structures that is computationally less expensive than the normal Bloch mode expansion technique.

Das defines the action integral corresponding to the motion of displacement in a crystal:

$$A = \frac{1}{2} U^{\dagger} \left[\int [N(r)]^T \overleftarrow{D}(q) C \overrightarrow{D}(q) [N(r)] dV - \omega^2 \int \rho(r) [N(r)]^T [N(r)] \right] U$$

where U is the Bloch function, ω is the eigenfrequency, C is the stiffness tensor, $D(q)$ are wave vector q-dependent first order differential operators, and $N(r)$ is the shape function defined in [12]. The action function is integrated over the discretized material geometry through finite element methods. The discretization of the geometry occurs through a process called meshing, in which the unit cell is divided into smaller sub-geometries. In the case of a phononic crystal, defects tend to cover large areas of physical space and therefore necessitate the choice of large unit cell.

Wavevector q is varied over points reciprocal space within the irreducible Bernoulli zone and eigenfrequencies are calculated at each. The RBME carefully selects points of high symmetry within reciprocal space and intermediate points in between. This process greatly reducing the amount of computation compared to the normal Bloch mode expansion. The figure below visualizes the points of high symmetry in reciprocal space.

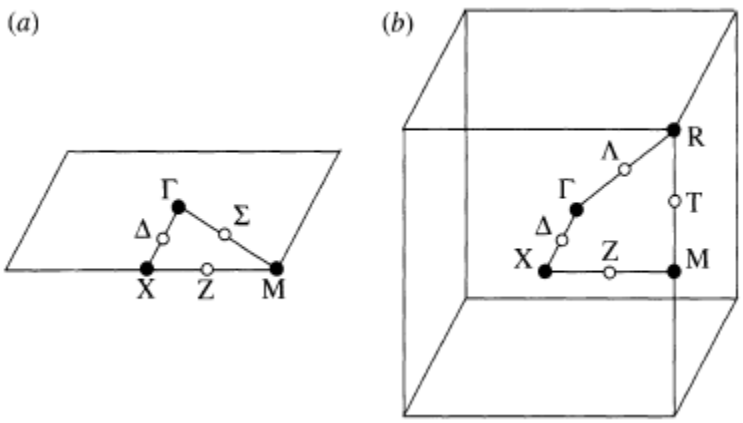


Figure 12: Selected points of high symmetry (black) and intermediate points (hollow) in reciprocal space. (a) is shows a 2D reciprocal representation while (b) shows the equivalent in 3D. [12]

The figure below shows an example of a band structure generated through this method overlayed with a band structure generated from COMSOL.

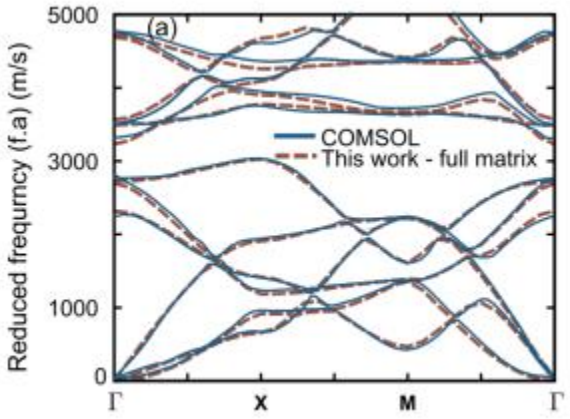
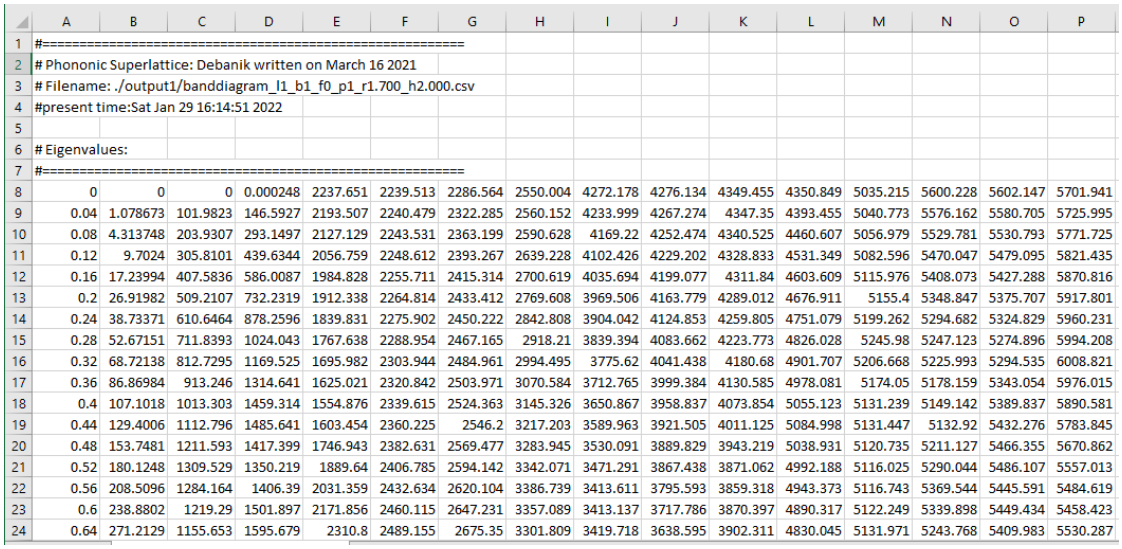


Figure 13: Example band diagram generated through Das's RBME band structure generation tool.

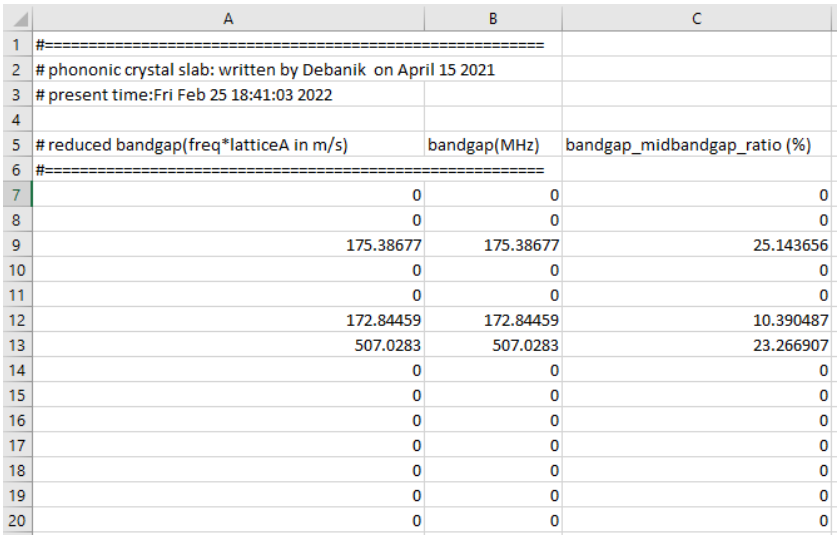
The program generates three data files for each simulated crystal design: a JSON (JavaScript Object Notation) formatted file containing the design/geometry parameters (lengths, materials used, etc., a CSV (comma-separated values) formatted file containing a set of arrays

representing the phononic band structure, and a CSV formatted file containing bandgap information. Examples of each of these files are shown in the figures below.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	#==========															
2	# Phononic Superlattice: Debanik written on March 16 2021															
3	# Filename: ./output1/banddiagram_l1_b1_f0_p1_r1.700_h2.000.csv															
4	#present time:Sat Jan 29 16:14:51 2022															
5																
6	# Eigenvalues:															
7	#==========															
8	0	0	0	0.000248	2237.651	2239.513	2286.564	2550.004	4272.178	4276.134	4349.455	4350.849	5035.215	5600.228	5602.147	5701.941
9	0.04	1.078673	101.9823	146.5927	2193.507	2240.479	2322.285	2560.152	4233.999	4267.274	4347.35	4393.455	5040.773	5576.162	5580.705	5725.995
10	0.08	4.313748	203.9307	293.1497	2127.129	2243.531	2363.199	2590.628	4169.22	4252.474	4340.525	4460.607	5056.979	5529.781	5530.793	5771.725
11	0.12	9.7024	305.8101	439.6344	2056.759	2248.612	2393.267	2639.228	4102.426	4229.202	4328.833	4531.349	5082.596	5470.047	5479.095	5821.435
12	0.16	17.23994	407.5836	586.0087	1984.828	2255.711	2415.314	2700.619	4035.694	4199.077	4311.84	4603.609	5115.976	5408.073	5427.288	5870.816
13	0.2	26.91982	509.2107	732.2319	1912.338	2264.814	2433.412	2769.608	3969.506	4163.779	4289.012	4676.911	5155.4	5348.847	5375.707	5917.801
14	0.24	38.73371	610.6464	878.2596	1839.831	2275.902	2450.222	2842.808	3904.042	4124.853	4259.805	4751.079	5199.262	5294.682	5324.829	5960.231
15	0.28	52.67151	711.8393	1024.043	1767.638	2288.954	2467.165	2918.21	3839.394	4083.662	4223.773	4826.028	5245.98	5247.123	5274.896	5994.208
16	0.32	68.72138	812.7295	1169.525	1695.982	2303.944	2484.961	2994.495	3775.62	4041.438	4180.68	4901.707	5206.668	5225.993	5294.535	6008.821
17	0.36	86.86984	913.246	1314.641	1625.021	2320.842	2503.971	3070.584	3712.765	3999.384	4130.585	4978.081	5174.05	5178.159	5343.054	5976.015
18	0.4	107.1018	1013.303	1459.314	1554.876	2339.615	2524.363	3145.326	3650.867	3958.837	4073.854	5055.123	5131.239	5149.142	5389.837	5890.581
19	0.44	129.4006	1112.796	1485.641	1603.454	2360.225	2546.2	3217.203	3589.963	3921.505	4011.125	5084.998	5131.447	5132.92	5432.276	5783.845
20	0.48	153.7481	1211.593	1417.399	1746.943	2382.631	2569.477	3283.945	3530.091	3889.829	3943.219	5038.931	5120.735	5211.127	5466.355	5670.862
21	0.52	180.1248	1309.529	1350.219	1889.64	2406.785	2594.142	3342.071	3471.291	3867.438	3871.062	4992.188	5116.025	5290.044	5486.107	5557.013
22	0.56	208.5096	1284.164	1406.39	2031.359	2432.634	2620.104	3386.739	3413.611	3795.593	3859.318	4943.373	5116.743	5369.544	5445.591	5484.619
23	0.6	238.8802	1219.29	1501.897	2171.856	2460.115	2647.231	3357.089	3413.137	3717.786	3870.397	4890.317	5122.249	5339.898	5449.434	5458.423
24	0.64	271.2129	1155.653	1595.679	2310.8	2489.155	2675.35	3301.809	3419.718	3638.595	3902.311	4830.045	5131.971	5243.768	5409.983	5530.287

Figure 14: CSV formatted band structure data where each column represents a unique array/eigenmode in the band structure. One of the three outputs of the data generation package.



	A	B	C
1	#==========		
2	# phononic crystal slab: written by Debanik on April 15 2021		
3	# present time:Fri Feb 25 18:41:03 2022		
4			
5	# reduced bandgap(freq*latticeA in m/s)	bandgap(MHz)	bandgap_midbandgap_ratio (%)
6	#==========		
7	0	0	0
8	0	0	0
9	175.38677	175.38677	25.143656
10	0	0	0
11	0	0	0
12	172.84459	172.84459	10.390487
13	507.0283	507.0283	23.266907
14	0	0	0
15	0	0	0
16	0	0	0
17	0	0	0
18	0	0	0
19	0	0	0
20	0	0	0

Figure 15: CSV formatted band gap data where each row represents the bandgap data, if present (non-zero) between two eigenmodes. Number of rows is always equal to number of eigenmodes minus one. One of the three outputs of the data generation package.

Figure 16: JSON formatted design geometry data. One of the three outputs of the data generation package.

Details on the exact training data generated for this study can be found in Chapter 4. This includes the exact geometry values that were swept over along with plots showing the band structure content in each dataset.

3.3 Data Formatting

The training data is first ingested by several python scripts that format the data in a way appropriate for neural net training. Each piece of data is encoded as a row in a 2-D array, with each column representing a different data parameter. For discrete parameters such as material, a one-hot encoding structure is used. In this case, each possible value in a particular category is assigned a binary digit, with one assigned a '1' and the others assigned '0.' For example, when encoding substrate material of type Si into the array with 3 total choices: Si, SiC, and W, three bits are used. In this case, the bits corresponding to SiC and W contain 0 and the bit corresponding to Si contains 1. An example is shown below.

Index	Radius	BaseHeight	PillarHeight	Si	SiC	W
1541	2.6	3.2	0	1	0	0
1592	2.63	3.2	0	0	0	1
1796	2.75	3.2	0	0	1	0

Figure 17: Visual representation of how training data is formatted before being given to a neural network, where each row represents a different PnC design. The first 3 non-index columns, "Radius," "BaseHeight," and "PillarHeight" are continuous values representing PnC geometry with units of 1E-7m (1/10 microns). The last 3 columns, "Si," "SiC," and "W" encode the substrate material. This format is scalable with more feature columns if needed.

Then, for each of the three categories of data (geometry, band gap info, band structure info), the information from the individual design files is compiled into three files. It was found that saving the training data in this format made running later tests faster, as it is quicker to load in the training data in this format, which is ready to be used to train the neural network than to have to load and reformat the training data. It also removes the need to parse potentially thousands of files, a computationally expensive process compared to only needing to access three.

The data is then split into two batches: the training set and testing set. The training set is what the neural net uses to learn and iterate over multiple epochs. The test set is only used after the neural net is done training to evaluate its performance and does not influence the training process. For these experiments, 10% of the data was randomly held back for testing.

Once the data is split, all parameters are then scaled between 0 and 1. This is known as "min-max scaling" and is done because the ranges and means of different parameters can differ by high orders of magnitude. Scaling prevents a single feature from dominating the others during training. It is a common misconception that this scaling should happen before the training/testing split. The scaling should happen after so that the training set has no influence on the testing set.

3.4 Choices for Representing Phononics Training Data

When designing a neural network, it is important to consider how best to represent the information the network will train from. There were multiple methods conceived for representing the phononic bandgap information each with its own advantages and challenges.

This section will include discussions on how practical each method is for predicting phononics data. Though some of the methods may successfully train the neural network, it is important to remember that after the network is trained, the data given by the user will need to be sent to the trained network in the same format it was trained in. For example, as will be mentioned

in the section on the “full band structure” method, it is impractical for the user to be expected to supply a full band structure diagram as input.

For the photonics problem previously explored by the MANTIS team, reflectance and transmission spectra were used to train a neural network. For the phononics problem, there are multiple choices: bandgap size with center frequency, band structure diagram, or density of states diagram.

3.4.1 Largest Bandgap Approach

For this method, each design in the training data is represented with the bandgap size and center frequency of its largest bandgap. The choice to use the largest bandgap was arbitrary as the smallest or a randomly picked bandgap could have also been used.

Compared to the other methods described in this section, training on bandgap size and center frequency alone is the most favorable approach from an ease-of-use perspective. For an application that predicts a crystal design, supplying a desired bandgap and center frequency is straightforward and intuitive for the user. Training also takes the least amount of time for this approach, as it uses the smallest amount of data (i.e., sets of two floating point numbers).

The figure below shows the neural network architecture for implementing this method in a neural network. The input layer receives the desired bandgap and center frequency, and the output layer predicts geometry parameters.

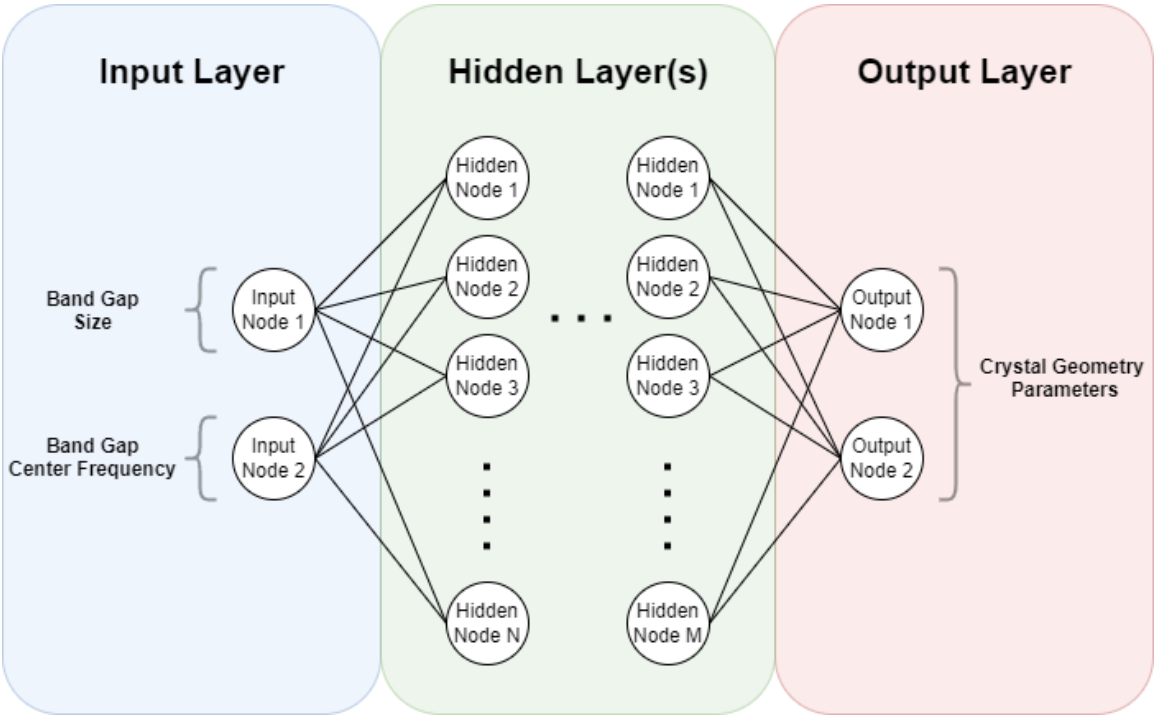


Figure 18: DPN (design predicting network) architecture diagram for bandgap, center frequency approach. Number of hidden layers and number of nodes per hidden layer can vary. Output layer is representative of the datasets containing circular holes, as there are two variable geometry parameters in those datasets (See Chapter 4).

3.4.2 Full Band Diagram Approach

Another approach is to train the neural network using the entire phononic band structure. Training on band diagrams comes with the advantage of having the most amount of data for the algorithm to learn from. The biggest issue with this approach comes when attempting to use it to predict crystal designs; it is almost completely unviable for the user to supply a novel band structure diagram without already having simulated a crystal design.

When using band structures for training data, the input layer must be formatted in such a way that every sample of each eigenmode array can be captured by a single neuron. The figure

below shows the DPN architecture for the full band diagram approach. The input layer receives the band structure values, flattened to a 1248 sized 1-D array, and the output layer contains predictions for two geometry values.

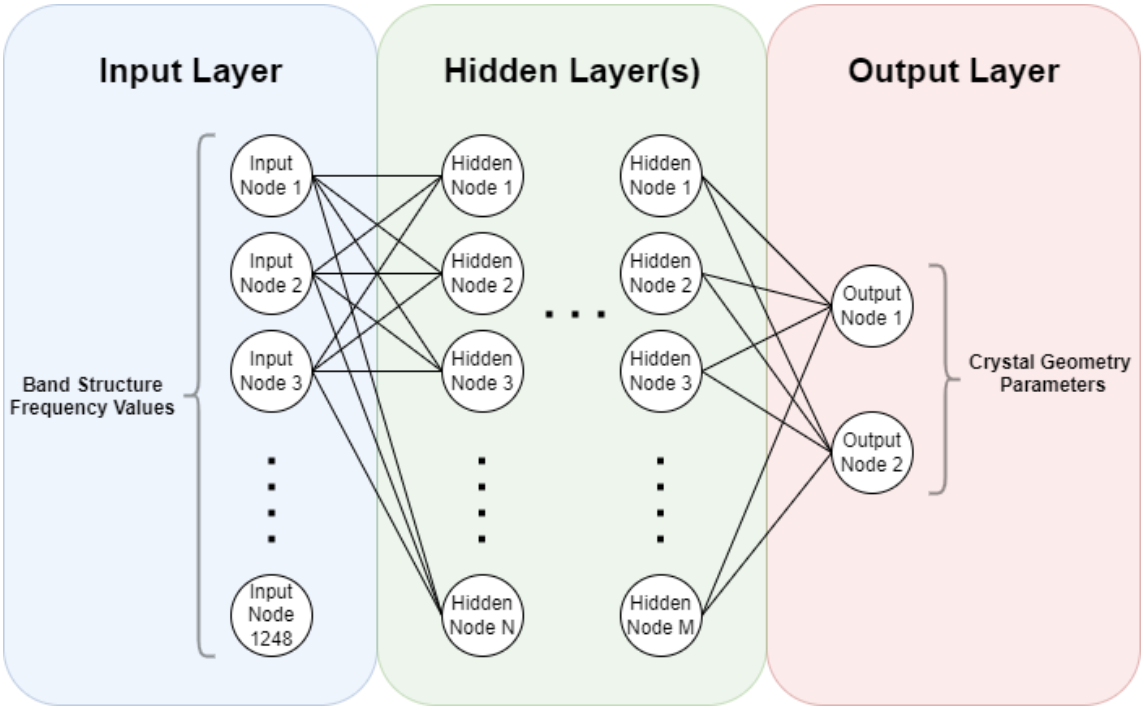


Figure 19: DPN (design predicting network) architecture diagram for band structure approach. The input layer represents the 16 eigenmode arrays with 78 samples per array totaling in 1248 values. Number of hidden layers and number of nodes per hidden layer can vary. Output layer is representative of the datasets containing circular holes, as there are two variable geometry parameters in those datasets (See Chapter 4).

3.4.3 Density of States Approach

A third approach is to use a density of states (DOS) representation of the phononic band structure. This is plotted as a one-dimensional line, where the amplitude is proportional to the number of eigenstates present in a particular “slice” of frequency space. If there are no eigenstates

in a particular frequency region of the band structure diagram (a bandgap), then the density of states will equal zero in those regions. An example phononic band structure diagram and equivalent density of states diagram are shown in the figure below.

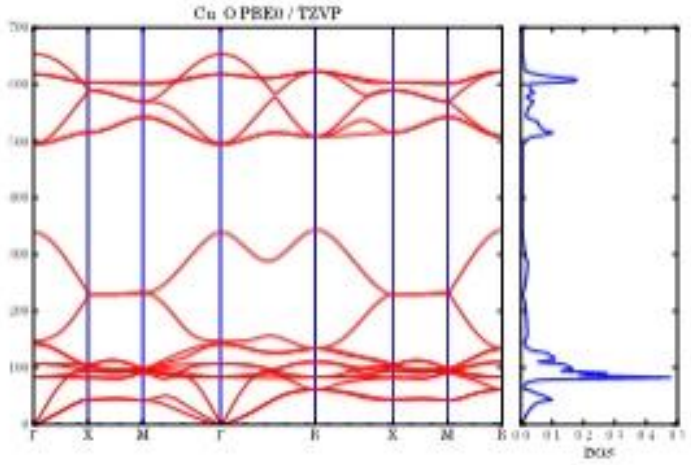


Figure 20: Example density of states diagram (right) calculated from band structure (left). Peaks in the DOS occur in the frequency ranges where multiple eigenmodes overlap, and zeros occur in the frequency ranges corresponding to bandgaps. [13]

The practicality associated with using DOS to represent PnC training data is between that of the "largest bandgap" and "full band structure" approaches. DOS is a one-dimensional line, so it is easier for the user to supply than a full band structure, but not as easy as supplying a simple band gap and center frequency pair. Zeros are used in the regions where the user wants to represent a bandgap but figuring out the best way to "fill" the non-bandgap regions requires careful consideration.

To help remedy this, two types of DOS representations are explored: scalar and binary. The scalar representation is taken as a histogram over the frequency axis of the corresponding band diagram. The resulting DOS array is formed from the bin counts of the histogram. The binary

representation is obtained by first finding the scalar DOS, then changing each value greater than zero to a one. The advantage of the binary representation is that it is easy for the user to construct. I.e., zeroes are placed in the bandgap region while ones are place elsewhere. The figure below shows both representations next to the same corresponding band structure diagram. For the ML studies conducted in Chapter 5, 100 bins was chosen, but a different number can be chosen.

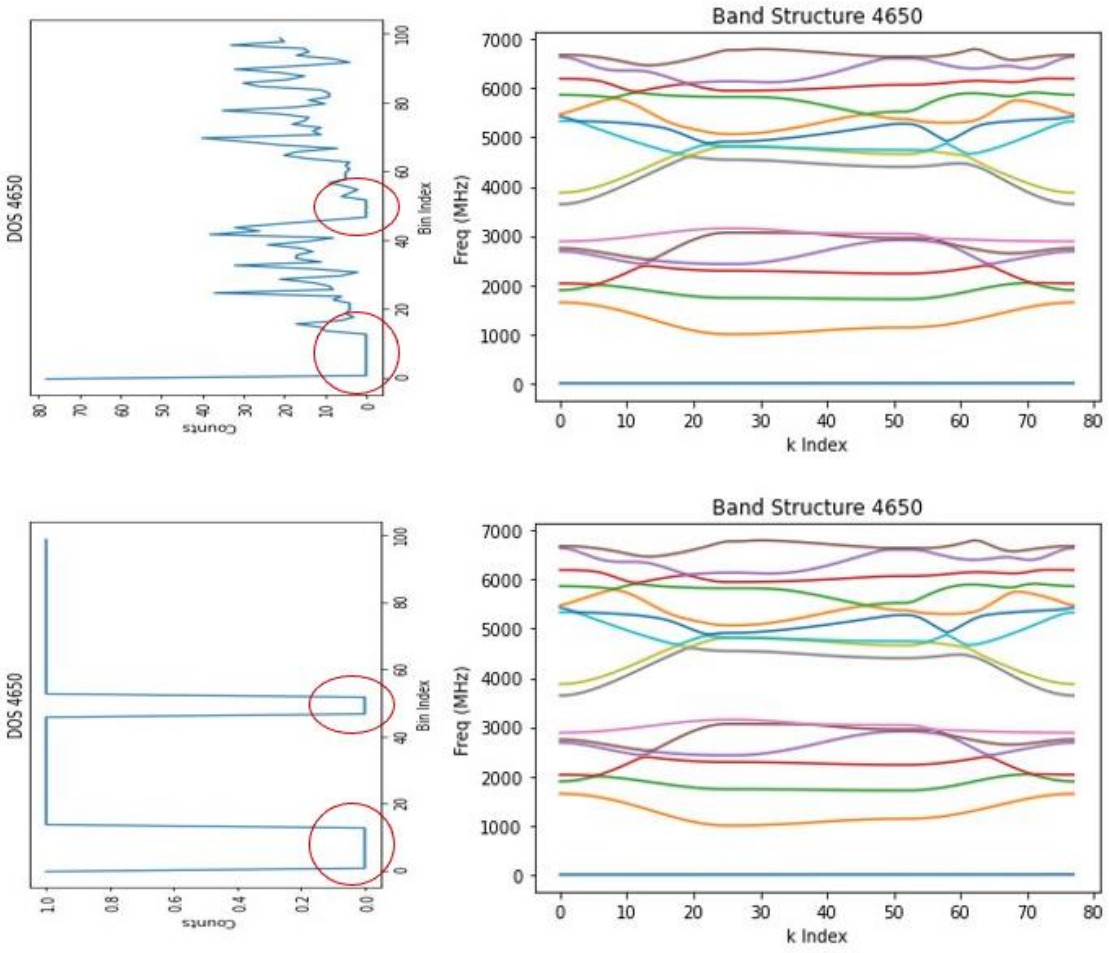


Figure 21: Scalar representation (top) and binary (bottom) representation of density of states. Diagrams are aligned such that bandgap regions in DOS and corresponding band diagrams align horizontally (circled in red on DOS diagrams).

3.5 Neural Network Architectures

The main type of architecture explored was a multilayer perceptron, or MLP. This is one of, if not the most basic form of neural networks. It consists of only an input layer, densely connected (each node is connected to all nodes from the previous layer) hidden layers, and an output layer.

A network can be constructed to predict band structure properties. This will be referred to as a response predicting network (RPN), referred to as the "forward" problem. A network that predicts crystal geometry from band structure is known as a design predicting network (DPN), referred to as the "inverse" problem. The data is separated by input and output, corresponding to the type of neural network structure. For an RPN, geometry metadata is the input, and band structure information is the output. For a DPN, band structure information is the input, and geometry metadata is the output.

3.6 Neural Network Parameter Choices

There are many parameters to choose from when creating a neural network. These include number of layers, number of nodes per layer, activation functions per layer, the addition of other layer types such as "dropout" or "batch normalization" layers, learning rate, optimizer, among others.

There is a common sentiment that it is difficult to know exactly what choice of parameter values are best for a specific application, and that one of the best methods is to research and adopt the parameter choices others have used for similar problems. Unless unknown to the author, there does not appear to be any cataloged parameter choices for training a neural network with phononics data.

The parameter values used for each experiment are listed in the Appendix section of this paper. In general, if a set of parameters resulted in favorable results, or were at least representative of the best possible results observed trying multiple different configurations, they were used and cataloged in the Appendix. This is not to say that there are not more optimal or less redundant choices of parameters that exist. In fact, pursuing optimal tuning parameters is an in-depth and time-consuming effort, so it is almost certain that there are better choices for parameter values. Though tuning these parameters typically only result in slight optimizations.

3.7 Neural Network Training

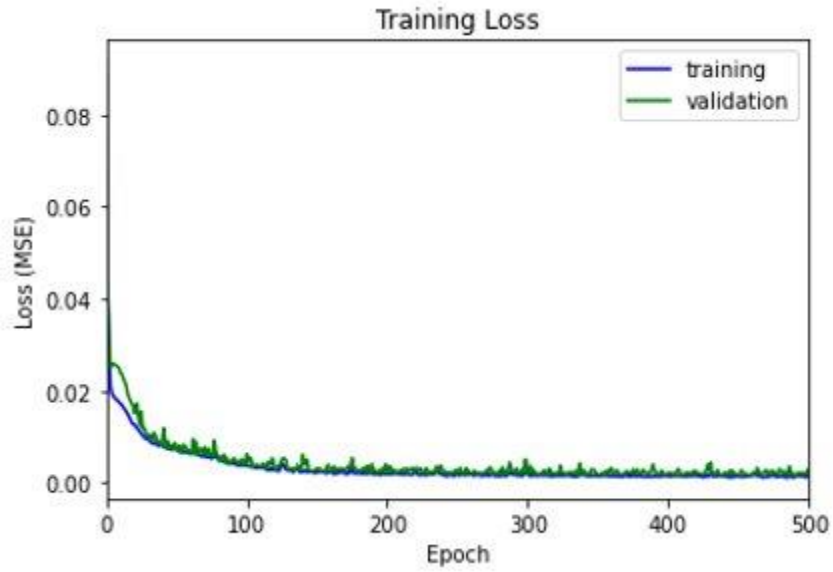


Figure 22: Training loss plot from training a neural network.

Neural network training works by minimizing a loss function that is representative of the predictive performance of the neural network. Each "epoch," or generation, the neural network adjusts its weights and biases according to the performance achieved in the previous epoch (see section 2.4.2).

There are a number of tools that help maximize the output of the training process. One of them is known as a callback. Callbacks work by keeping track of the best loss value achieved during the training process. In Keras, a parameter known as "patience" can be set that controls the number of epochs the neural network will continue to iterate through after the last lowest loss value has been achieved.

After training, the training and validation loss curves were plotted (see Figure 22). It is typical to see a large drop in loss at the start of training, then a slow decay. Figure 22 shows that for this example, training become redundant after around the 200th epoch. The final loss value generally makes it apparent as to how well the neural network learned during training.

3.8 Performance Metrics

When evaluating the performance of any algorithm, it is important to pick and define appropriate metrics. The following metrics were used to quantize performance for the tests ran in this study.

3.8.1 Coefficient of Determination (R^2)

The coefficient of determination is a metric used to determine the strength of the relationship between two variables.

$$R^2 = \left(\frac{n \sum xy - \sum x \sum y}{\sqrt{\left((n \sum x^2 - (\sum x)^2)(n \sum y^2 - (\sum y)^2) \right)}} \right)^2$$

where x represents truth and y represents predictions. This metric is used to evaluate the predicting performance of the neural network. The value ranges between 0 and 1 where 1 is a perfect fit and 0 is no fit. The figure below shows visual representations of data scoring both well and poorly in terms of correlation coefficient (R), which coefficient of determination is derived from.

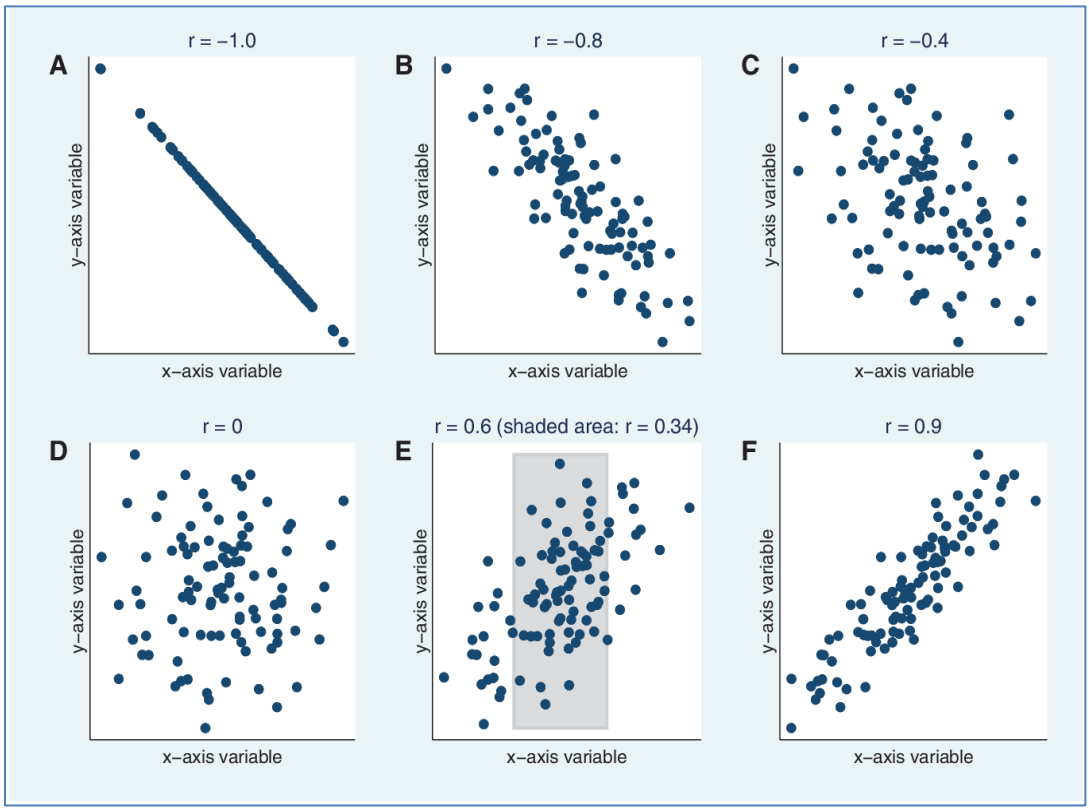


Figure 23: Examples of how correlation coefficent changes with different relationships between x and y-axis variables. [14]

3.8.2 RMSE

RMSE (root mean square error) is used to determine the neural networks', specifically the RPNs', ability to correctly predict the phononic band structure. RMSE is defined as the following:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\left(\boldsymbol{Predicted_i - Actual_i}\right)^2}{\boldsymbol{N}}}$$

where "Predicted" represents the predicted band structure frequency values and "Actual" represents the true frequency values.

3.8.3 Percent Correct

When solving a classification problem, "percent correct" is used as a performance metric. Percent correct is calculated as the number of "correct" answers divided by the total number of decisions made:

$$Percent\ Correct = \frac{Number\ of\ Correct\ Decisions}{Number\ of\ Total\ Decisions} \times 100$$

In the case of the one-hot encoded decisions, which are binary in nature, the parameter corresponding to the number the neural net delivers that is closest to '1' is taken as the decision.

CHAPTER 4. TRAINING DATASETS DESCRIPTIONS AND ANALYSIS

This chapter details the various PnC training datasets used. Training data analyzing scripts were made as a part of the phononics ML package developed alongside this study. These scripts plot a variety of different parameters for a given dataset, and this chapter lays out those plots for each dataset.

For each dataset, the metadata of each dataset is given. This includes the parameter ranges and step size for each sweep, and various plots describing the band structure information.

Next, a series of plots characterizing the bandgap information for each dataset is given. These include analysis of the number of bandgaps per design, the size and center frequencies of the bandgaps, and the effect of the crystal geometry on the bandgap characteristics of the design regimes. These plots are valuable as they provide context to spread of information used to train a neural net and inform the user as to whether a particular dataset is suitable for a given application given its bandgap regime. For these plots, frequency and geometry parameters are reported as normalized to the lattice constant, ‘a’, which is the size of a unit cell in the crystal. Frequency is multiplied by ‘a’ and geometry parameters are divided by ‘a’.

Initially, a batch of data consisting of information for 1010 designs was created. This dataset consisted of square geometry phononic crystals with a circular hole in each unit cell and a lattice constant of 1 micron. The radius of the hole was varied from 0.17e-6 m to 0.47e-6 m in 101 0.03e-6 m increment steps. The height of the host material was varied from 0.2e-6 m to 0.29e-6 m in 10 0.01e-6 m steps. 101 x 10 = 1010 hence 1010 designs. This could be seen as a very small

amount of data to train a neural net or machine learning algorithm with, but it was quick and convenient to generate a smaller amount of data to start testing out the neural net creation.

Upon loading in this batch of data into Python, it was discovered that for this parameter range, only 37 out of 1010 designs (~4%) contain bandgaps. When including these zero-bandgap data points, the neural net tended to always report '0' because the large number of zero compared to non-zero values skewed the regression. Because of this, a toggleable parameter was implemented in the code that lets the user control whether zero-bandgap data points are included when training. Unless stated otherwise, ML results obtained using the bandgap/center frequency method reported in this paper were obtained without the use of zero-bandgap data points.

4.1 Si with Holes

Table 1: Sweep parameters for Si with holes dataset

Sweep Parameters				
	Min	Max	Step size	# Steps
Base height (m)	0.20e-6	0.7e-6	0.010e-6	51
Circle radius (m)	0.17e-6	0.47e-6	0.003e-6	101

Table 2: Other parameters for Si with holes dataset.

Other Dataset Parameters	
Number of designs	5151
Number of designs with bandgaps	442 (8.58%)
Lattice constant (m)	1.00e-6
Inclusion arrangement	Square
Inclusion type	Circular holes

This dataset is the first in a series of three datasets consisting of a substrate (referred to as the “base”) embedded with circular holes. The above table shows that only 8.58% of the designs in this dataset contain bandgaps. This is a relatively small portion of the dataset, especially in the cases where having large amounts of zero valued bandgaps could skew the neural network training.

Most designs contained only 1 bandgap, but some designs contained up to 5 bandgaps. This shows that for methods in which only a single bandgap value is used to train the neural network (see largest bandgap method from Chapter 3), not much information is wasted.

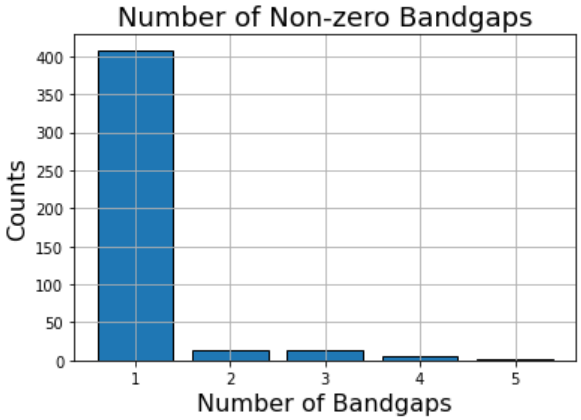


Figure 24: Histogram of number of bandgaps per design.

The majority of the sizes of these bandgaps are below the 100 m/s (100 MHz with the lattice constant a = 1e-6 m) range, with some as high as 500 m/s. The center frequencies of the existing bandgaps mainly fall around 3000 m/s or 3GHz, forming a somewhat Gaussian falloff visible in the figure below.

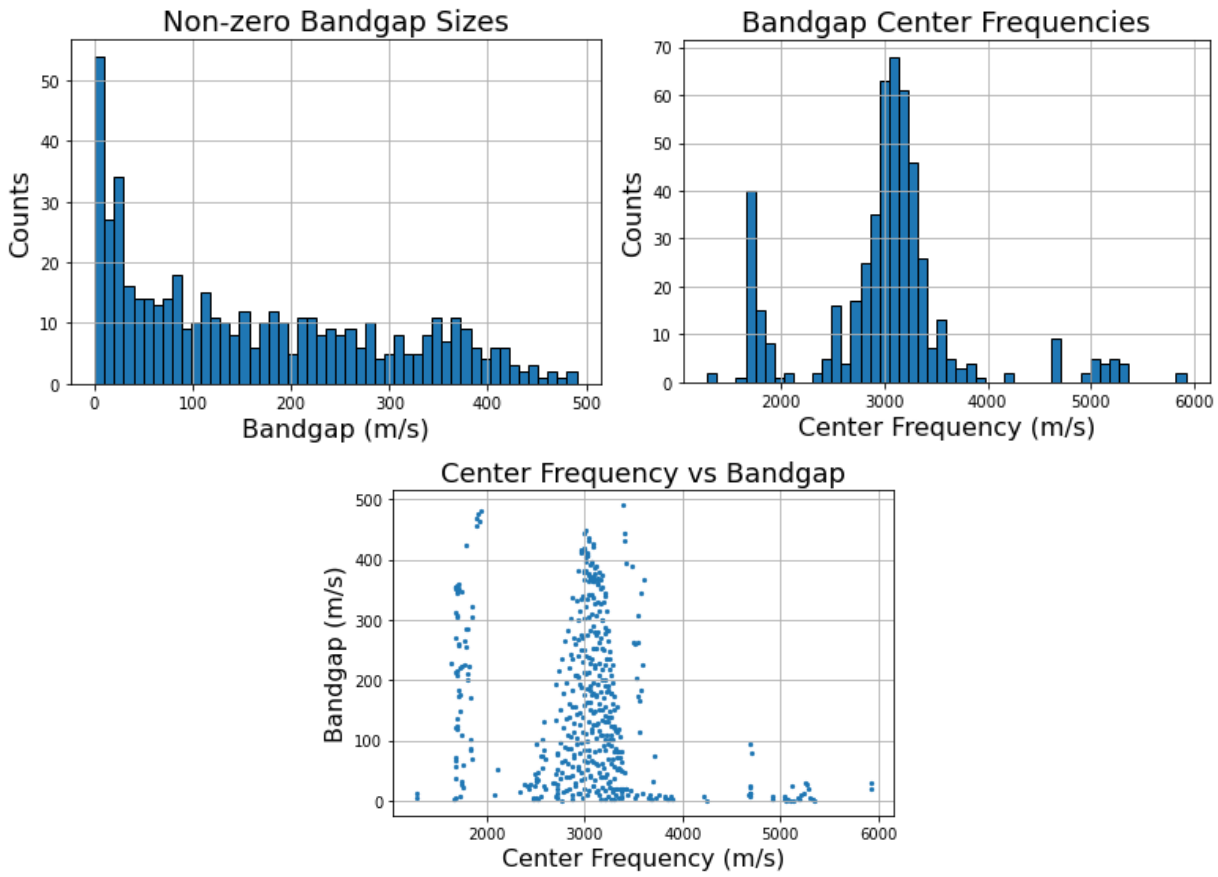


Figure 25: Histogram of bandgap sizes (left). Histogram of corresponding bandgap center frequencies (left). Scatterplot of bandgap/frequency clusters (bottom).

The software package also generates heatmaps that visualize the relationship between the crystal geometry parameters and various band structure information. Shown below are the relationships between geometry both the largest bandgap size per design and the number of bandgaps per design.

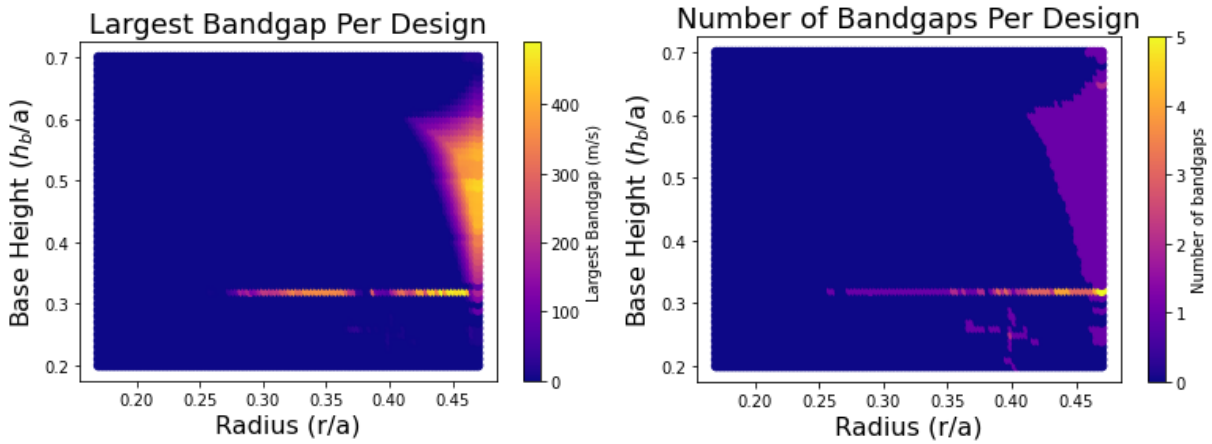


Figure 26: Heatmaps showing hole radius and base height as a function of two different parameters: largest bandgap per design (left) and number of bandgaps per design (right).

A line spanning approximately 0.25 r/a to 0.45 r/a is visible. This line seems to indicate that for one particular base height value, there exists a span of radii in which designs with large number of bandgaps and larger than average bandgap sizes exist. This is believed to be an anomaly in the data generation process and is not seen in any other dataset.

4.2 SiC with Holes

Table 3: Sweep parameters for SiC with holes dataset.

Sweep Parameters				
	Min	Max	Step size	# Steps
Base height (m)	0.20e-6	0.7e-6	0.010e-6	51
Circle radius (m)	0.17e-6	0.47e-6	0.003e-6	101

Table 4: Other parameters for SiC with holes dataset.

Other Dataset Parameters	
Number of designs	5151
Number of designs with bandgaps	383 (7.44%)
Lattice constant (m)	1.00e-6
Inclusion arrangement	Square
Inclusion type	Circular holes

This dataset was intended to be identical to the "Si with holes" dataset in every way except that instead of using Si as the base material, SiC (silicon carbide) was used instead. SiC is a cheap and easy to work with material in terms of PnC fabrication.

From the histogram below showing the number of bandgaps per design in the dataset, the vast majority of designs with bandgaps contained one bandgap, with some containing two.

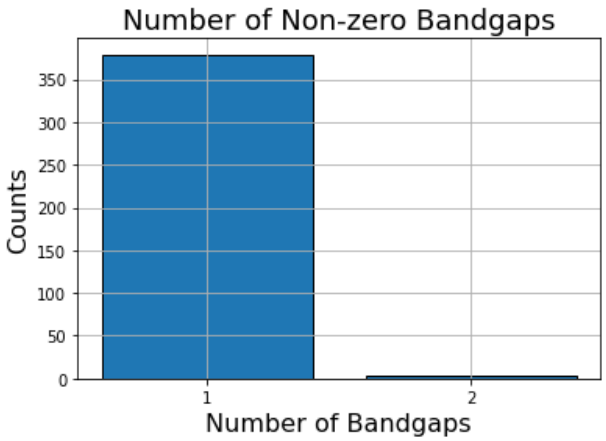


Figure 27: Histogram of number of bandgaps per design.

The bandgap sizes in this dataset are localized in lower frequencies than the previous Si dataset. A large peak of bandgaps can be seen below 50 m/s, with a decaying amount present at higher ranges. The center frequencies for this material are localized in two peaks: one approximately at 4250 m/s and the other at 5400 m/s.

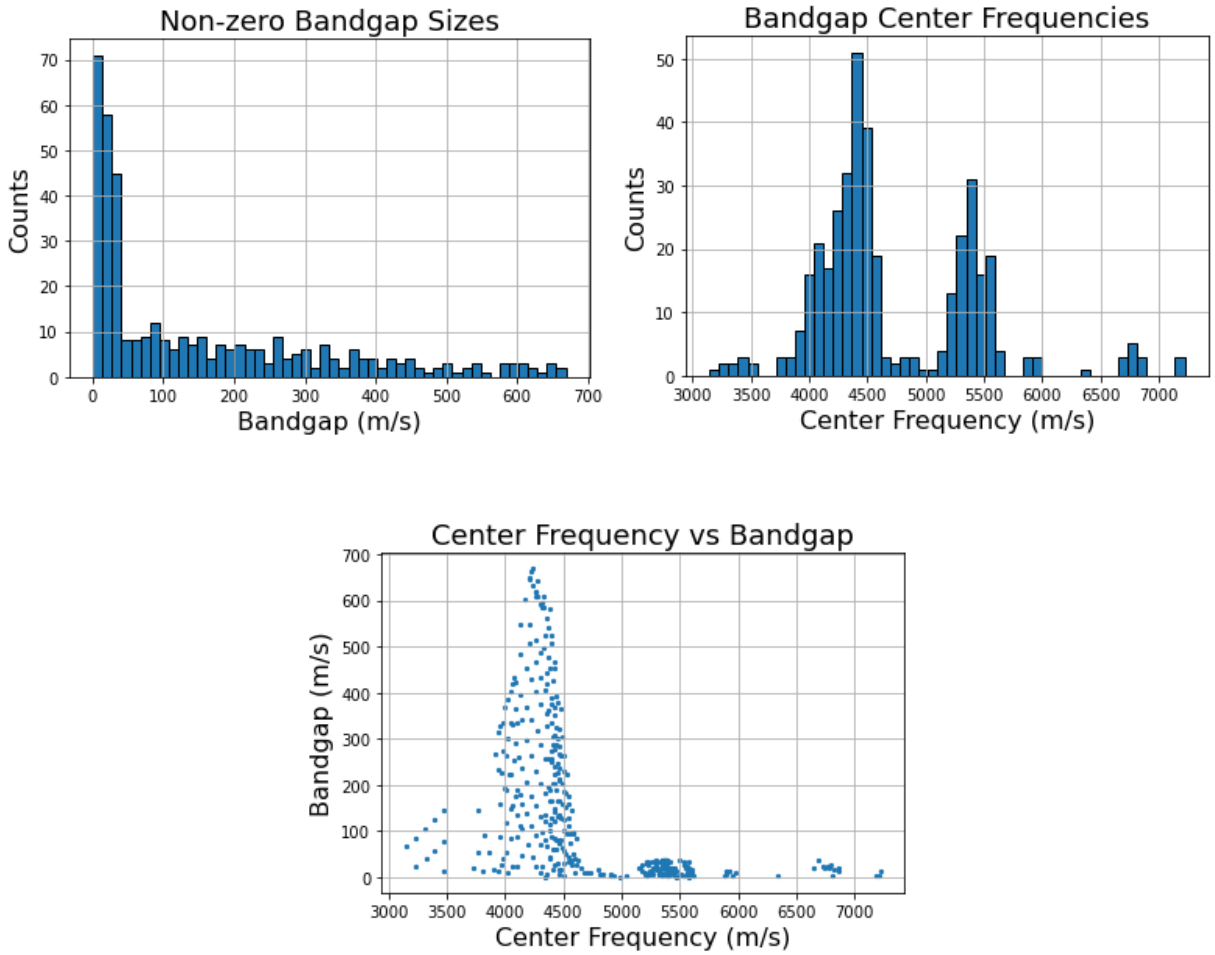


Figure 28: Histogram of bandgap sizes (left). Histogram of corresponding bandgap center frequencies. Scatterplot of bandgap/frequency clusters (bottom).

Looking at the heatmaps for this dataset, the parameter range in which bandgaps exist follows roughly the same shape as the Si dataset. Though there is a more chaotic trend overall,

with small patches of regions containing bandgaps disconnected from the main zone observed in the Si dataset.

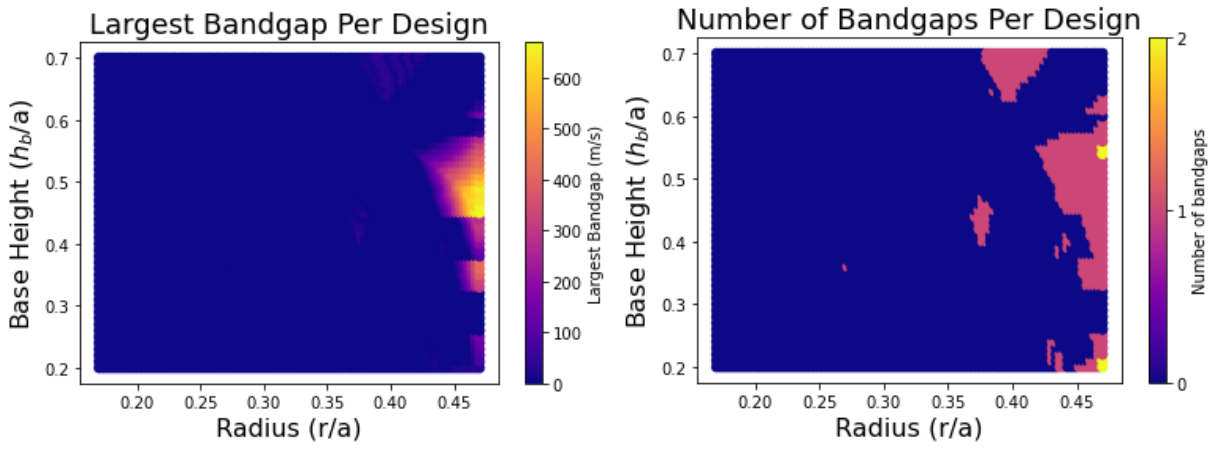


Figure 29: Heatmaps showing hole radius and base height as a function of two different parameters: largest bandgap per design (left) and number of bandgaps per design (right).

4.3 W with Holes

Table 5: Sweep parameters for W with holes dataset.

Sweep Parameters				
	Min	Max	Step size	# Steps
Base height (m)	0.20e-6	0.7e-6	0.010e-6	51
Circle radius (m)	0.17e-6	0.47e-6	0.003e-6	101

Table 6: Other parameters for W with holes dataset.

Other Dataset Parameters	
Number of designs	5151
Number of designs with bandgaps	721 (14.00%)
Lattice constant (m)	1.00e-6
Inclusion arrangement	Square
Inclusion type	Circular holes

Similarly to the SiC with holes dataset, this dataset is was generated with the same geometry sweep parameters as the Si with holes dataset, except that W (tungsten) was used as the base material. W is another material that has gained interest for use in PnCs.

The “Other Dataset Parameters” table shows that 14% of the designs in this dataset contain bandgaps. This is nearly double the amount found in the other Si and SiC datasets. The number of bandgaps per design follows a similar trend to the SiC dataset, with the majority of designs containing one bandgap, and a small amount containing two.

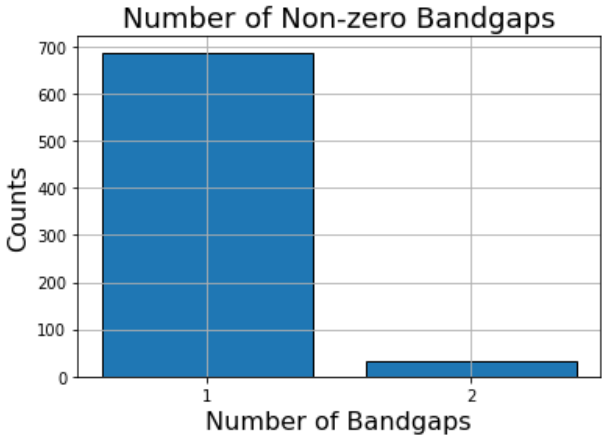


Figure 30: Histogram of number of bandgaps per design.

Similarly to the SiC dataset, the bandgap sizes are highly localized to the below 50 m/s range. Also similarly, the number of bandgaps of larger widths decrease, though with a sharper decline. Center frequencies of these bandgaps are found within two main peaks as well, at approximately 1700 m/s and 2000 m/s.

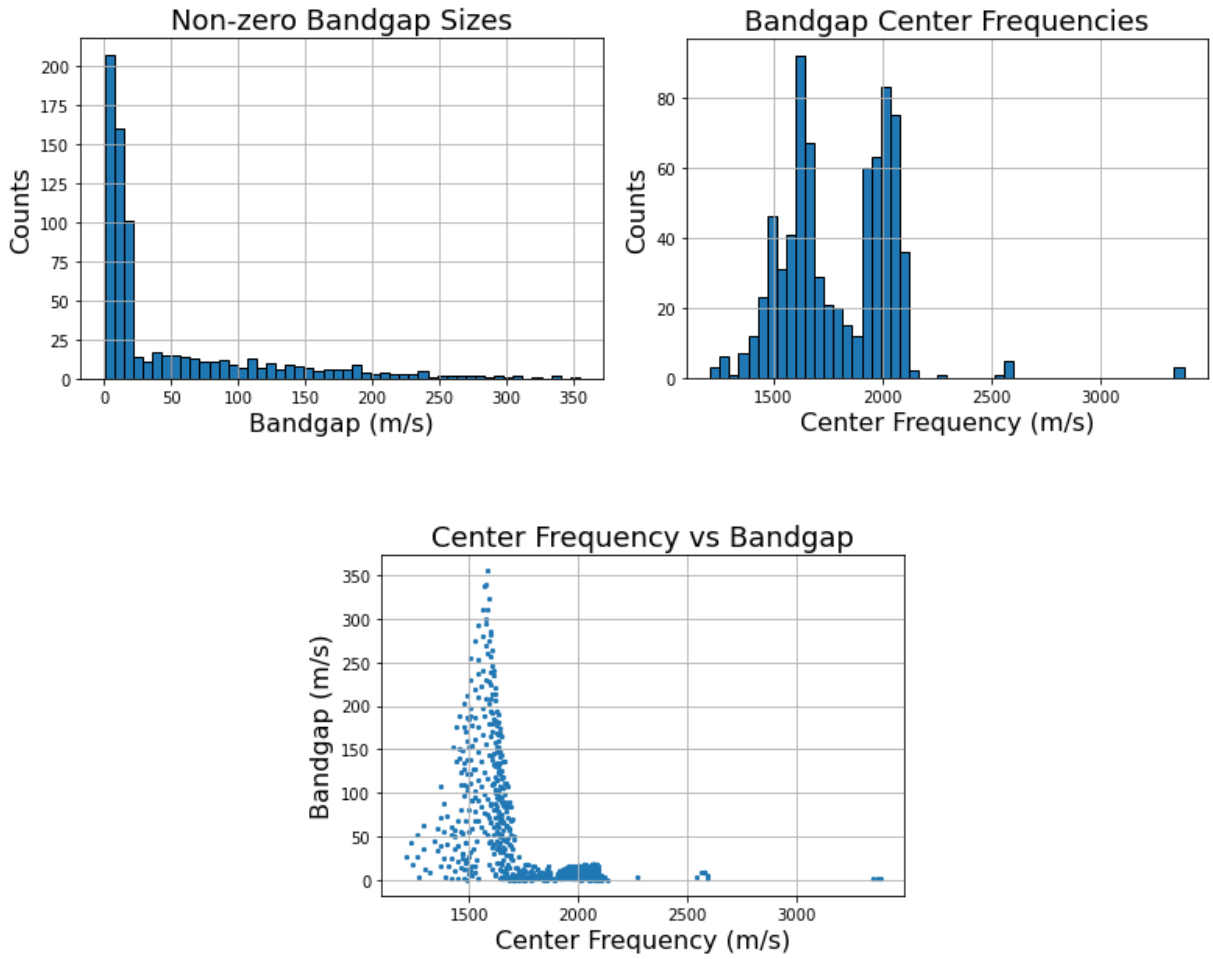


Figure 31: Histogram of bandgap sizes (left). Histogram of corresponding bandgap center frequencies. Scatterplot of bandgap/frequency clusters (bottom).

The heatmaps for this dataset follow similar trends to the SiC dataset, except that from the "Number of Bandgaps Per Design" heatmap, a new, large cluster containing bandgaps can be seen between 0.5 and 0.7 base height. The points at which this cluster approximately intersects with the main cluster found in the previous two datasets contain two bandgaps.

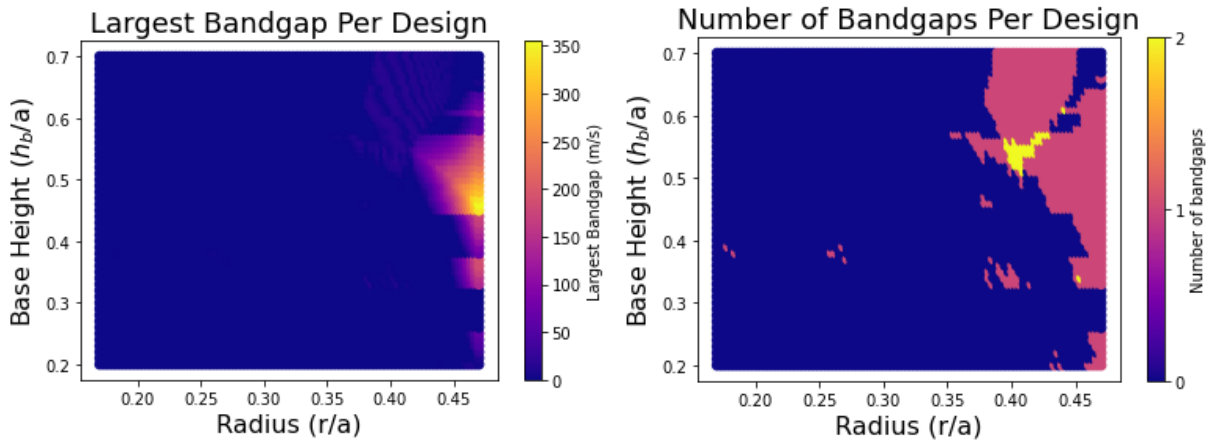


Figure 32: Heatmaps showing hole radius and base height as a function of two different parameters: largest bandgap per design (left) and number of bandgaps per design (right).

4.4 Si, SiC, W With Holes Combined

For this dataset, the previous Si, SiC, and W-containing datasets were combined. As such, the sweep parameters and parameter ranges were identical to the pure Si, SiC, and W dataset.

This dataset brought with it the first discrete parameter for the neural network to consider: base material. This made it possible to begin experimenting with classification problems i.e., identifying the correct material. Ideally, when searching for PnC designs given desired band characteristics, it is necessary to query for both continuous (height, radius, etc.) and discrete (material, pillar shape, etc.) parameters.

4.5 Si With W Pillars

Table 7: Sweep parameters for Si with W pillars dataset.

Sweep Parameters				
	Min	Max	Step size	# Steps
Base height	0.20e-6	0.70e-6	0.0125e-6	41
Pillar radius	0.17e-6	0.47e-6	0.0075e-6	41
Pillar height	0.20e-6	0.70e-6	0.02e-6	26

Table 8: Other parameters for Si with W pillars dataset.

Other Dataset Parameters	
Number of designs	43706
Number of designs with bandgaps	42966 (98.31%)
Lattice constant (m)	1.00e-6
Inclusion arrangement	Square
Inclusion type	Circular pillars

For this dataset, cylindrical pillars were chosen as the inclusion type. Using pillars in the crystal design results in three parameters to sweep over as compared to two in previous datasets: base height, pillar height, and pillar radius. The total number of data points is a result of 41 base heights x 41 pillar radii x 26 pillar heights = 43706. Because of the extra parameter introduced, DPN results for datasets containing pillars will include pillar height prediction performance. Pillar radius is treated as the same parameter as the circular hole radius in previous datasets.

It is notable that introducing pillars as the inclusion geometry leads to a large amount of non-zero bandgaps as compared to the datasets with circular holes. This is due to the added geometrical complexity of the pillar.

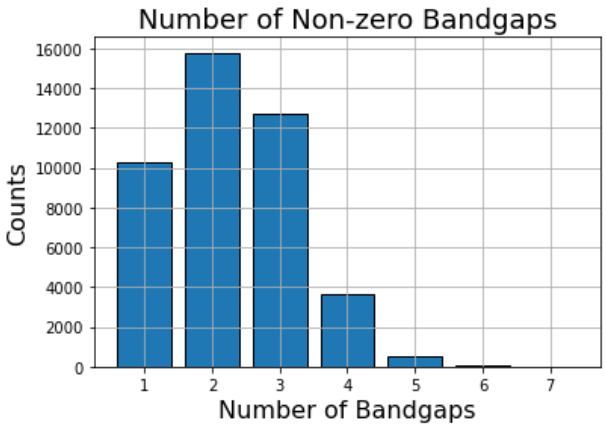


Figure 33: Histogram of number of bandgaps per design.

This dataset contains by far the largest number of low (sub 20 m/s) bandgap sizes. The center frequencies lie mainly in two peaks at around 1600 m/s and 2250 m/s.

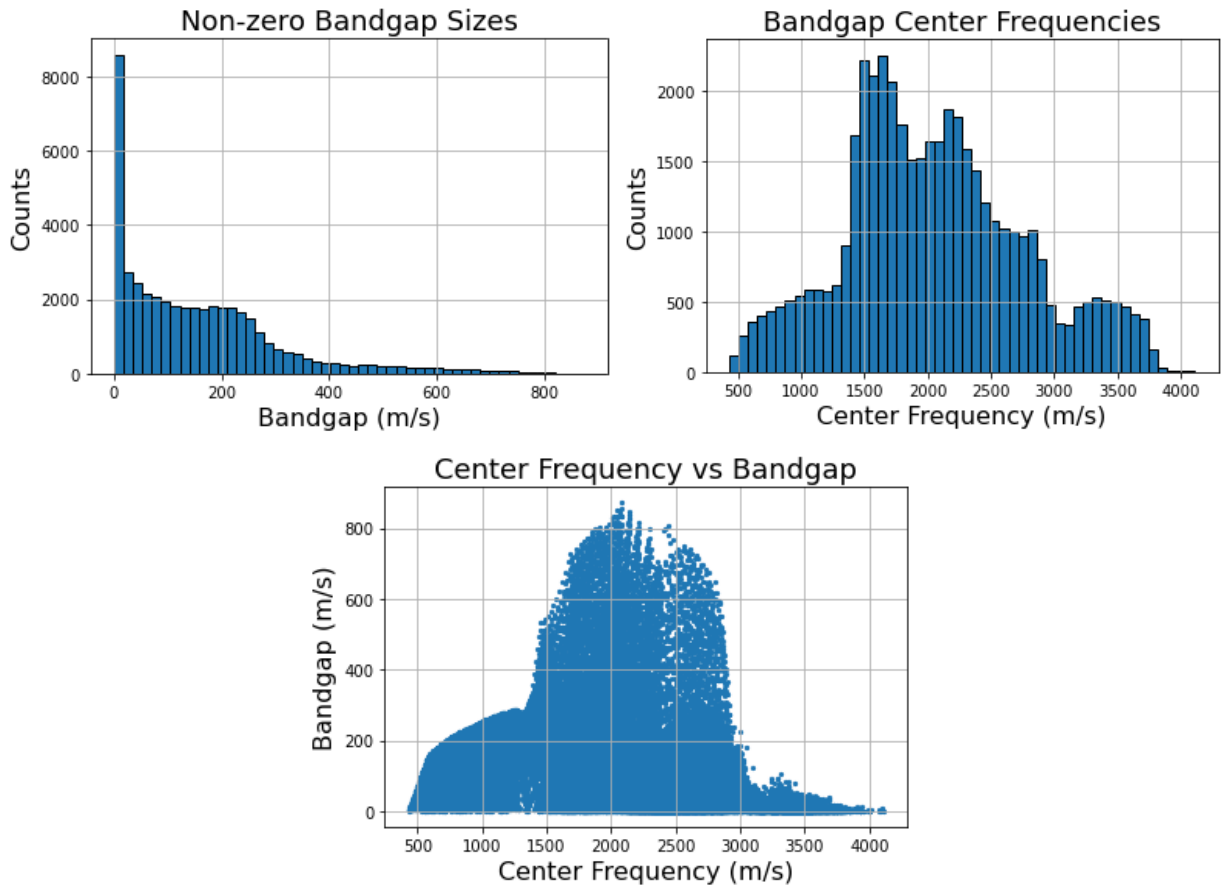


Figure 34: Histogram of bandgap sizes (left). Histogram of corresponding bandgap center frequencies (right). Scatterplot of bandgap/frequency clusters (bottom).

As a third geometrical parameter was included in this dataset, the generating the heatmaps shown in the descriptions of the other dataset requires showing the relationship between each of the three parameters, resulting in relations between radius and base height, radius and pillar height, and pillar height and base height. The resulting 6 heatmaps are shown below:

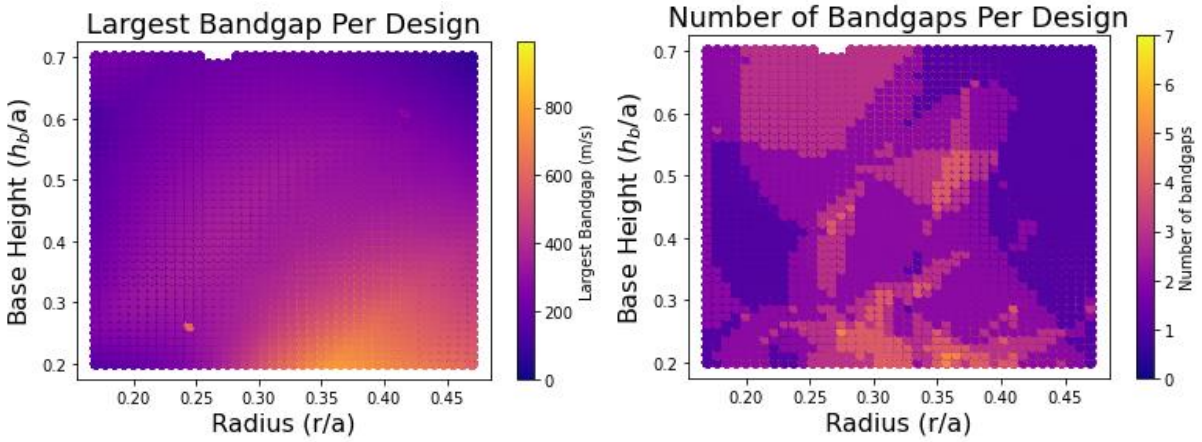


Figure 35: Heatmaps showing hole radius and base height as a function of two different parameters: largest bandgap per design (left) and number of bandgaps per design (right).

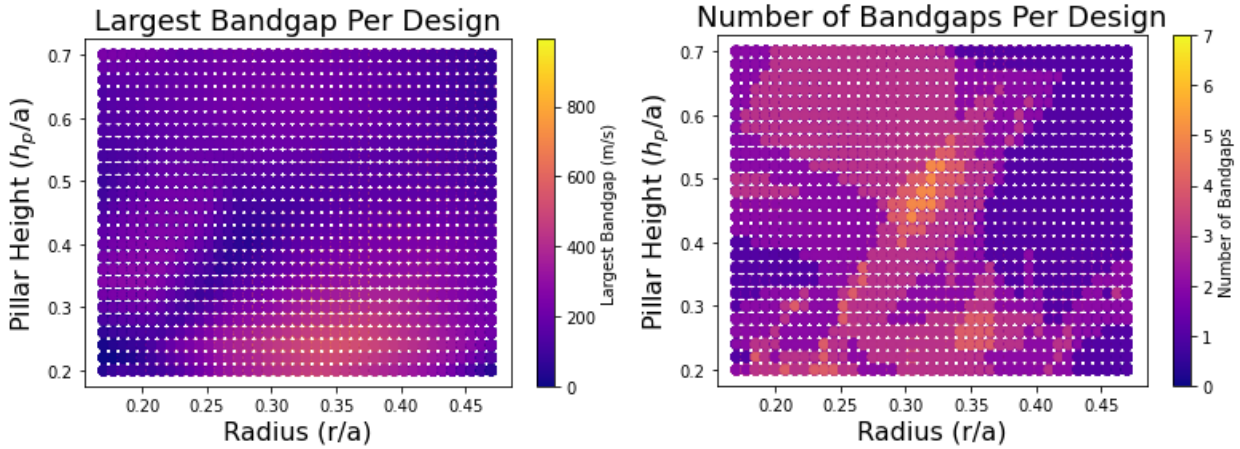


Figure 36: Heatmaps showing hole radius and pillar height as a function of two different parameters: largest bandgap per design (left) and number of bandgaps per design (right).

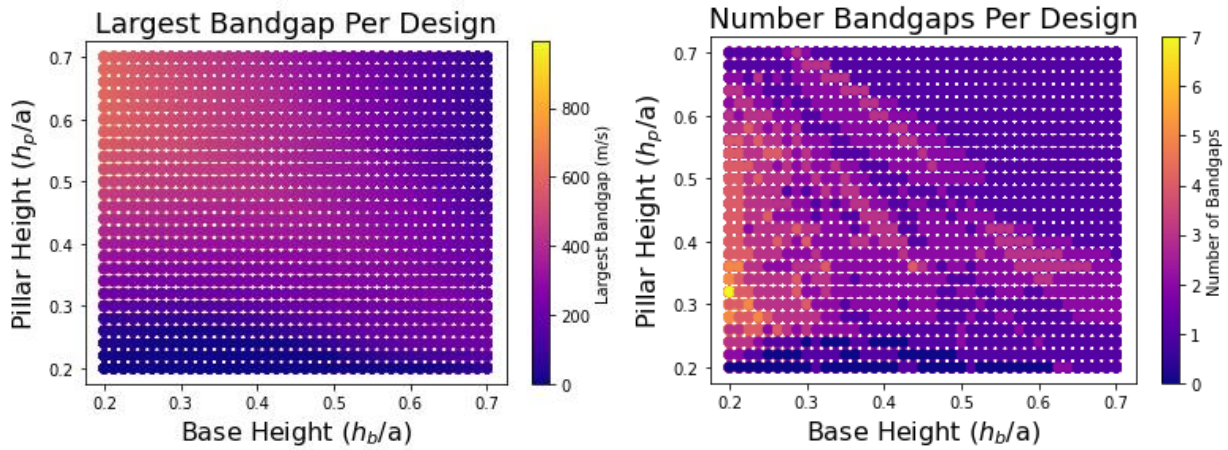


Figure 37: Heatmaps showing base height and pillar height as a function of two different parameters: largest bandgap per design (left) and number of bandgaps per design (right).

4.6 SiC With W Pillars

Table 9: Sweep parameters for SiC with W pillars dataset.

Sweep Parameters				
	Min	Max	Step size	# Steps
Base height	0.20e-6	0.70e-6	0.0125e-6	41
Pillar radius	0.17e-6	0.47e-6	0.0075e-6	41
Pillar height	0.20e-6	0.70e-6	0.02e-6	26

Table 10: Other parameters for SiC with W pillars dataset.

Other Dataset Parameters	
Number of designs	43706
Number of designs with bandgaps	40319 (92.25%)
Lattice constant (m)	1.00e-6
Inclusion arrangement	Square
Inclusion type	Circular pillars

This dataset's generation parameters were identical to those of the Si with W pillars dataset, but here the base material is chosen to be SiC.

The number of designs with bandgaps is again found to be much higher than any of the datasets containing circular holes, at about 93%. This is a notable difference between this and the previous pillar-containing dataset however, which may need to be taken into consideration when running tests that are sensitive to the number of designs with zero bandgaps. Of the designs with bandgaps, we can see that most contained only one, similar to the datasets with circular holes.

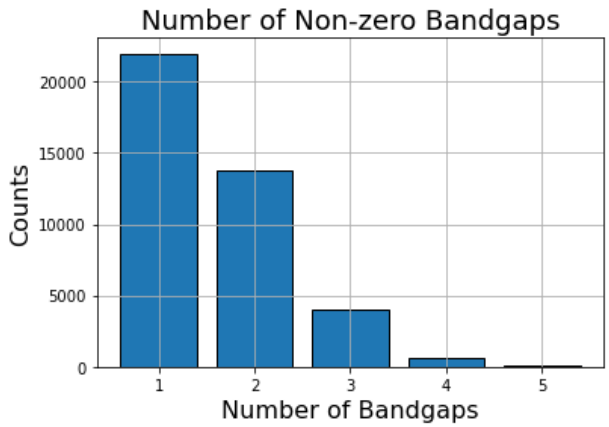


Figure 38: Histogram of number of bandgaps per design (left).

The bandgap sizes in this dataset are similar to those found in the previous dataset with pillars. The center frequencies though are more localized to a single peak close to 2000 m/s.

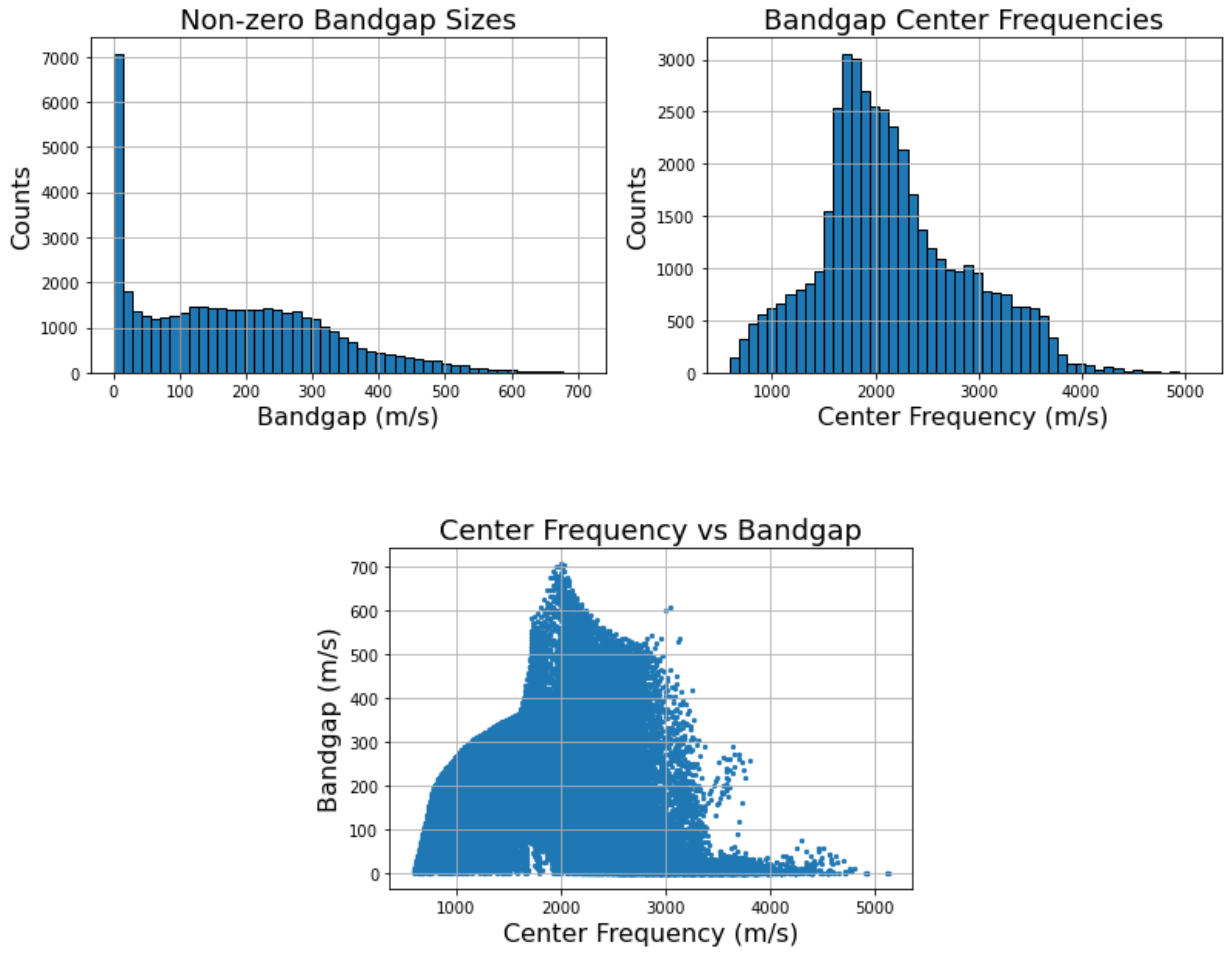


Figure 39: Histogram of bandgap sizes (left). Histogram of corresponding bandgap center frequencies. Scatterplot of bandgap/frequency clusters (right).

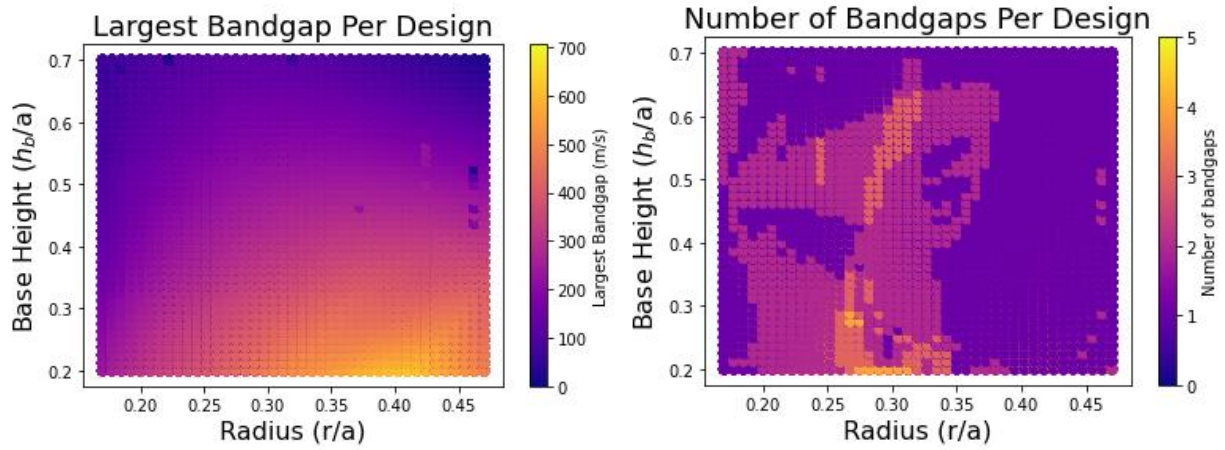


Figure 40: Heatmaps showing hole radius and base height as a function of two different parameters: largest bandgap per design (left) and number of bandgaps per design (right).

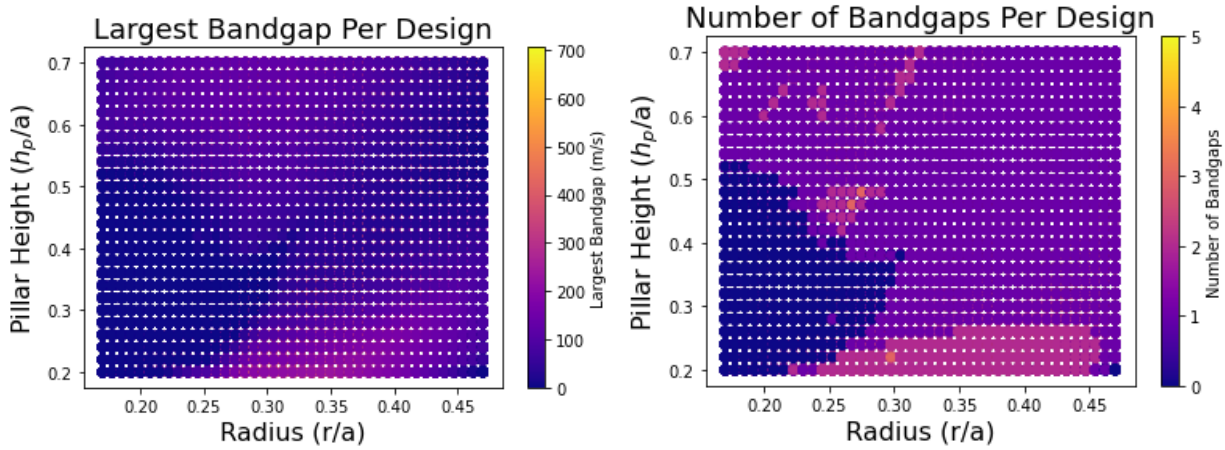


Figure 41: Heatmaps showing hole radius and pillar height as a function of two different parameters: largest bandgap per design (left) and number of bandgaps per design (right).

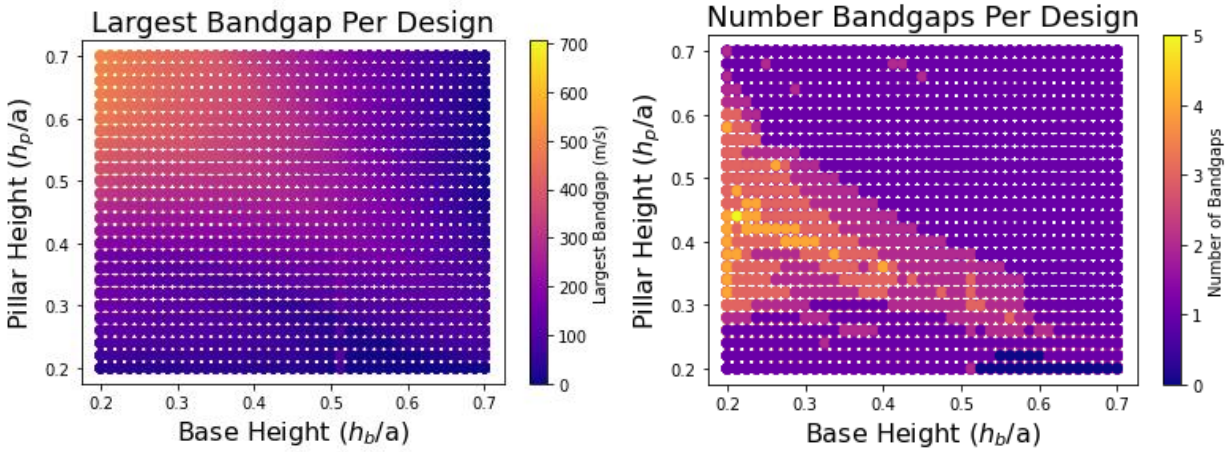


Figure 42: Heatmaps showing base height and pillar height as a function of two different parameters: largest bandgap per design (left) and number of bandgaps per design (right).

CHAPTER 5. MACHINE LEARNING STUDIES

This chapter contains the results for the various machine learning studies performed. The chapter is split into three sections: RPN studies, DPN studies (containing a proposed best inverse design solution), and a section exploring the training-testing split.

The RPN and DPN sections are divided into subsections, each containing results for the different methods of formatting the training data (largest bandgap, DOS, full band structure). Within each of these sections, the same results are repeated for either all or a subset of the training datasets described in the previous chapter.

The results in this chapter do not necessarily represent the best possible results obtainable as a tuning experiment could offer slight performance optimizations. The neural network parameters used for each experiment are recorded in the Appendix section of this paper.

5.1 RPN Results

This section contains the performance results acquired from training response predicting networks (RPN) with the training data described in the previous chapter, formatted in various methods. As the RPN aims to predict the properties of the phononic band behavior given crystal geometry parameters, this section catalogues the best prediction results gained for each test.

5.1.1 Largest Bandgap Method

The first attempted machine learning experiment was with the largest bandgap method using an RPN. The natural first step was first attempt this with the datasets containing circular holes as they hold the smallest number of variable geometry parameters.

At first, the neural network was showing very low predictive correlation with the test data. Almost all the predicted values were zero or close to zero. Since the circular hole datasets contain only a small number of designs with bandgaps (see Chapter 4 for details about training datasets), most of the output values in the training set were zero for this method. This was influencing the neural network to associate almost every design with zero bandgap and center frequency. As mentioned in Chapter 4, a parameter was added in the code that controls whether the zero bandgap designs are included in the training set. After removing the zero bandgap designs and training, we arrive at the results in the figures below:

Si with Holes

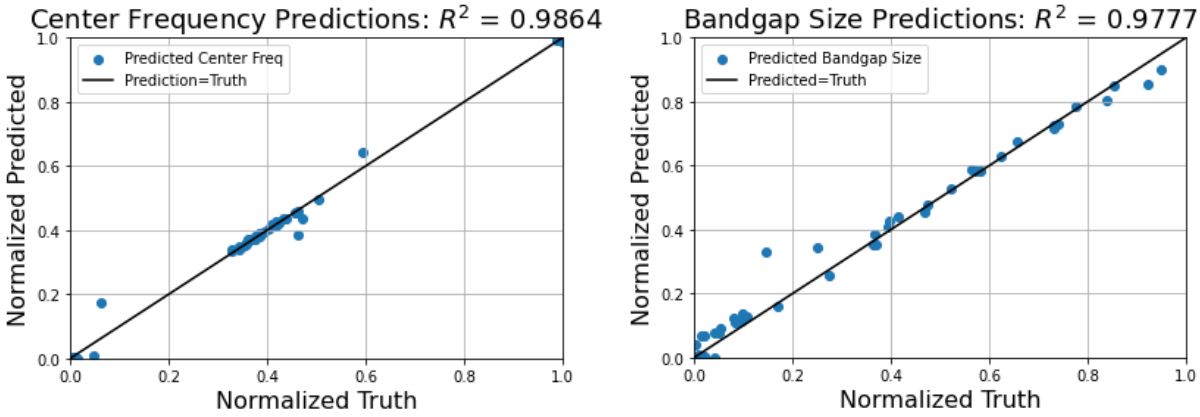


Figure 43: RPN results for “Si with holes” dataset formatted using the largest bandgap method.

SiC with Holes

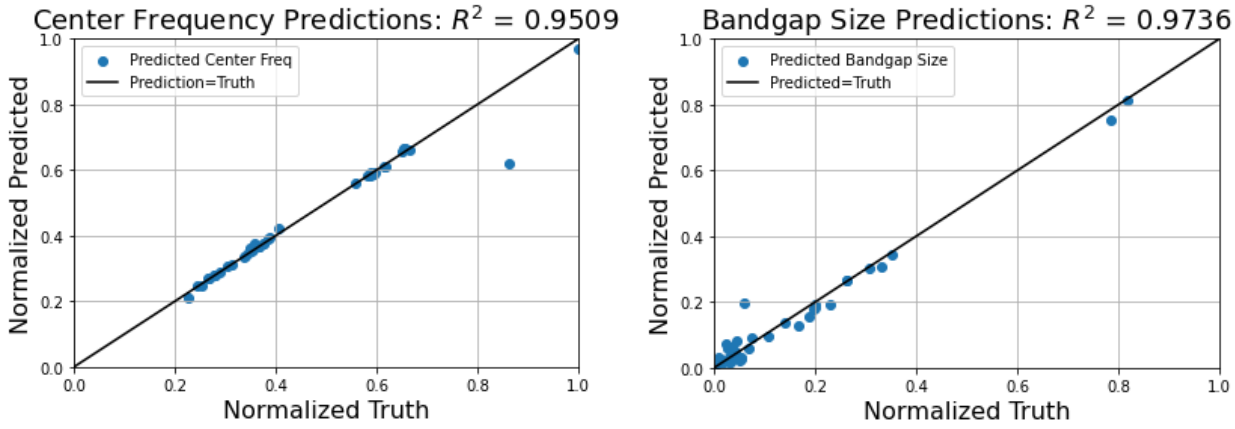


Figure 44: RPN results for "SiC with holes" dataset formatted using the largest bandgap method.

W with Holes

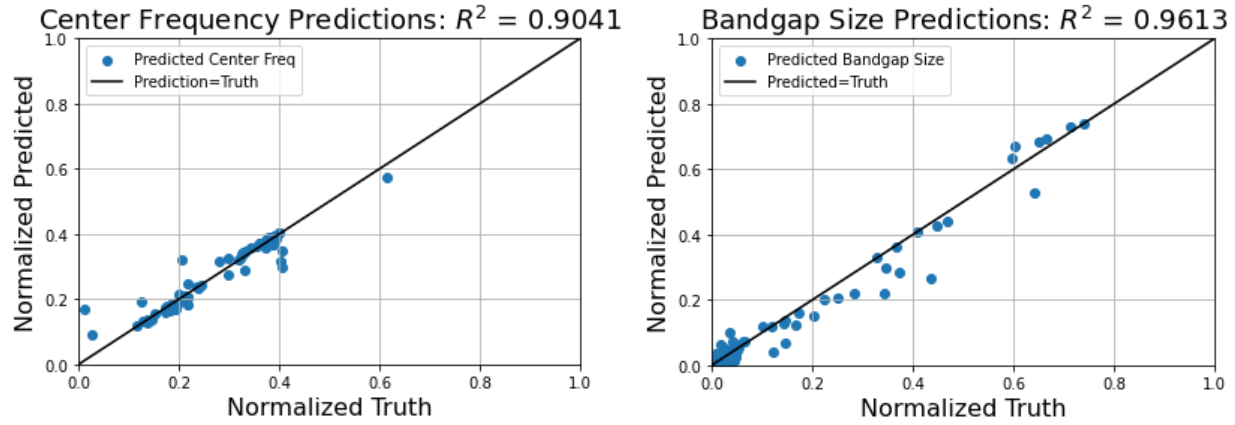


Figure 45: RPN results for "W with holes" dataset formatted using the largest bandgap method.

Si, SiC, W Combined

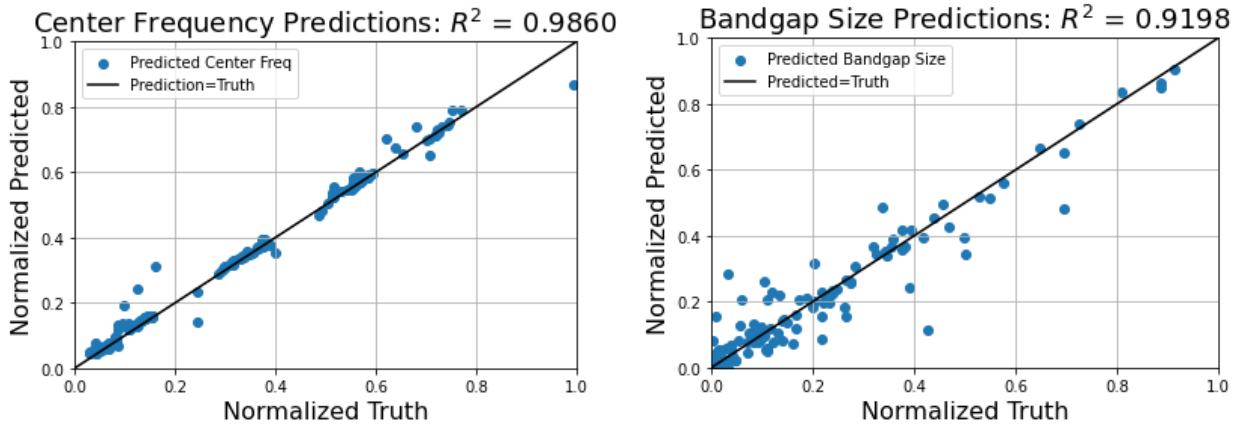


Figure 46: RPN results for "Si, SiC, W combined" dataset formatted using the largest bandgap method.

After removing the zero bandgap designs from the training set, the neural network shows strong predictive ability using this method. The neural network was also able to perform well when given the combined dataset of all three materials, showing that it can adapt its predictions when given the base material type as input.

Next, the datasets containing pillars were tested:

Si with W Pillars

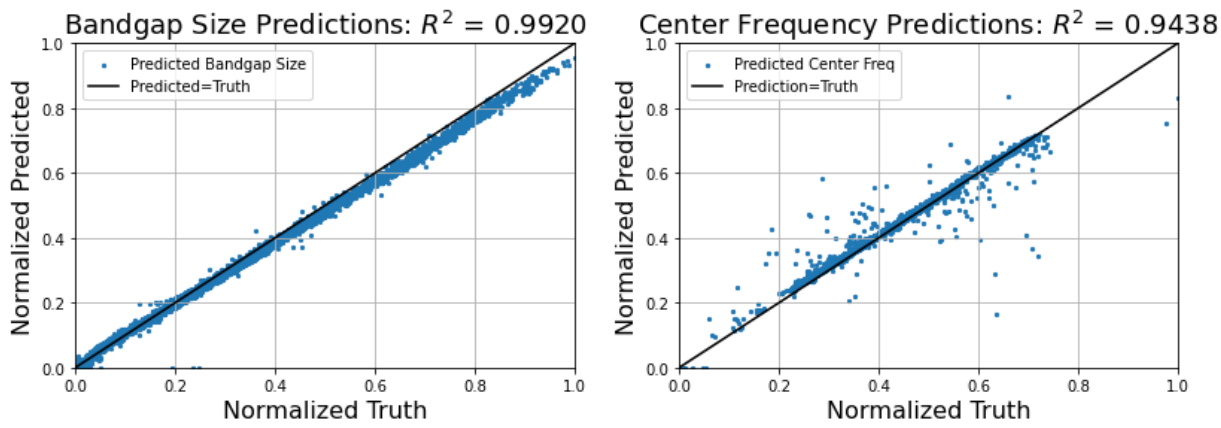


Figure 47: RPN results for "Si with W pillars" dataset formatted using the largest bandgap method.

SiC with W Pillars

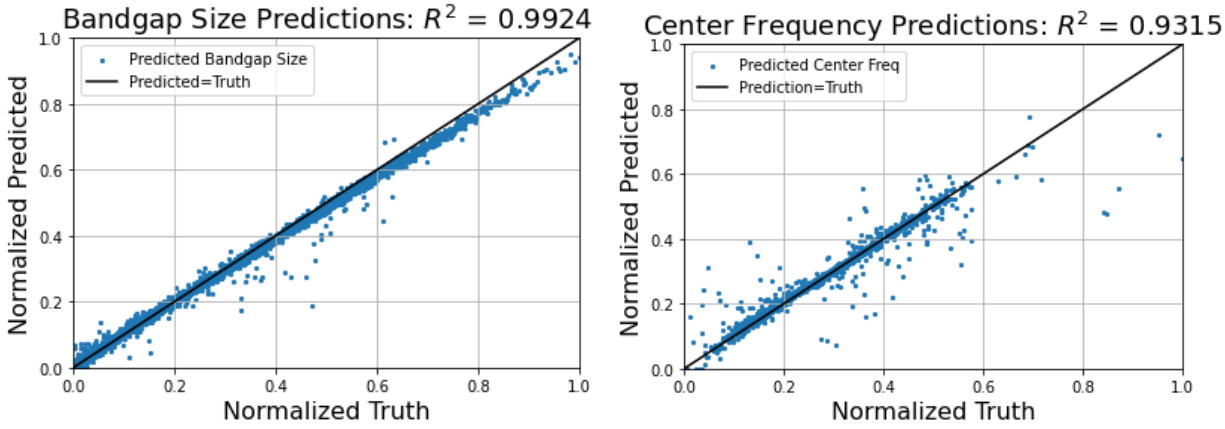


Figure 48: RPN results for "SiC with W pillars" dataset formatted using the largest bandgap method.

A slight increase in performance over the circular hole datasets is likely a function of one or two factors: first, the pillar datasets have a much higher number of non-zero bandgap datapoints. It is possible that having a low number of data points to train on has a larger negative impact on this formatting method than others. In section 5.3, a test is specifically performed on the effect

lowering the amount of data available to train the neural network, but that test is not performed using the largest bandgap method. Second, the pillar datasets include three geometric parameters for the neural network to recognize patterns from compared to the two found in the circular hole datasets.

It is also noteworthy that performance results varied much for the circular hole datasets between training "runs," even when those runs were using the same exact parameters. No formal test was done to measure the variance of results between runs, but it was observed that the coefficient of determination for the circular holes datasets tended to vary on the magnitude of 0.1 while the coefficient of determination for the pillar datasets varied on the order of 0.01.

These results show that the ability to use geometry parameters to predict the largest bandgap and center frequency is viable. The necessary level of predictive performance depends on the application, but all coefficient of determination values were above 0.90, showing strong correlation between predicted and truth.

5.1.2 DOS Method

As the DOS was conceived as a tool for more easily representing the band structure during crystal geometry predictions (a DPN focused application), the DOS method was not thoroughly tested using an RPN. This capability is programmed into the package used for this study, however, and it would be possible to perform this experiment if the interest arises.

5.1.3 Band Structure Method

For this test, the ability for a neural network to predict a full band structure was tested. First, a neural network is trained using band structure information. Once training is complete, the

neural network is shown the geometry parameters of the data that was reserved for testing and a band structure diagram is generated. An example of the true band structure compared to the neural network's prediction are shown below.

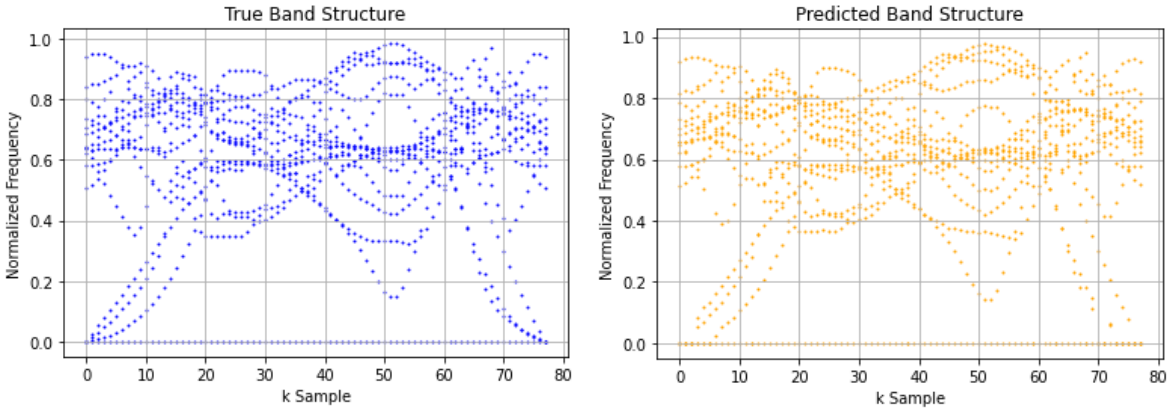


Figure 49: Phononic band structure taken from the training data (left/blue) and the neural network band structure prediction when given the corresponding unit cell geometry (right/yellow). Each array contains 78 samples. Frequency values are normalized to be between 0 and 1 based on the values from all designs in the training dataset.

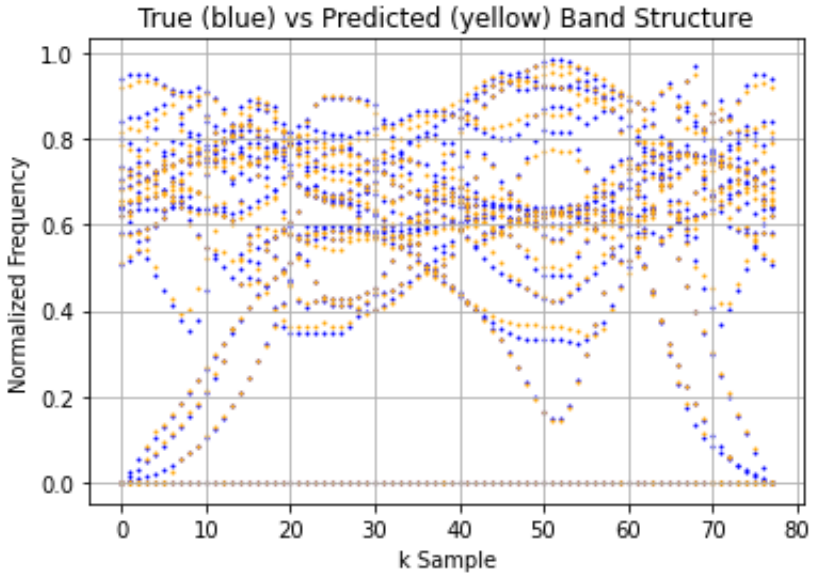


Figure 50: True (blue) and Predicted (yellow) band structure from the previous figure overlapped for comparison. For each x-axis value, there exists a truth and predicted value. The y-axis distance between them is the prediction error.

To quantify performance, each predicted frequency value is compared to the equivalent truth value. The metric used is RMSE (defined in section 3.8). The RMSE value is calculated for each truth, prediction band structure pair in the testing set, and the resulting RMSE values are histogrammed to better understand the spread of errors throughout the test set.

Included with the histogram of RMSE values for each experiment is a histogram of the frequency range values contained in each band structure of the test set. The reason for this is that another rough metric for prediction accuracy can be obtained by comparing the average prediction error to the average range of frequency values. I.e., seeing higher frequency prediction error when the frequency ranges are higher is preferable to seeing the same high error when the average ranges are lower. The figures below show the results of this test for each dataset:

Si with Holes

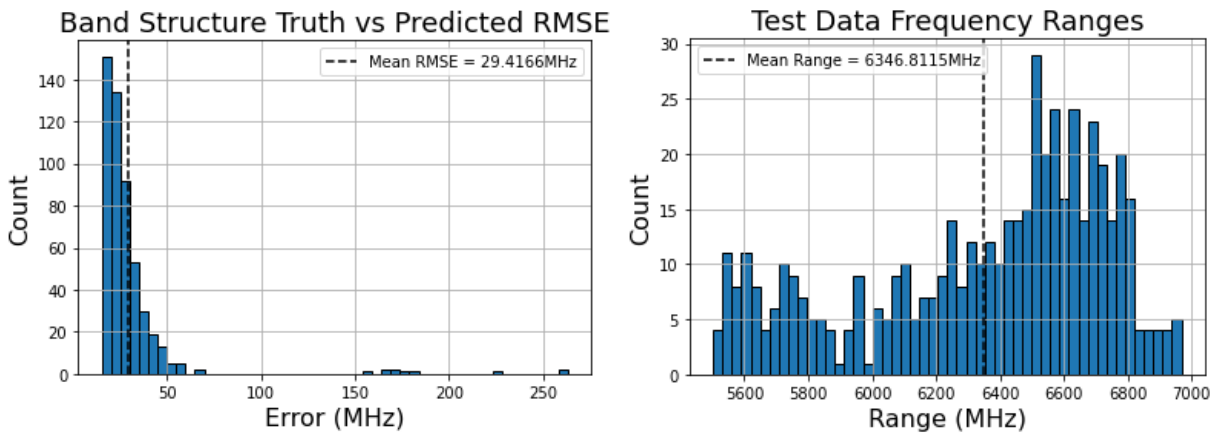


Figure 51: Histogram of band structure prediction errors for "Si with holes" dataset (left). Histogram of training dataset band structure frequency ranges (right).

This test was first done using the “Si with holes” dataset. The mean frequency prediction RMSE for was approximately 29.42 MHz, while the mean band structure frequency range for the test set was approximately 6347 MHz. Dividing these two means gets us a rough metric for average percent prediction error:

29.42 MHz / 6347 MHz = 4.635e-3 = 0.4635% average error

W with Holes

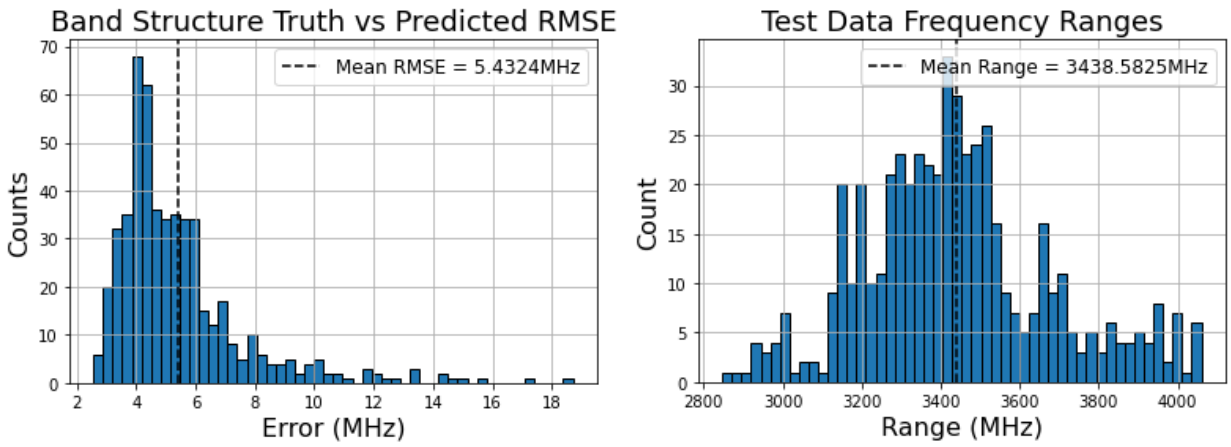


Figure 52: Histogram of band structure prediction errors for "W with holes" dataset (left). Histogram of training dataset band structure frequency ranges (right).

The mean frequency prediction RMSE for “W with holes” was approximately 5.432 MHz, while the mean band structure frequency range for the test set was approximately 3439 MHz. Dividing these two means gets us a rough metric for average percent prediction error:

5.432 MHz / 3439 MHz = 1.580e-3 = 0.1580% average error

SiC with Holes

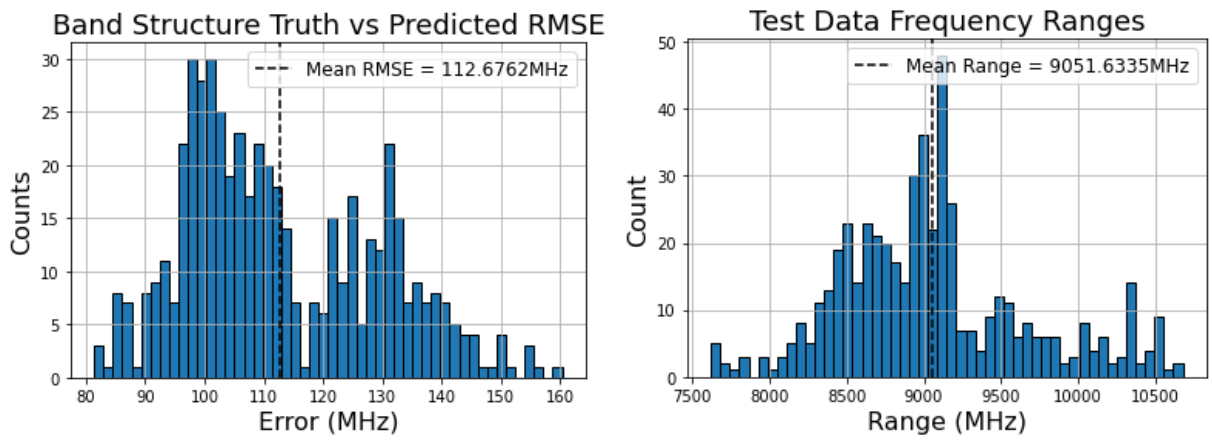


Figure 53: Histogram of band structure prediction errors for "SiC with holes" dataset (left). Histogram of training dataset band structure frequency ranges (right).

The mean frequency prediction RMSE for “SiC with holes” was approximately 112.7 MHz, while the mean band structure frequency range for the test set was approximately 9052 MHz. Dividing these two means gets us a rough metric for average percent prediction error:

112.7 MHz / 9052 MHz = 12.45e-3 = 1.245% average error

Si with W Pillars

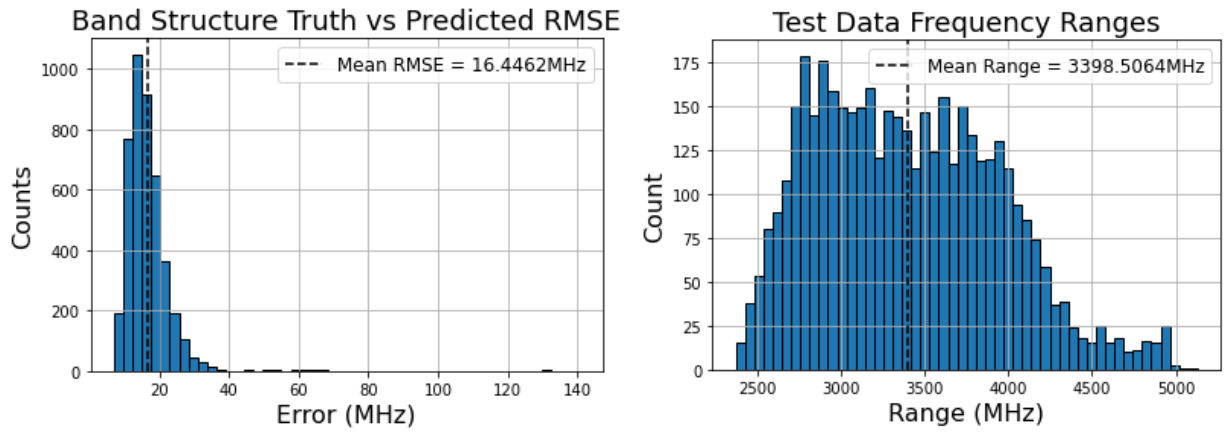


Figure 54: Histogram of band structure prediction errors for "Si with W pillars" dataset (left). Histogram of training dataset band structure frequency ranges (right).

The mean frequency prediction RMSE for “SiC with holes” was approximately 16.45 MHz, while the mean band structure frequency range for the test set was approximately 3399 MHz. Dividing these two means gets us a rough metric for average percent prediction error:

16.45 MHz / 3399 MHz = 4.840e-3 = 0.04839% average error

SiC with W Pillars

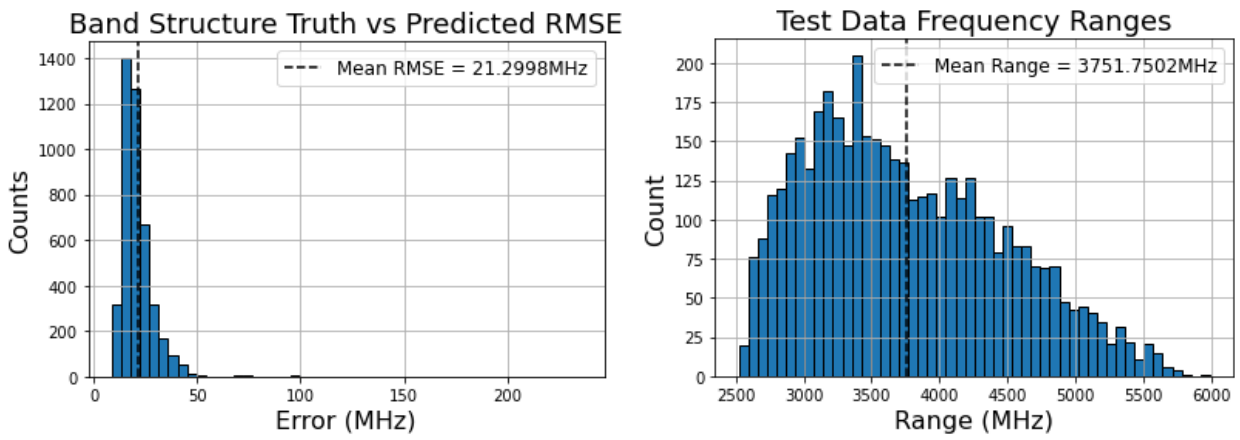


Figure 55: Histogram of band structure prediction errors for "SiC with W pillars" dataset (left). Histogram of training dataset band structure frequency ranges (right).

The mean frequency prediction RMSE for “SiC with holes” was approximately 21.30 MHz, while the mean band structure frequency range for the test set was approximately 3752 MHz. Dividing these two means gets us a rough metric for average percent prediction error:

21.30 MHz / 3751 MHz = 5.678e-3 = 0.05678% average error

Overall, the neural networks’ ability to predict band structure given crystal geometry is favorable. From the performance results, we can expect an average error close to 1% or less when making predictions on individual points in the band structure. It is again important to note, that these performance values are not necessarily (and not likely) optimized as finding the optimal configuration of neural network tuning parameters is a lengthy process that was not thoroughly explored in this paper’s study. The high level of performance of this method is significant as it is used in the proposed inverse design solution given in section 5.2.3.2.

5.2 DPN Results

This section contains the prediction results obtained from training neural networks in a DPN (design predicting network) configuration. As implied by the word “design” in “design predicting network,” these neural networks were tasked with predicting crystal design geometry given band structure information.

5.2.1 Largest Bandgap Method

Similarly to the RPN results, the largest bandgap method was first checked using the DPN. The results from the datasets containing circular holes are first shown in the figures below:

Si with Holes

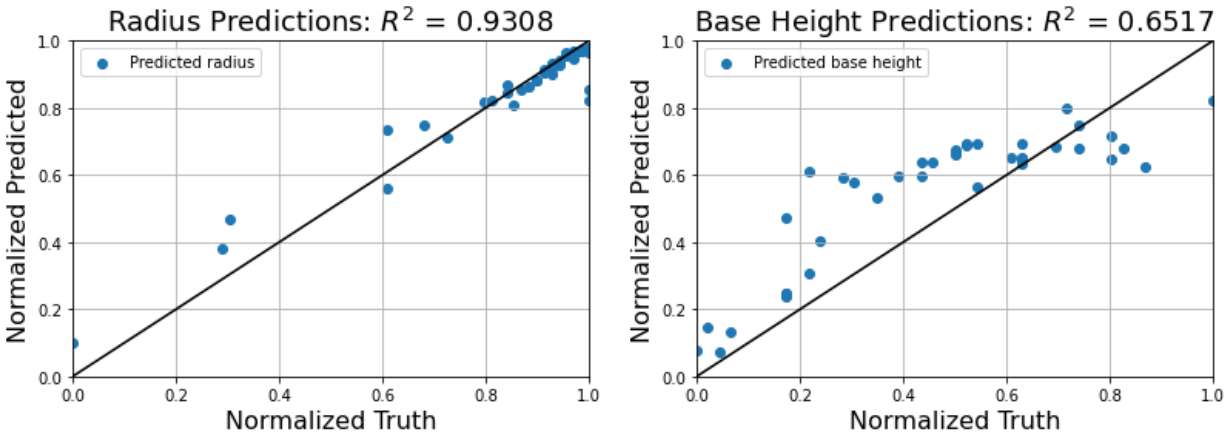


Figure 56: DPN results using the “Si with Holes” dataset and largest bandgap method.

SiC with Holes

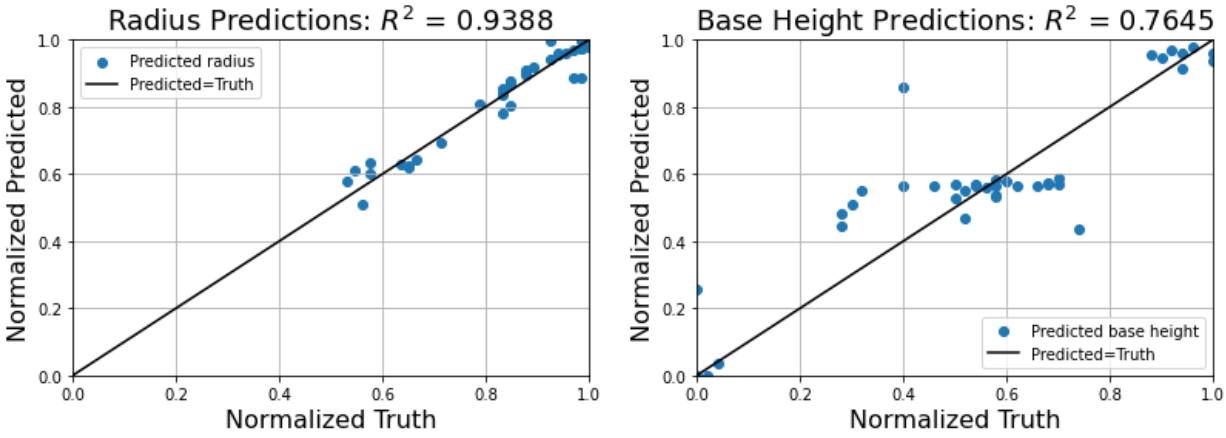


Figure 57: DPN results using the "SiC with Holes" dataset and largest bandgap method.

W with Holes

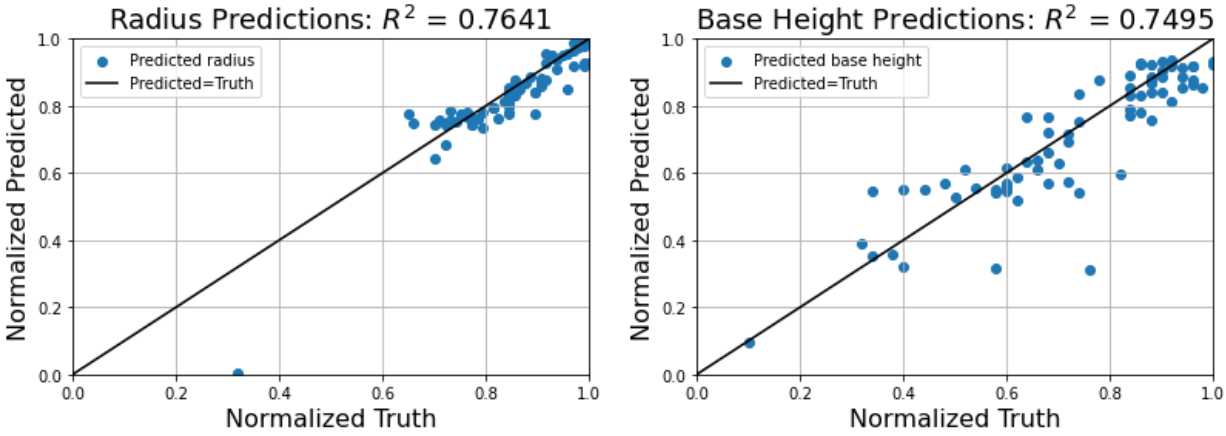


Figure 58: DPN results using the "W with Holes" dataset and largest bandgap method.

Si, SiC, W Combined

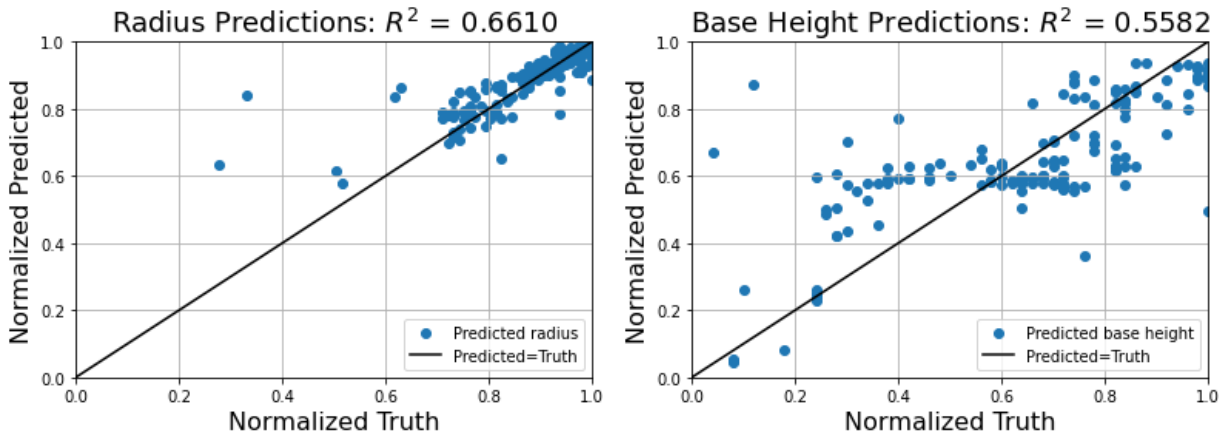


Figure 59: DPN results using the "Si, SiC, W combined" dataset and largest bandgap method.

Since the dataset used for this test contained multiple material types (Si, W, and SiC), the accuracy of material prediction was taken. The figure below shows output of the one-hot encoded material columns output from the trained neural network. The value closest to '1' is taken as the answer.

3	4	5
0	0	0.997293
0	0	0.999576
0	1.00798	0
0	0	0.997984
0	0	0.997843
0	0	1.00011
0	0	0.997535
0	0	0.997549
0	0.988365	0
0.996859	0	0
0	1.00084	0
0.964844	0	0
0	1.00745	0

Figure 60: Example material prediction columns from neural network output. The columns here represent Si, SiC, and W respectively.

For this experiment, the neural network predicted the correct material 92.87% of the time.

From these results we see that for the circular hole datasets, the base height prediction tends to be worse than the radius prediction, which could be due to a higher sampling of radius values as seen in the sweep parameter tables for these datasets in Chapter 4.

Next, the same test is ran using the two datasets containing pillars:

Si with W Pillars

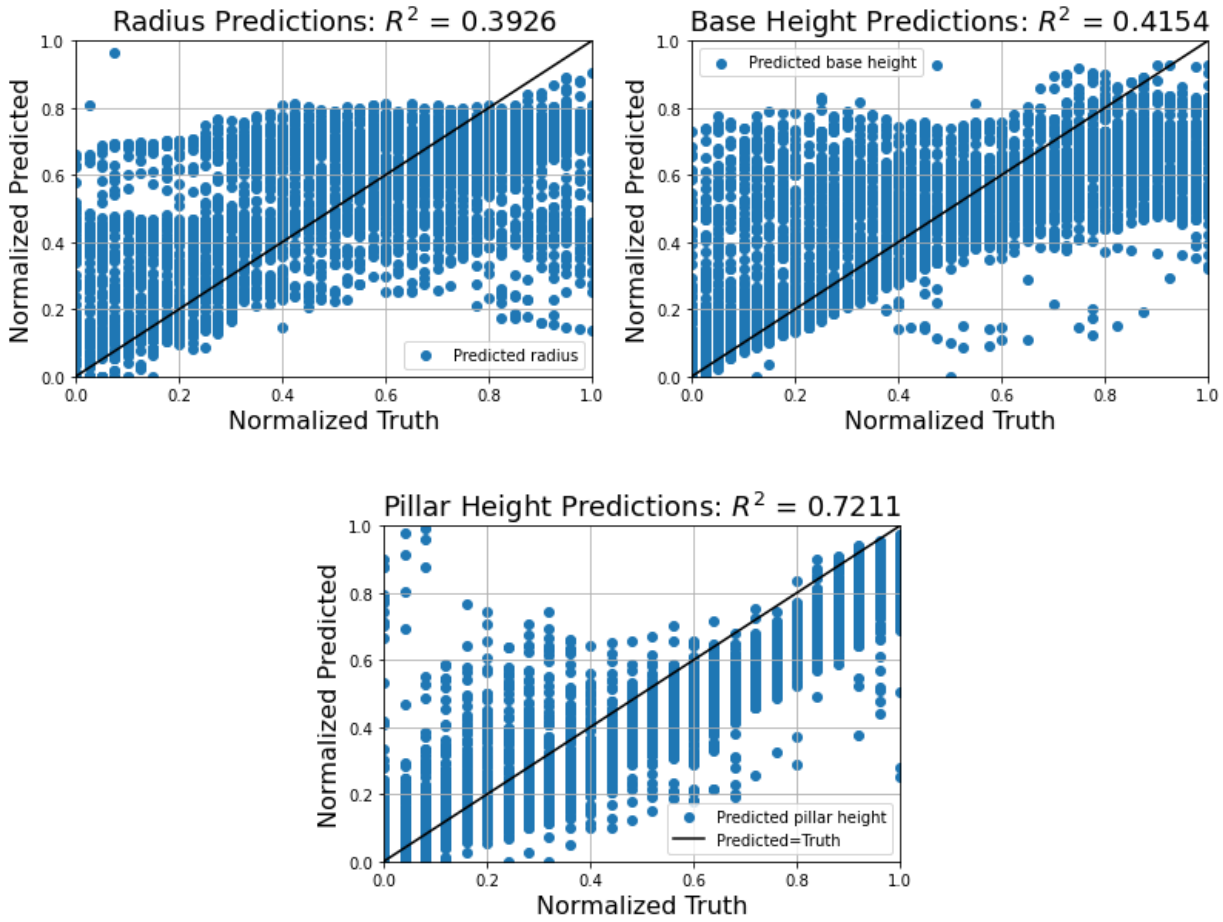


Figure 61: DPN results using the “Si with W pillars” dataset and largest bandgap method.

SiC with W Pillars

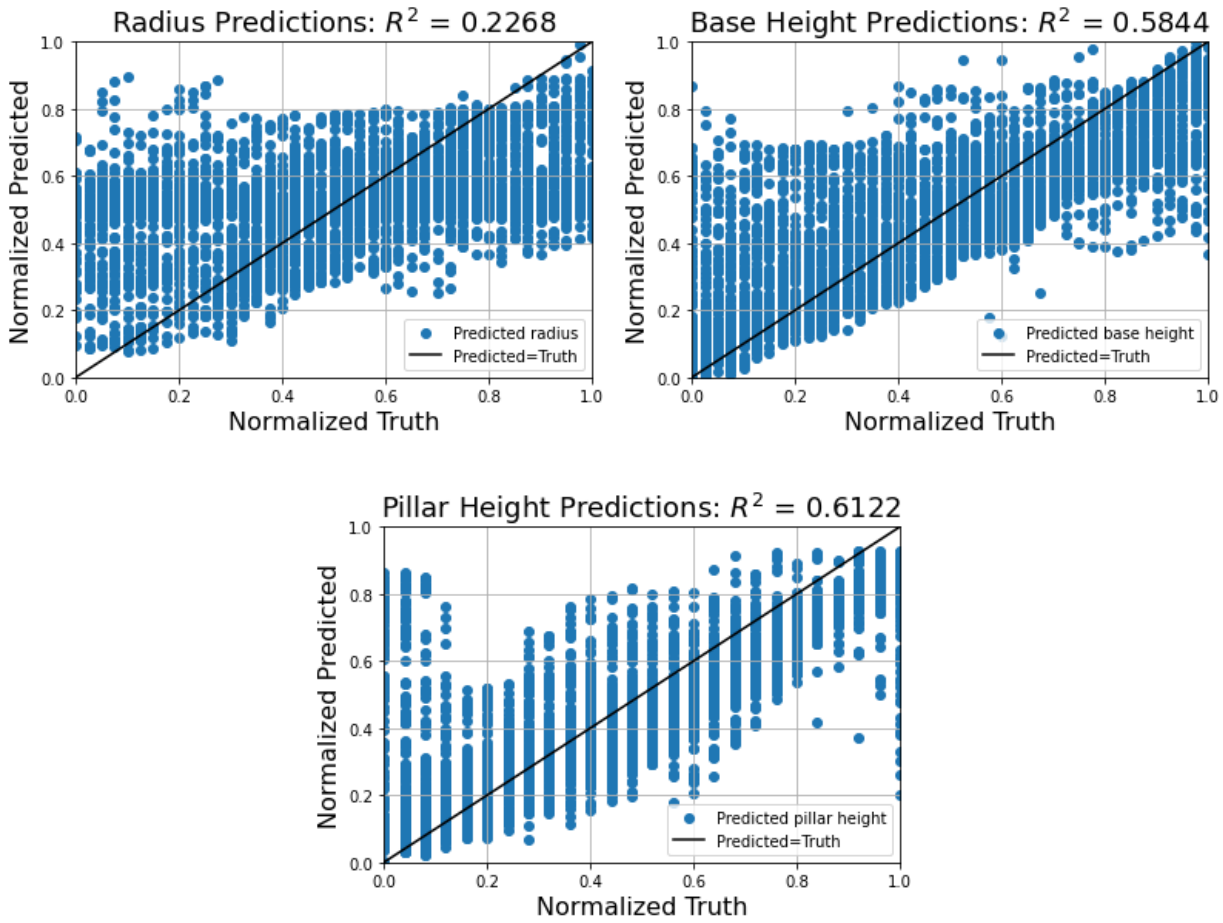


Figure 62: DPN results using the "SiC with W pillars" dataset and largest bandgap method.

It can be seen that the unlike the RPN, the DPN performed worse on the pillar containing datasets than the circular hole containing datasets. Where the RPN benefitted from having a 3rd geometry parameter at the input as extra information to help narrow in on a prediction, the DPN has an extra parameter to predict. From these results we can see that, for the largest bandgap method, having more information at the input of the neural network is generally helpful and having more output parameters to predict generally hurts performance.

From these results, it appears that the largest bandgap may suffice for phononic crystals with lower amounts of geometry features but fails as datasets become more complex with added variable geometry features (as will likely be the trend in future data generation efforts).

5.2.2 Full Band Structure Method

For these tests, the neural network receives band structure diagrams at the input and PnC geometry at the output. In this configuration of neural network, the most amount of non-redundant band structure information possible is supplied to the neural network (we cannot supply any extra information about the phononic band structure than the full band structure diagram itself). It can be seen from the performance plots below that this method generally results in a model with high predictive accuracy.

The figures below show the results from training DPN's with the full band structure method. Note that in these scatterplots, a series of small vertical lines are visible. This is because of the repetitive nature of the truth values in the testing data. By the nature of sweeping over geometry parameters for data generation, multiple different designs are created with the same value of a particular geometry parameter, but different values for the rest of the geometry parameters. The step size between these sweep parameters is also held constant, hence the even spacing between the vertical lines. This trend was not as visible in the DPN largest bandgap method results because only the non-zero bandgap designs were used, and only 10% of those remaining designs were used to test the neural network.

First, results were obtained for the three datasets containing circular holes and the combination dataset containing all three:

Si with Holes

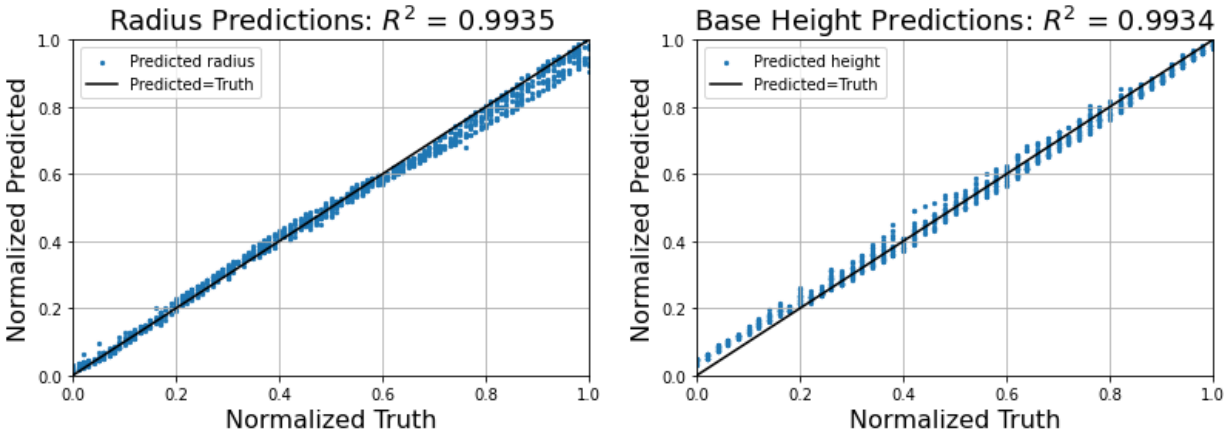


Figure 63: DPN results using the "Si with holes" dataset and full band structure method.

SiC with Holes

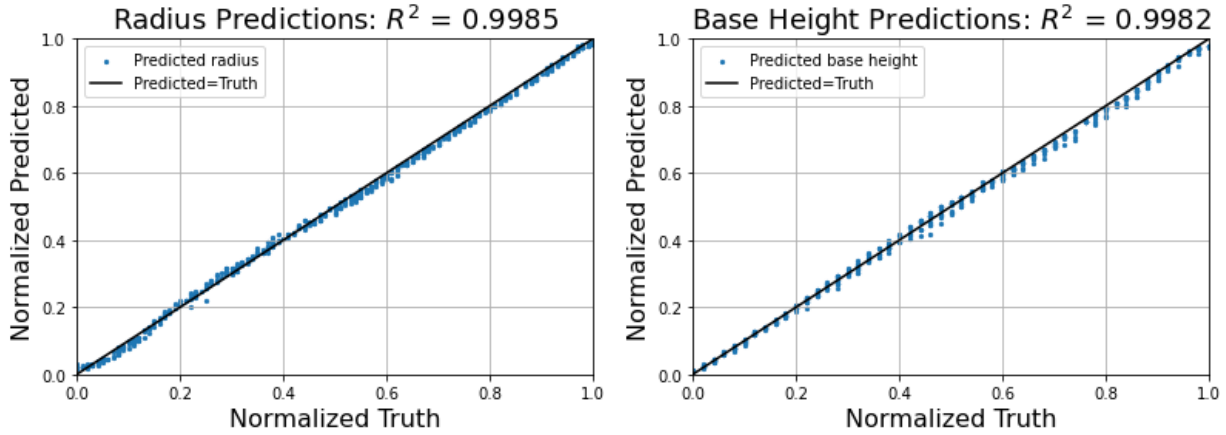


Figure 64: DPN results using the "SiC with holes" dataset and full band structure method.

W with Holes

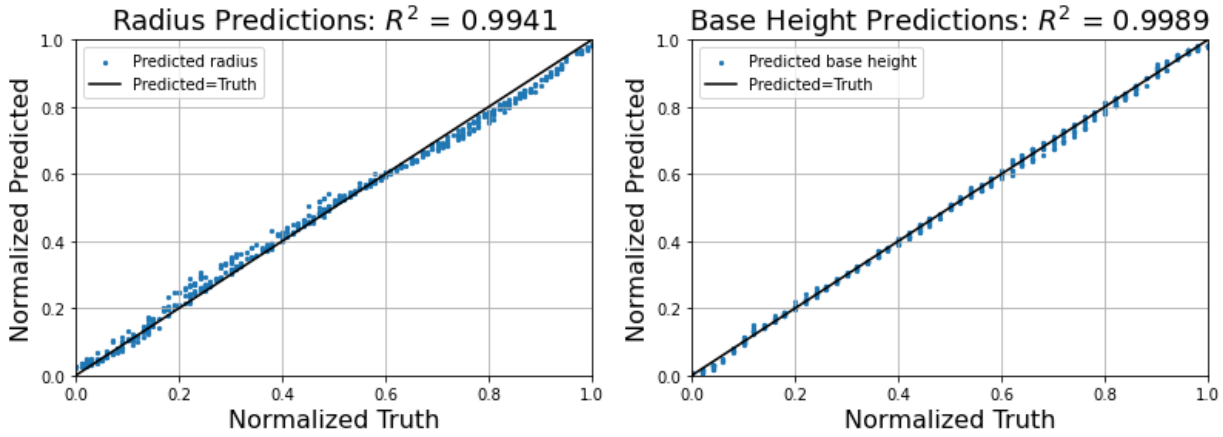


Figure 65: DPN results using the "W with holes" dataset and full band structure method.

Si, SiC, W Combined

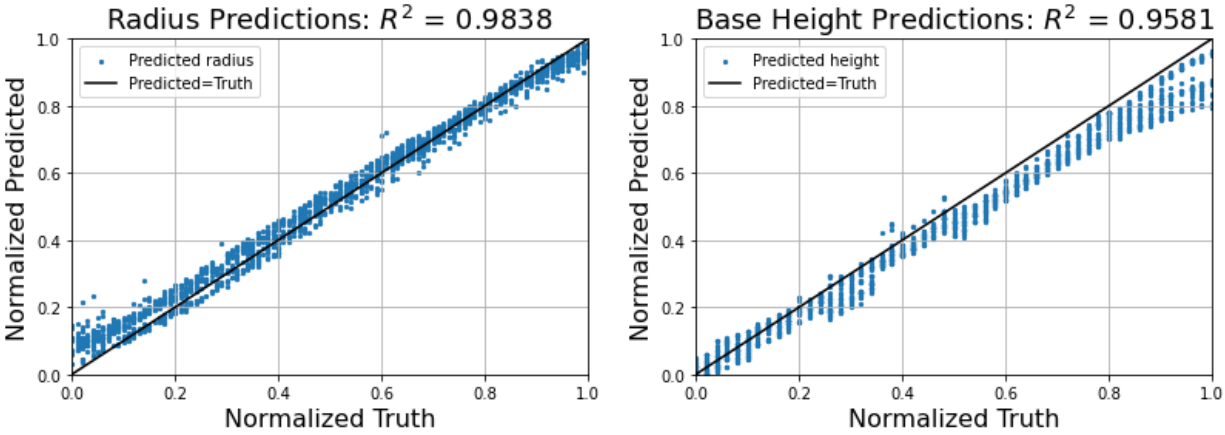


Figure 66: DPN results using the "Si, SiC, W combined" dataset and full band structure method.

As the combination dataset consists of different base materials, the percentage of correct material type predicted was recorded. The percentage correct of predicted material labels was 100%.

We can see that the neural networks perform well in predicting both radius and base height for the datasets containing circular holes. Material prediction for the combined dataset performed perfectly.

The next series of plots shows results for the datasets containing pillars. As the pillar datasets contain a third geometrical parameter, pillar height, the neural network output layer must predict three values.

Si with W Pillars

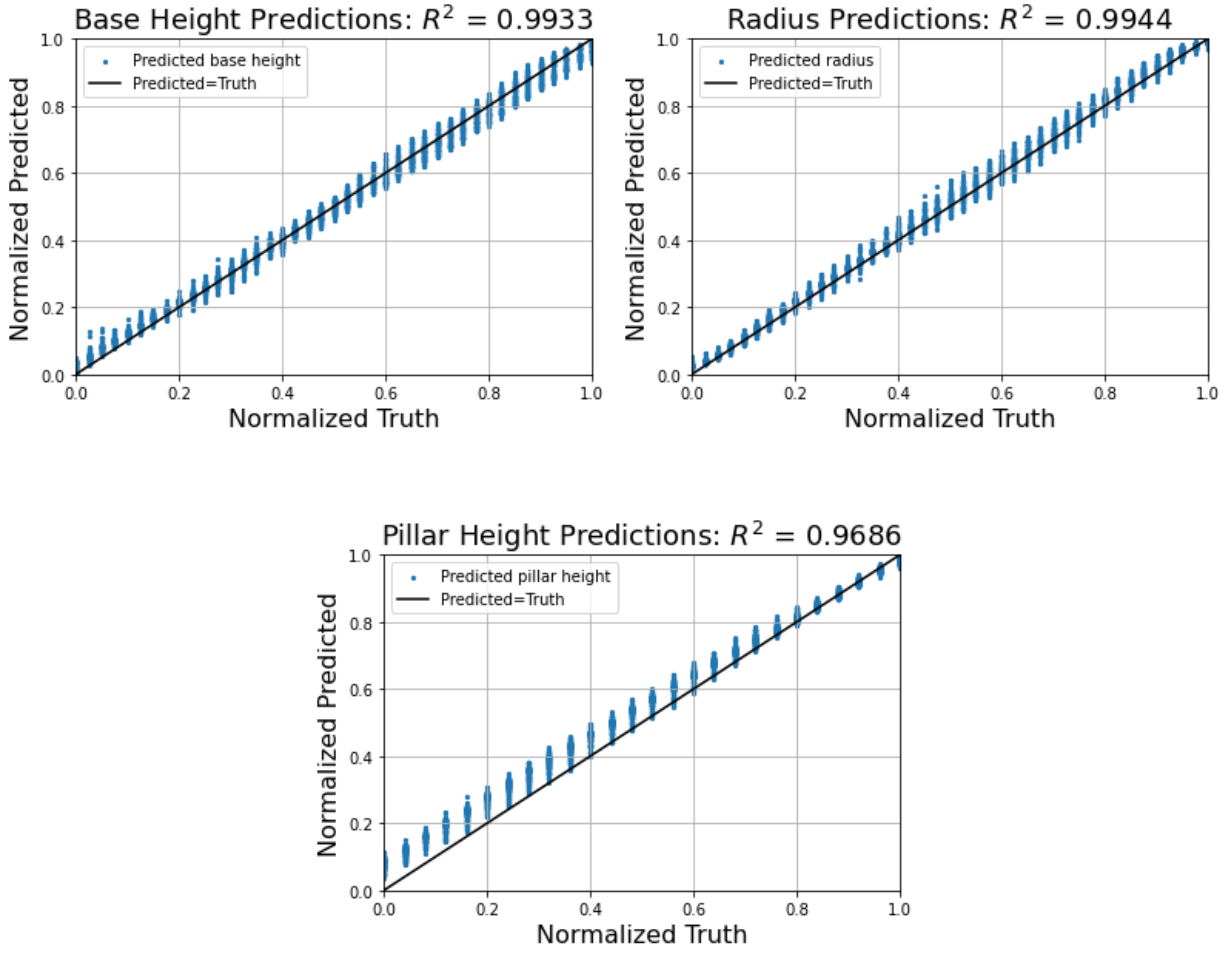


Figure 67: DPN results using the “Si with W pillars” dataset and full band structure method.

SiC with W Pillars

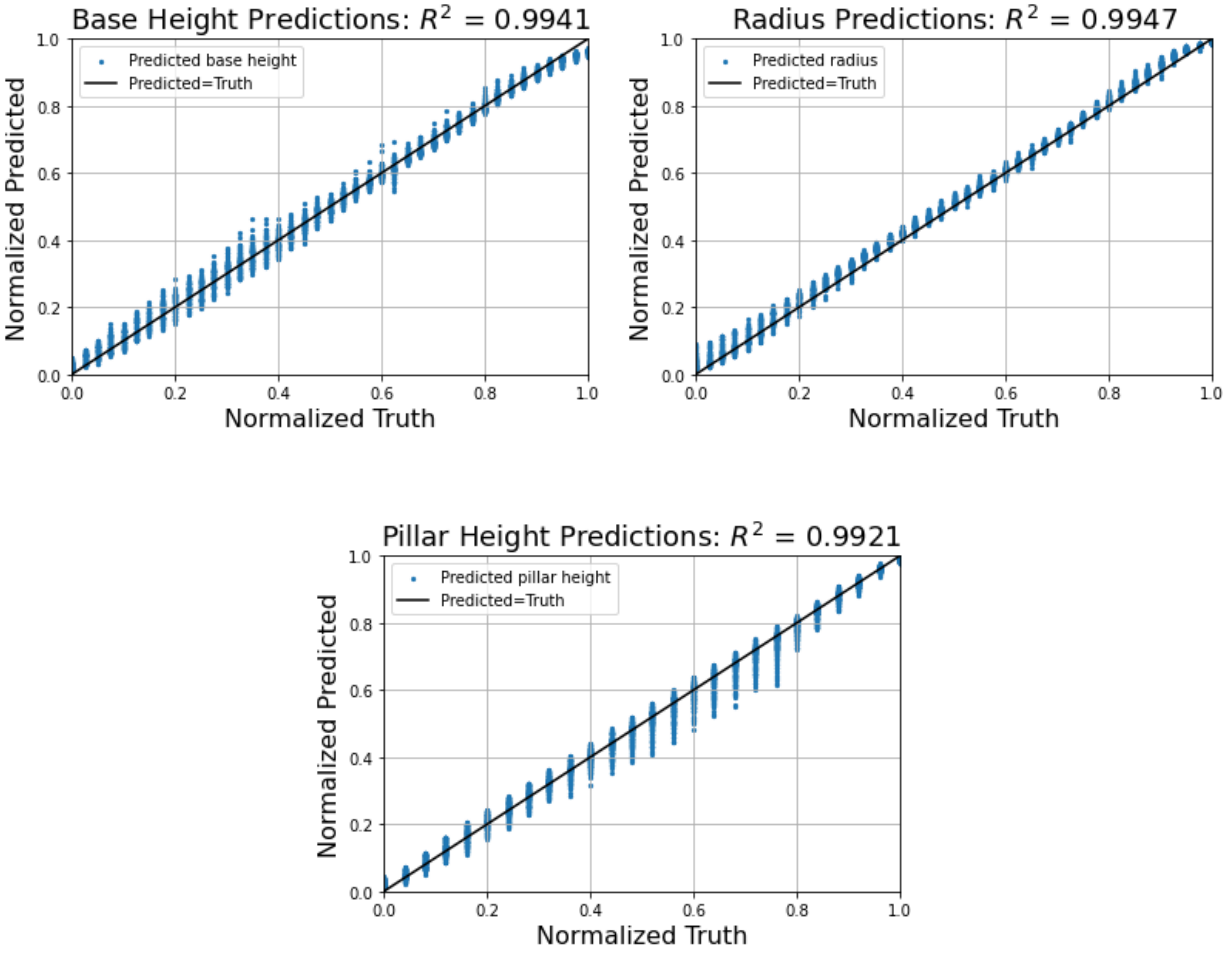


Figure 68: DPN results using the "SiC with W pillars" dataset and full band structure method.

Speaking to the overall results for the DPN full band structure method: though it currently lacks practicality, it does perform well. Most coefficients of determination were close to 1, showing very strong correlation. If, in the future, a method is developed to randomly generate band structures around a given bandgap, the full band structure could be revaluated.

5.2.3 DOS Method

For the final set of DPN results, the DOS method was explored. Given the inconsistent DPN performance using the largest bandgap method (section 5.2.1) and the lack of practicality in using the full band structure method, DOS as an "in-between" method was the logical next-best method to study.

5.2.3.1 Scalar Representation

As explained Chapter 3, the DOS was generate using two representations: scalar DOS and binary DOS. First, the scalar DOS representation is used in tandem with the circular hole datasets:

Si with Holes

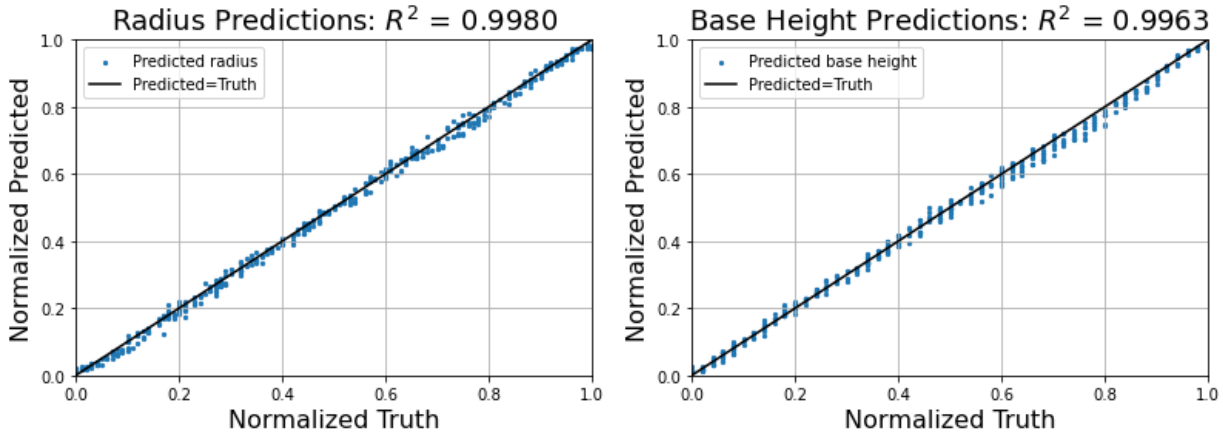


Figure 69: DPN results using the "Si with holes" dataset and scalar DOS method.

SiC with Holes

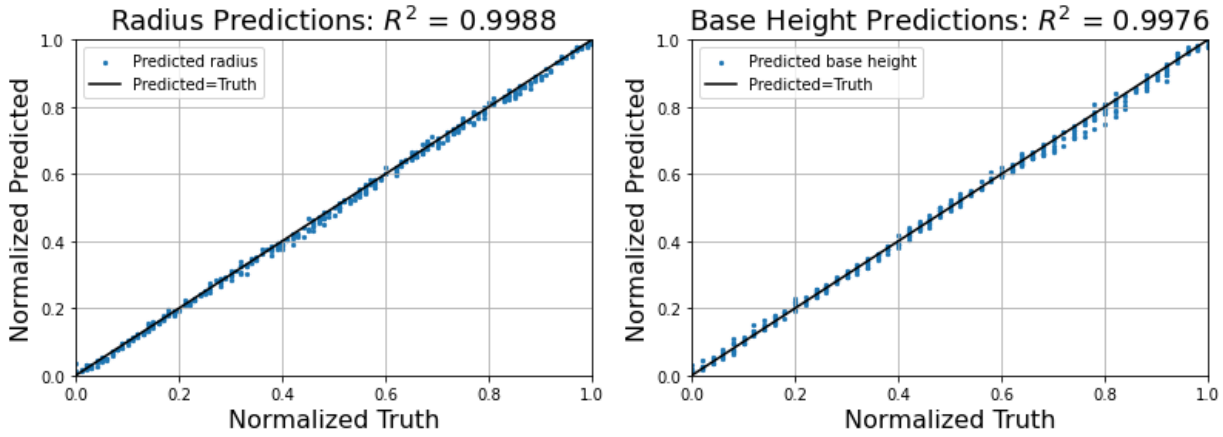


Figure 70: DPN results using the "SiC with holes" dataset and scalar DOS method.

W with Holes

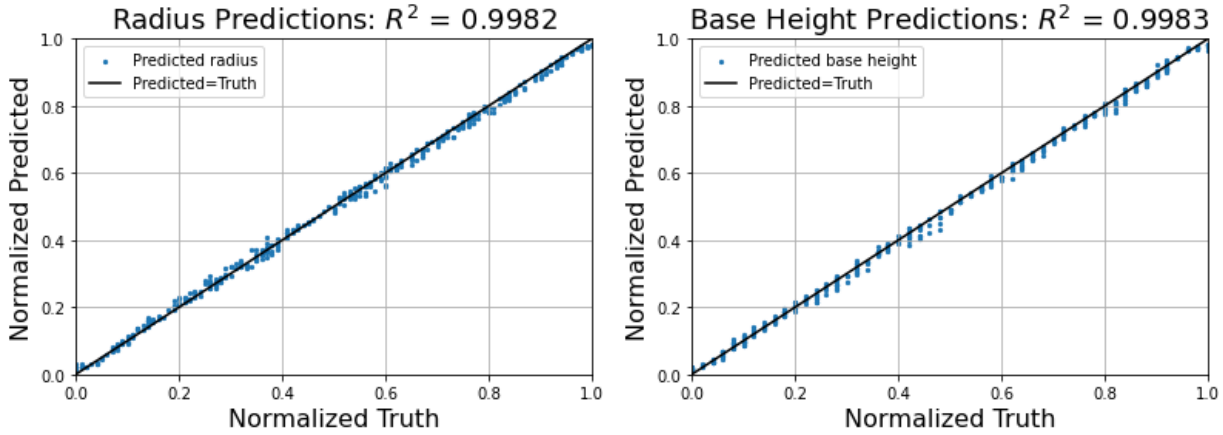


Figure 71: DPN results using the "W with holes" dataset and scalar DOS method.

Next, the pillar containing datasets are explored using the same method. Again, each set of results contains three predicted parameters, as the pillar datasets contain three variable parameters.

Si with W Pillars

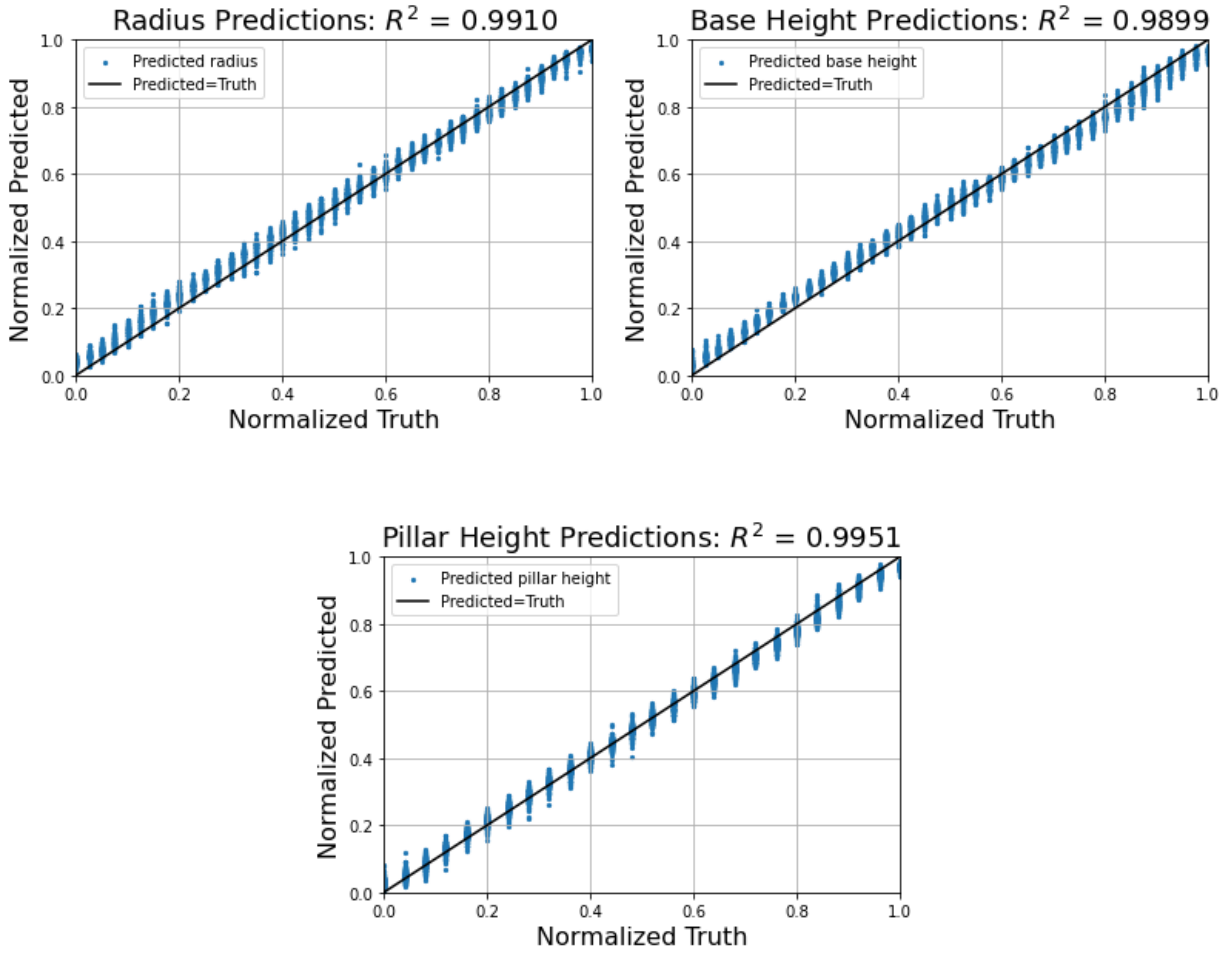


Figure 72: DPN results using the “Si with W pillars” dataset and scalar DOS method.

SiC with W Pillars

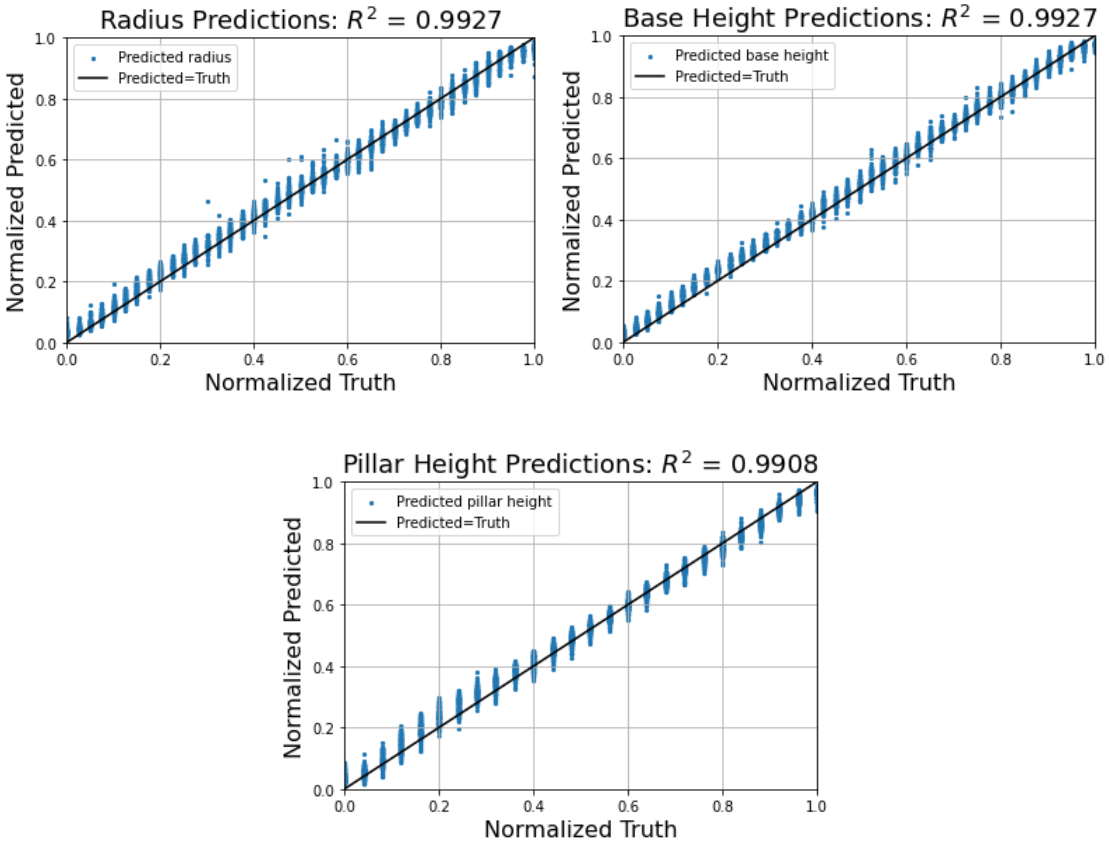


Figure 73: DPN results using the "SiC with W pillars" dataset and scalar DOS method.

Unintuitively, the scalar DOS method showed on par, and in some cases better, correlation to the full band structure method. According to these results, although the full band structure method supplies more information, that extra information is redundant. This shows that the scalar DOS method is more practical than the full band structure method in the sense that scalar DOS takes less time to process but offers the same predictive performance. It is possible however, that this could change when training on more complex training data that contains more geometry parameters to predict.

5.2.3.2 Binary Representation

Though the density of states representation has more value than the full band structure representation for use in a DPN, the challenge of generating data for the non-bandgap regions is still non-trivial. Because of this, another method of representing the DOS was conceived: the binary representation. The binary representation of DOS is identical to the scalar representation, except that every value in the array that is greater than zero is changed to one. This way, the DOS array becomes a square wave, as can be seen in section 3.4.3

The circular hole datasets showed results similar to those of the largest bandgap method before excluding the zero bandgap results. This is likely due to the nature of the binary DOS. The binary DOS will appear as an array of all ones to the neural net. Since most of the designs in this dataset do not contain bandgaps, the neural network is unable to draw patterns between the DOS and geometry. The figure below shows radius prediction results from leaving the zero bandgap designs in the training set:

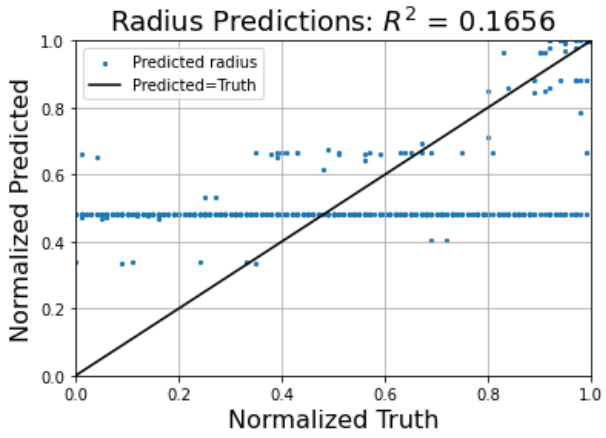


Figure 74: Results for binary DOS DPN using Si with holes dataset, not excluding zero bandgap data in the training set.

Removing the zero bandgap designs then training again, a clear increase in performance can be seen:

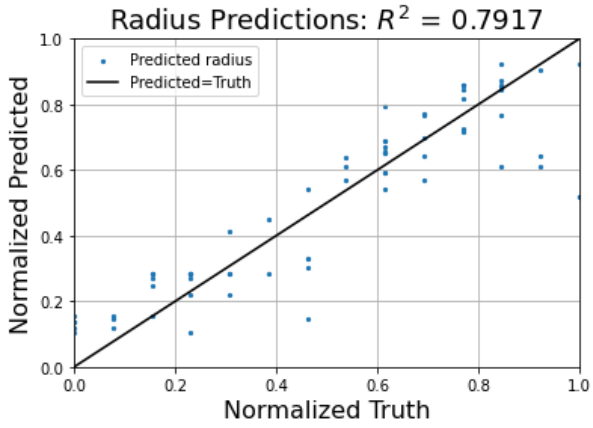


Figure 75: Results for binary DOS DPN using Si with holes dataset excluding zero bandgap data in the training set.

These results are characteristic of all the circular hole datasets. Next, binary DOS was tested using the two pillar-containing datasets. The larger bandgap method has shown to be viable for the circular hole datasets, The two sets of figures below show the results.

Si with W Pillars

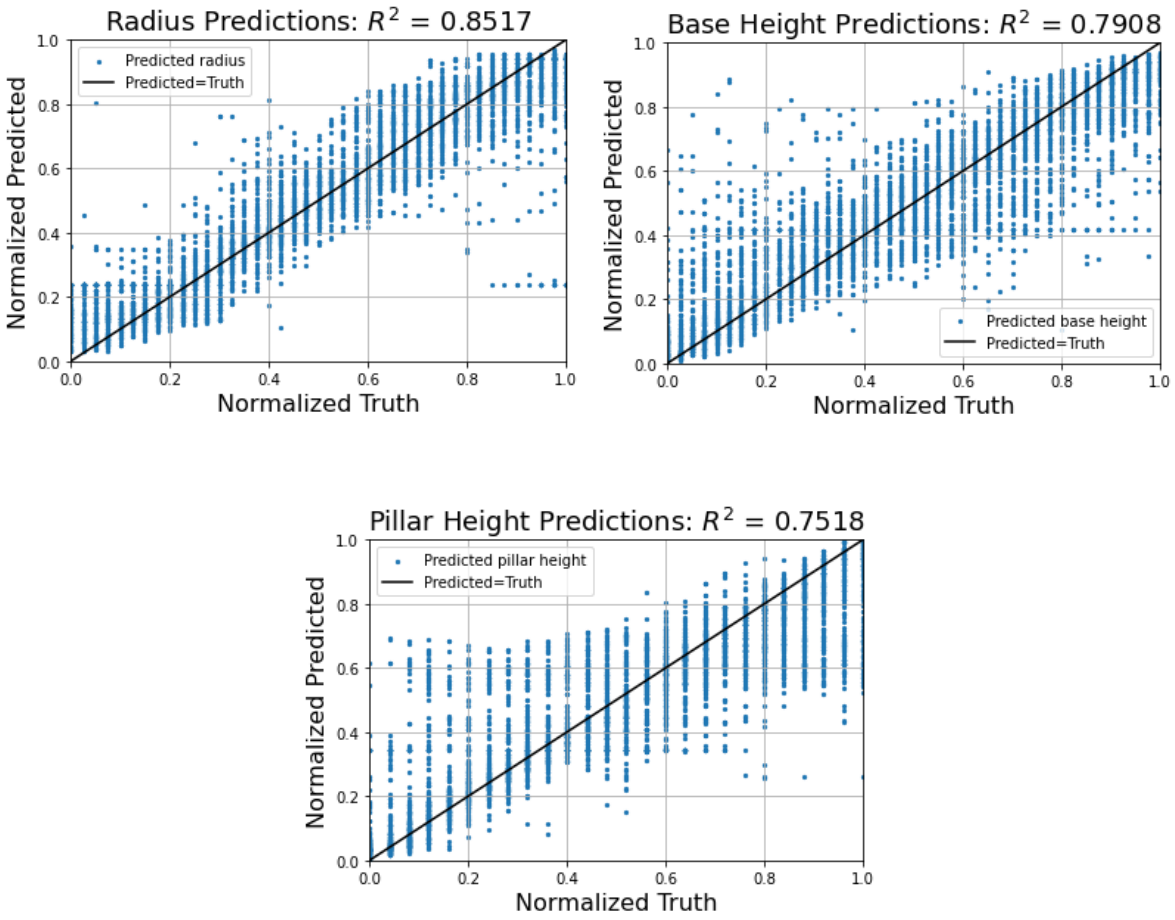


Figure 76: DPN results using the SiC with W pillars dataset and binary DOS method.

SiC with W Pillars

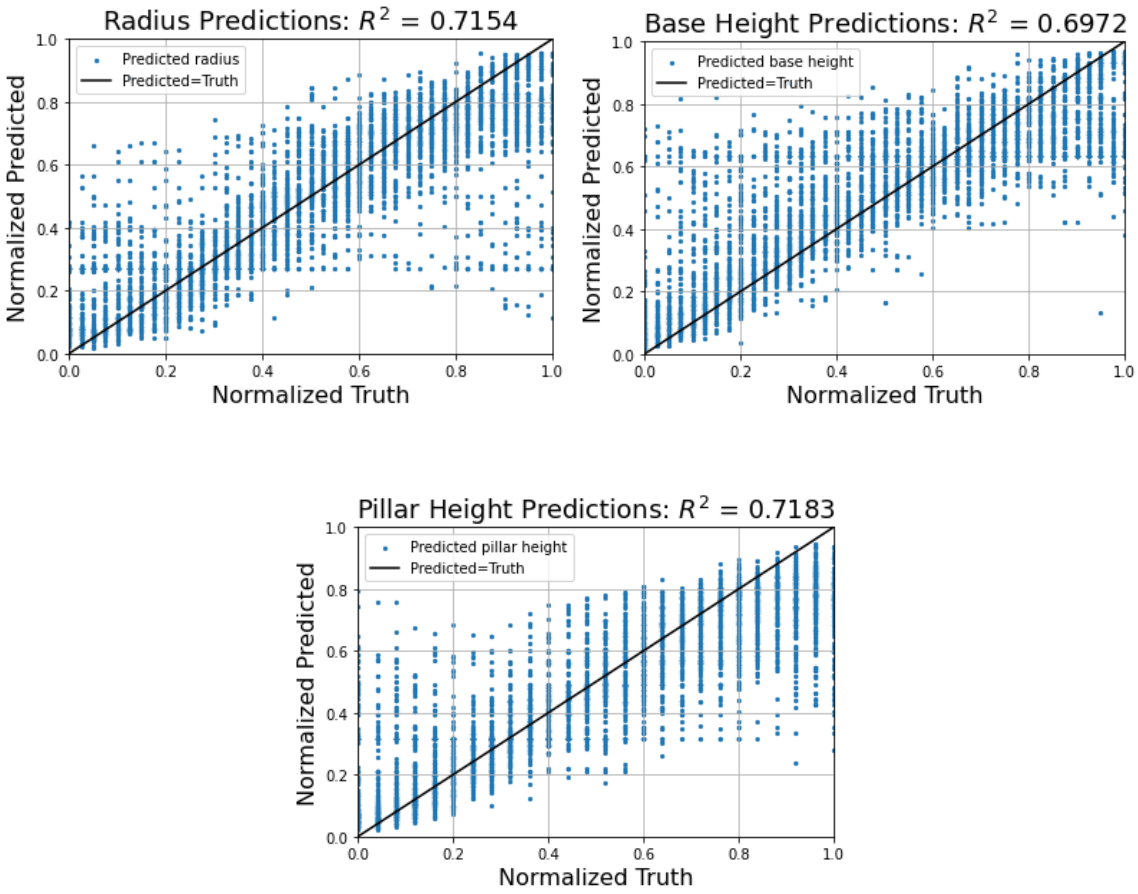


Figure 77: DPN results using the SiC with W pillars dataset and binary DOS method.

It can be seen that the binary DOS representation sees a significant hit in performance as compared to the scalar representation. This is expected due to the loss in information being fed to the neural network. Though worse than the scalar representation in terms of performance, we do see a performance increase over the largest bandgap method. Since it is fairly intuitive to use the binary representation of DOS (create an array with 0's in the region of the desired bandgap and 1's everywhere else), the next section describes a prediction scheme implemented to use this method for crystal inverse design.

5.2.3.3 Binary Representation Prediction Scheme

To use the DOS method to perform inverse crystal design, a prediction scheme using both a DPN to make predictions and an RPN to check the predictions was implemented. The scheme is summarized in the figure below:

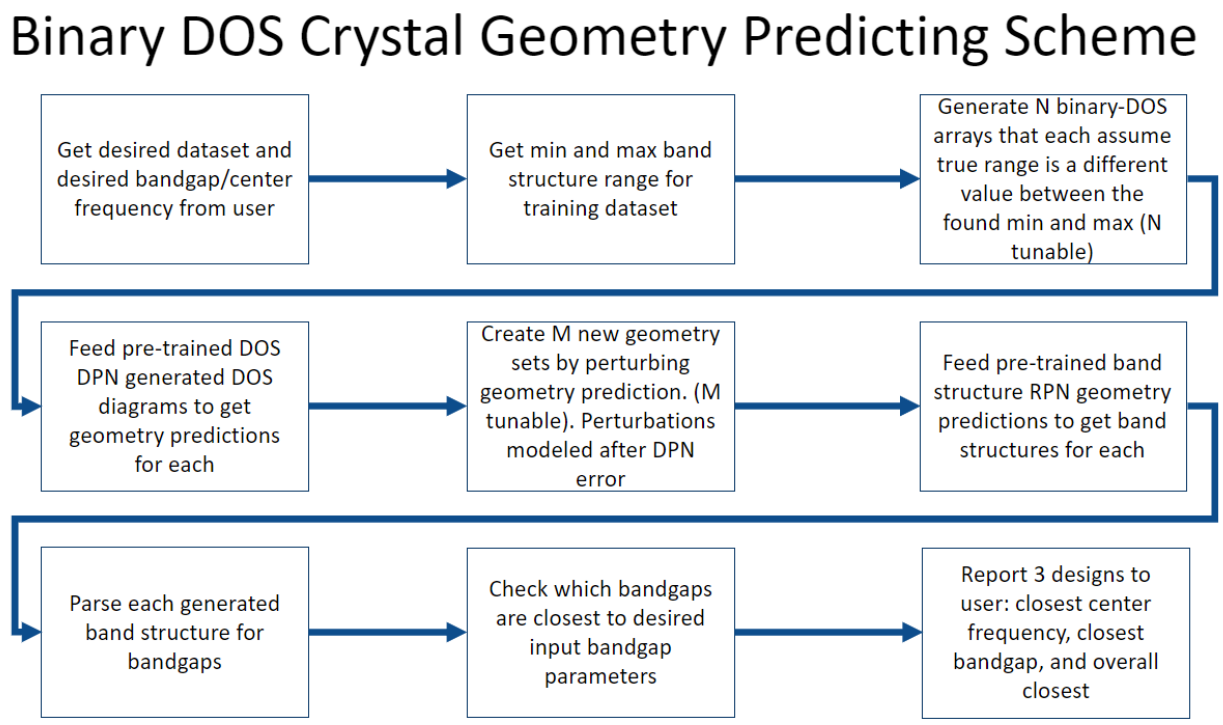


Figure 78: Flowchart demonstrating the binary DOS inverse design scheme developed for this study.

First, the desired bandgap and center frequency values are obtained from the user. Next, the minimum and maximum band structure frequency range values (range = max frequency – min frequency) are taken from the training dataset used to make predictions. In section 5.1.3, multiple histograms show that the band structure range values vary wildly within the same training dataset. This must be considered when deciding where to place the zeroes (bandgap region) in the DOS of

state. The figure below visualizes this problem. The figure shows how generating two binary DOS each representing the same bandgap size and center frequency can result in different arrays.

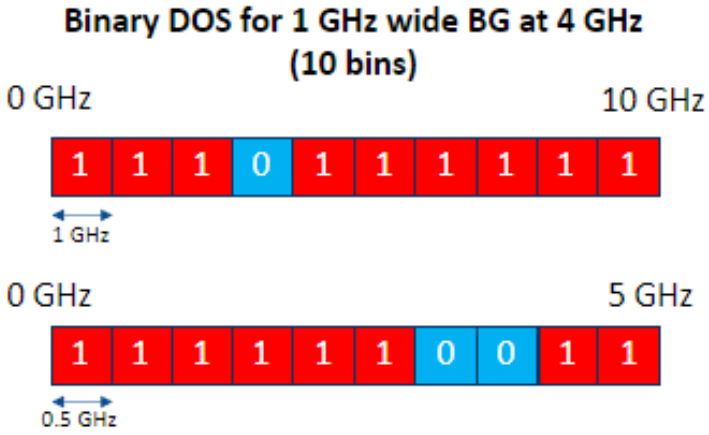


Figure 79: Given the user wants to find a design with a bandgap of 1000 MHz centered at 4000 MHz, the DOS can look different depending on the assumed band structure frequency range. Assuming 10,000 MHz range, and using a 10-point DOS, each bin in the DOS represents 1000 MHz of frequency space. Assuming 5,000 MHz however, each bin represents 500 MHz of frequency space.

To mitigate this problem, 'N' DOS are generated, each assuming a range value from a list spanning N linearly spaced values between the minimum and maximum range of the training dataset. N is a parameter specifiable by the user. The N binary DOS arrays are then sent to a binary DOS-trained DPN to make geometry value predictions. N sets of geometry predictions are generated.

These geometry predictions are then perturbed to create new geometry sets. The idea is to mitigate the variance in the DPN's predictions (see DPN binary DOS figures) by generating multiple geometry sets whose values are slightly altered from the values the DPN initially predicted. The perturbations are modeled using the error from the performance scatterplots. For

each geometry parameter, the standard deviation of the errors from the test set is used to generate a Gaussian distribution. For each perturbed set generated, each geometry parameter predicted by the DPN is changed by the amount pulled from that geometry parameter's generated Gaussian error function.

These geometry sets are then sent to a full band structure trained RPN which generates a band structure for each set of geometry parameters generated by the DPN. As seen in section 5.1.3, the band structure trained RPN is shown to predict accurate band structure values reliably. The band structures are then parsed for band gaps. These bandgaps are compared back to the original bandgap and center frequency value the user input and three potential geometry prediction candidates are highlighted: the one that produced the closest center frequency, the one that produced the closest bandgap, and the one that was "overall" closest. Overall closest here is defined as the set of geometry predictions that minimizes the sum of squared error:

$$\min\left[(predicted\ BG_i - desired\ BG_i)^2 + (predicted\ CF_i - desired\ CF_i)^2\right]$$

where BG is bandgap and CF is center frequency. Sum of squared error is used because CF and BG are often on different orders of magnitude. Squaring helps to normalize the errors and prevent one parameter from dominating the calculation. The following is an example output from the scheme. The queried bandgap was 500 MHz and the queried center frequency was 2000 MHz.

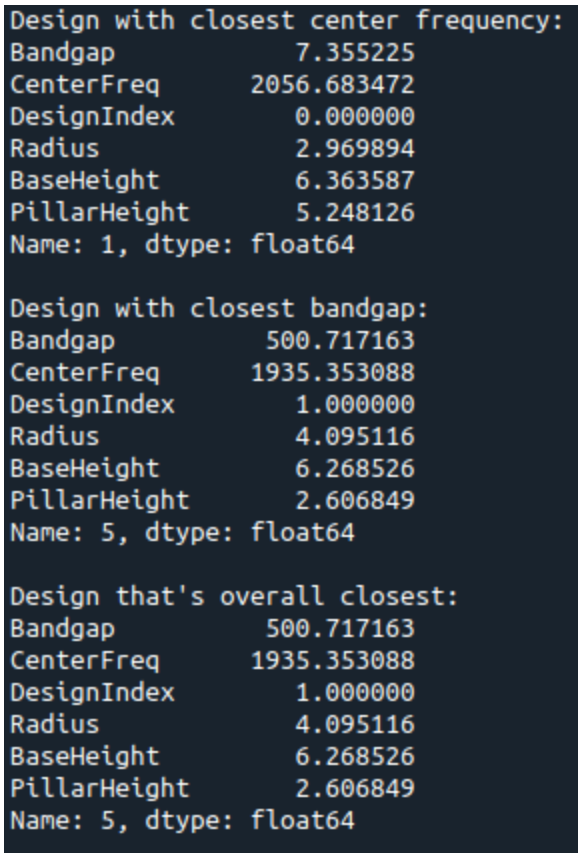


Figure 80: Best matches found according to three categories: closest center frequency, closest bandgap, and closest overall.

Using the metric defined above, the overall closest here happened to be same that scored the highest in center frequency prediction. Another design was found with a slightly closer center frequency bandgap, but significantly smaller bandgap size. The rest of the found designs are saved in a dataframe and viewable by the user. The band structure diagrams output from the RPN are also viewable. The band structure corresponding to the “overall best” result from the previous figure is shown in in the figure below.

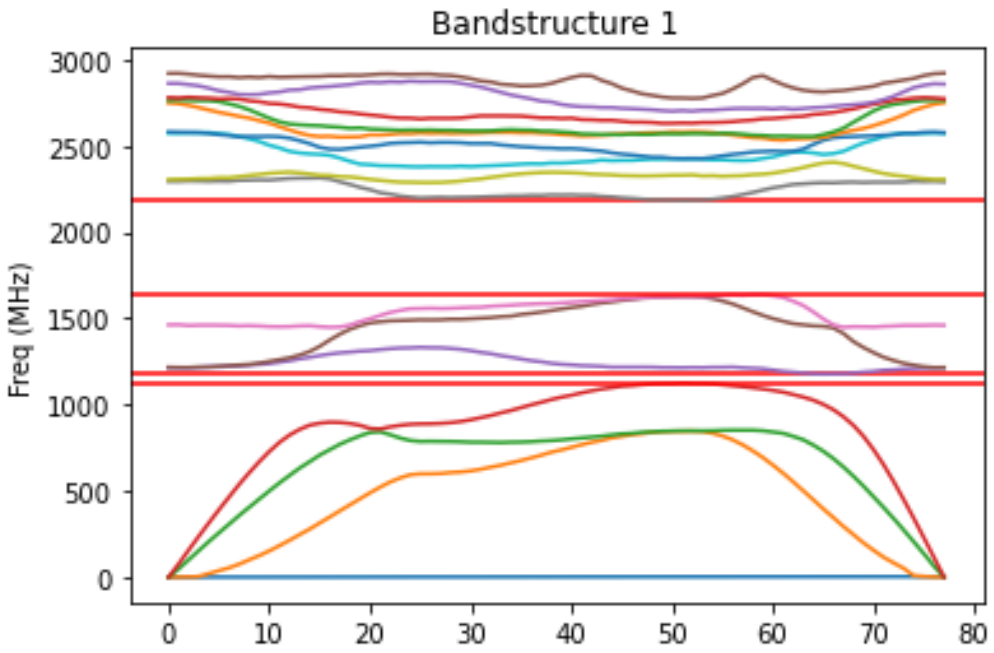


Figure 81: Band structure corresponding to `DesignIndex` 1 in the previous figure. The red horizontal lines correspond to bandgap regions. The upper two red lines contain the “closest” bandgap seen in the previous figure. Structure was generated by a band structure trained RPN given geometry parameters predicted by the DOS DPN.

In order to characterize the performance of this scheme, bandgap/center frequency pairs were taken from the testing set and given to the binary DOS inverse design predictor. Specifically, the test set was split into two new sets: one that models the error for calculating the perturbations, and another whose bandgap/center frequency pairs were used for performance characterization. To avoid biasing the outcome, it was important to not use bandgaps from the fraction of the test set that was used to generate the Gaussian error functions. In general, any data that effects the performance of the system should not also be used to calculate performance.

Three sets of plots were generated: bandgap and center frequency predictions of designs minimizing bandgap error, bandgap and center frequency predictions of designs minimizing center

frequency error, and finally bandgap and center frequency predictions of the "overall" closest (designs attempting to minimize both bandgap and center frequency). 20 binary DOS were generated per requested bandgap/center frequency pair, with 50 sets of DPN perturbations each. The results are shown in the figure below:

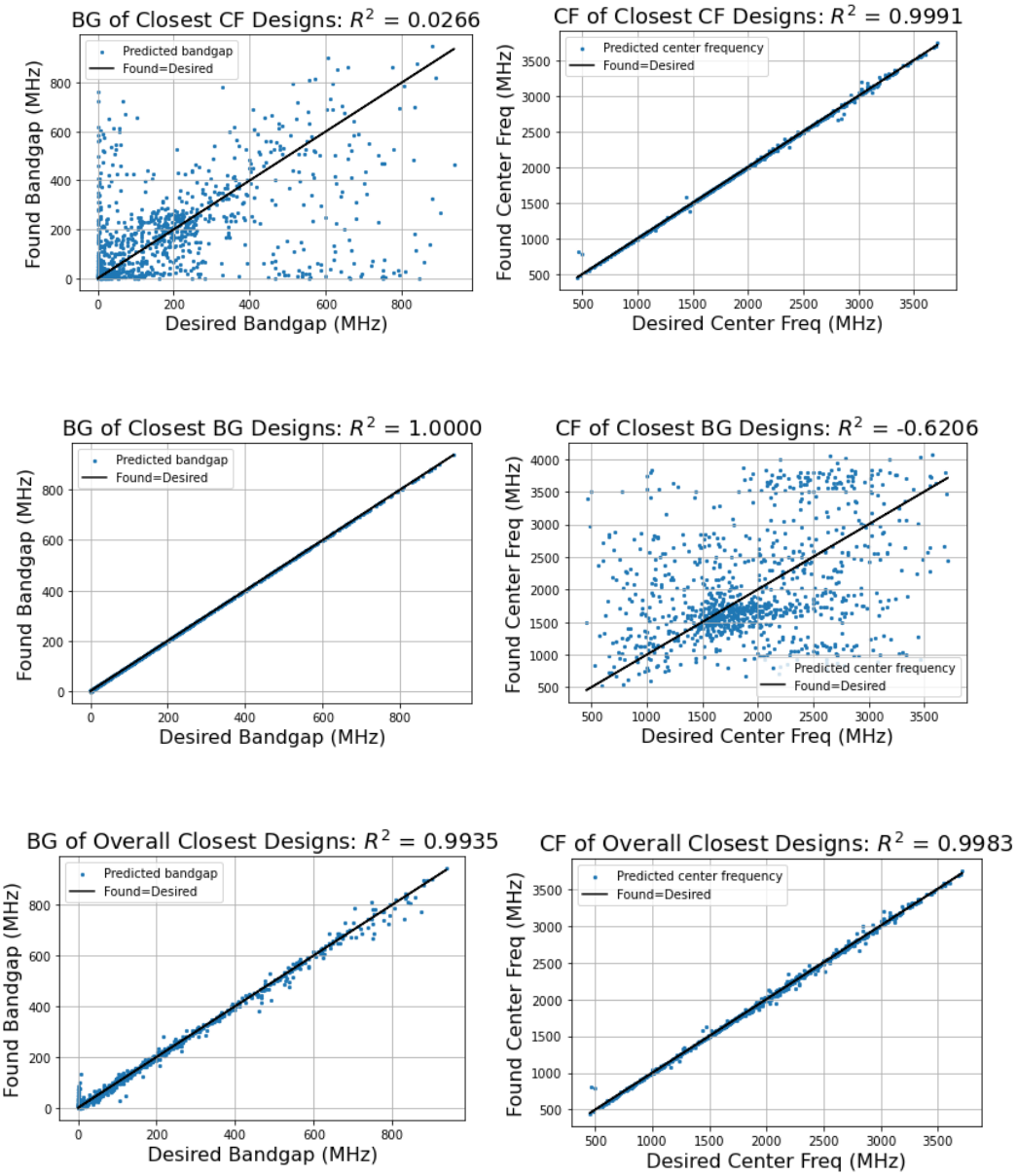


Figure 82: Binary DOS inverse design performance characterization results for designs minimizing center frequency error (top), designs minimizing bandgap error (middle), and designs minimizing both (bottom).

We can see that the designs corresponding to closest bandgap and closest center frequency minimize their corresponding parameter extremely accurately, while disregarding accuracy of the other parameter. We also see high correlation in both bandgap and center frequency predictions for the overall closest designs. From these results it is apparent that the binary DOS inverse prediction scheme shows strong predictive correlation both in situations where preciseness in one parameter is required without regard for the other parameter and in situations where both parameters are required to be as accurate as possible.

For the final test of the binary DOS inverse design scheme, the input bandgap and center frequency values were altered randomly by either plus or minus (50/50 chance) 10 percent before predictions were made. The results are shown in the figure below.

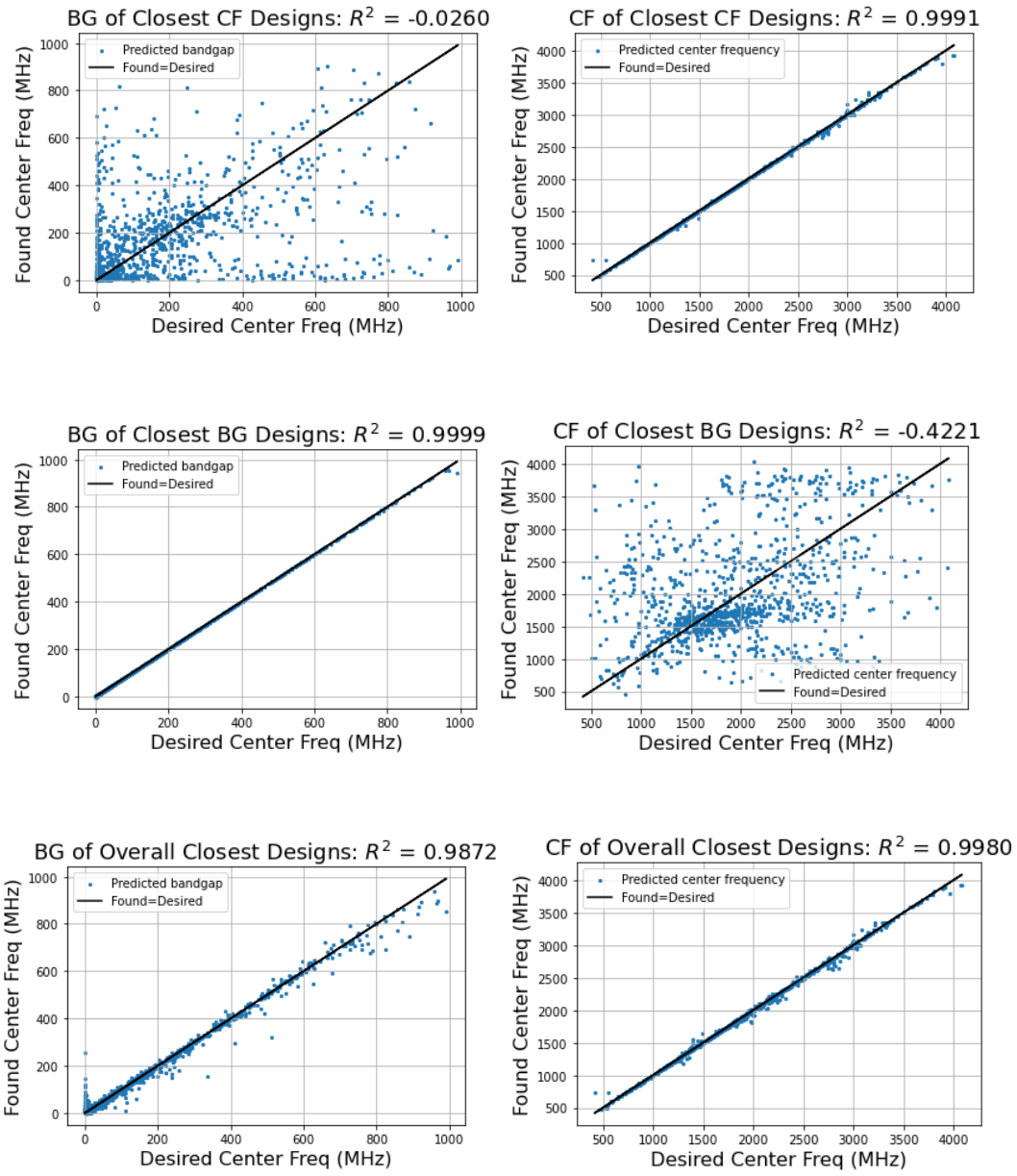


Figure 83: Binary DOS inverse design performance characterization results in which the input bandgap and center frequency values are altered by (+/-) 10%. Results shown for designs minimizing center frequency error (top), designs minimizing bandgap error (middle), and designs minimizing both (bottom).

Even though the values are altered slightly from the values known to exist from the testing set, the inverse design scheme was still able to find predictions closely matching the desired values.

Though these results show the scheme's robustness to finding geometry for bandgaps outside of what we know exists in the testing set, it is important to remember that these results are verified by an RPN, not by simulation. As shown in this paper, the RPN is shown to produce accurate band structures, but the next step for validation of accurate predictions would be to simulate the neural network PnC design predictions using COMSOL and verify that the reported designs actually result in the reported bandgap and center frequency values.

5.3 Training Data Split Test

When generating data for a neural network, it is desirable to need to generate as little data as possible. At a minimum, we want enough samples for the neural network to be able to accurately capture the model, but not too much data as to oversample as generating data takes long amounts of time. If the design space is oversampled, then the neural network's usefulness decreases as we see less "empty" regions in design space for the neural network to interpolate predictions from. To help gain an understanding for how much data we may need to generate for future data generation efforts, a tool was developed to plot the effect of changing the training data split ratios.

When training a neural network, the training data is split into three groups over the course of two different splits. During the first split, the "test" set is split from the rest of the training data. The test set is used to evaluate neural network performance after training and does not influence the training process. The remaining data is split between the "training" and "validation" sets. These two sets are the what the neural network uses to reduce its loss function during training. For these experiments, the ratios of these three groups were varied and their effects on the training process and prediction performance were noted.

Since the training process does not always converge to the lowest value (presumably this would occur less and less often with more training data) a target of 0.01 MSE was set for the neural networks to reach every iteration of split ratio. This value was picked because it was observed that performance above this value generally resulted in weak prediction performance. If the final loss value during training did not reach this threshold, then the neural network would be retrained until the loss value was reached. The number of attempts taken to reach this low loss value at each split ratio was recorded and included in the resulting plots.

Both of the following "split" tests were performed using the Si with holes dataset (containing 5151 values) formatted using the full band structure method.

5.3.1 Training-Validation Split

For this experiment, the "test" set is fixed to 10% while the ratio of the "training" to "validation" sets are varied. The x-axis of the plots below represents the percentage of data reserved for the validation set, ranging from 1% to 99%, where 1 – (percent reserved for validation) = percent reserved for training.

The plots below show the coefficient of determination scores after each Monte Carlo iteration. The scores remained in the high 90's until about 95% percent reserved for validation. After this point the prediction performance continues to degrade.

An inverse relationship to the coefficient of determination plots can be seen in "Split Percent vs Final Training Loss." In this plot, the final training loss value was taken for the attempt that reached lower than 0.01 loss. The final loss value steadily raises throughout the experiment,

until around the same iteration in which the prediction performance starts to degrade. At this point, the rate of increase in loss sharply increases.

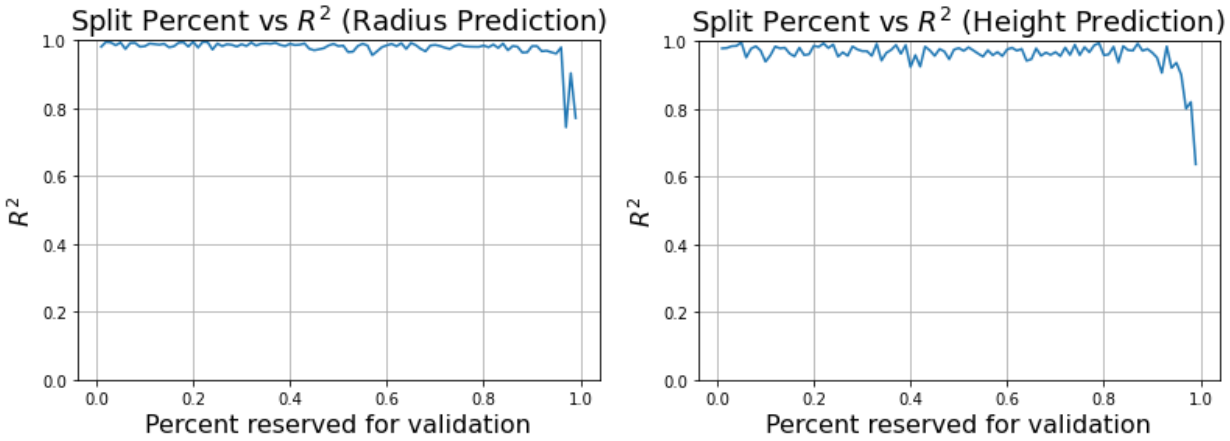


Figure 84: Radius (left) and base height (right) prediction performance for each value of percent data reserved for validation.

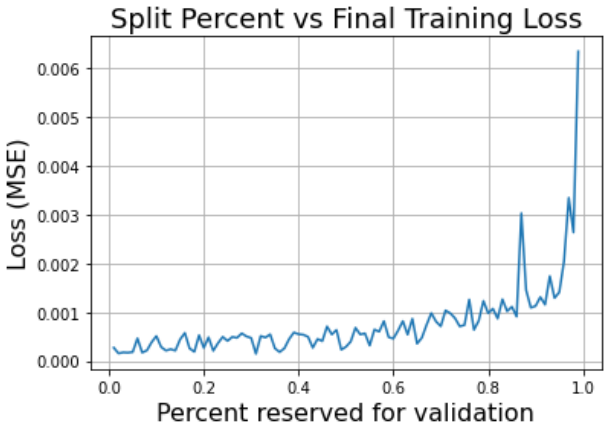


Figure 85: Final loss value reached at the end of training as a function of the percentage of data reserved for validation.

It was observed that the number of attempts needed to reach 0.01 loss was seemingly random. Anywhere between 1 to 6 attempts were needed to reach a low loss value. This was

unexpected as it was assumed that the neural network would take more attempts on average the less data it had access to for training. This shows that the tendency of the neural network to take multiple training attempts before a low loss is reached is affected by some factor that remained constant in this Monte Carlo experiment. It is likely that the low amount of training data for this dataset is a contributing factor.

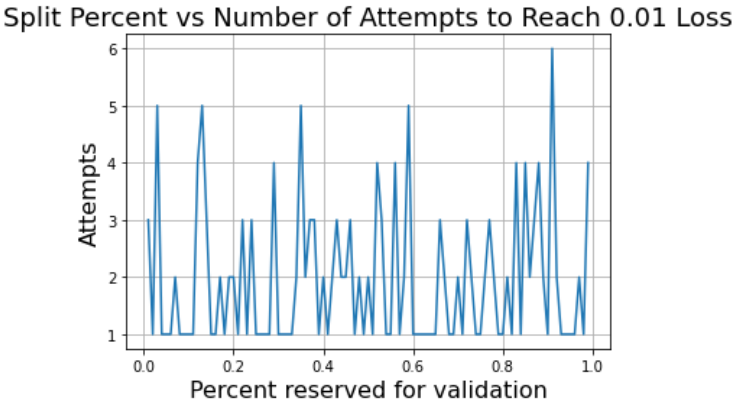


Figure 86: Number of attempts taken to reach the threshold loss value of 0.01 as a function of the percentage of data reserved for validation.

5.3.2 Training-Test Split

This experiment was conducted in the same manner as the previous experiment, except percent reserved for validation was fixed at 10%, while the amount reserved for test was varied. The results in the figures below show very similar patterns as seen in the training-validation split.

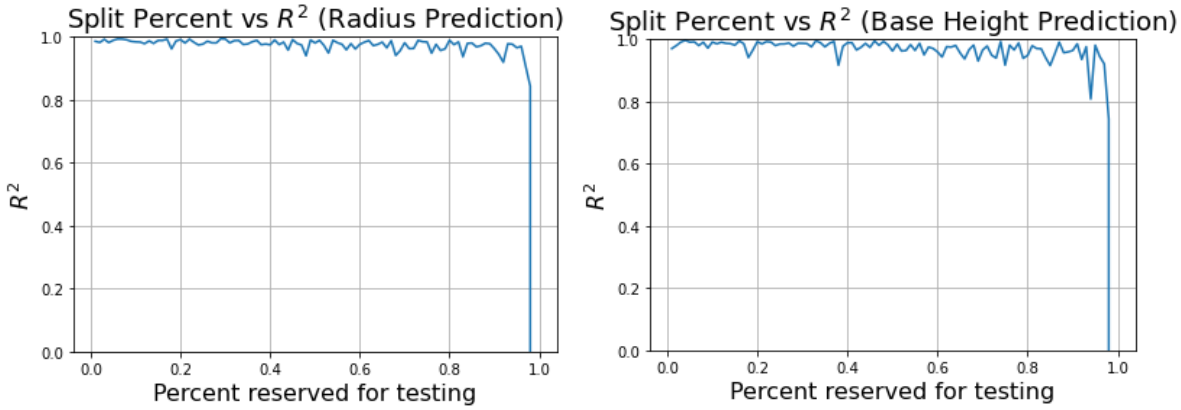


Figure 87: Radius (left) and base height (right) prediction performance for each value of percent data reserved for testing.

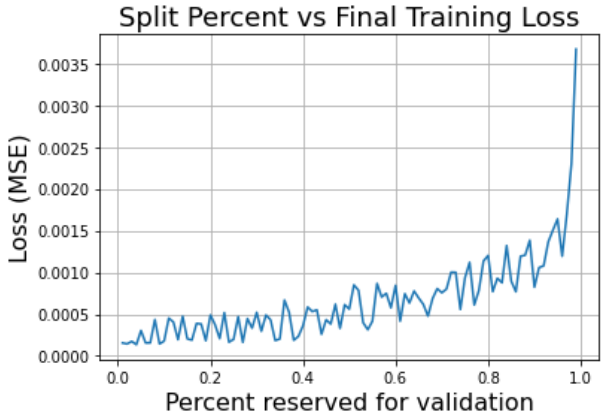


Figure 88: Final loss value reached at the end of training as a function of the percentage of data reserved for testing.

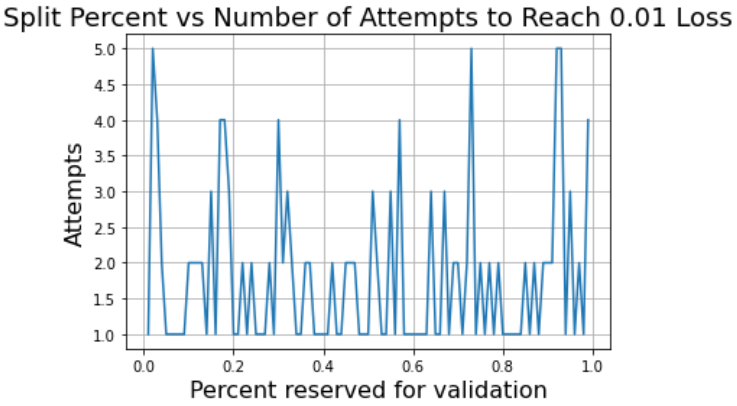


Figure 89: Number of attempts taken to reach the threshold loss value of 0.01 as a function of the percentage of data reserved for validation.

The tendency of these results to show that the neural network's performance doesn't vary drastically until extremely low amounts of data are reserved for testing is intuitive in the sense that the neural network is modeling physics. Physics problems often tend to scale well in neural networks.

CHAPTER 6. DISCUSSION AND CONCLUSIONS

In this study, a number of methods were demonstrated for formatting phononic crystal training data. From training neural networks with each of these methods, it is apparent that the trade-off between amount of information provided to the neural network is important to consider when creating an inverse design network. The lower information "largest bandgap method" is beneficial from an ease-of-use perspective as the user needs to supply less data, but the predictive performance is less than that of the more information dense, but more difficult to use "full band structure method."

The binary DOS method seems to be a promising answer for the inverse design problem. Section 5.2.3.2 shows that we can achieve decent to good predictive performance with this method, even though it offers less information to the neural network than the scalar DOS or full band structure methods. When used in tandem with the scheme described in 5.2.3.3., this method can be used to find designs that satisfy a bandgap, center frequency query very accurately. Binary DOS can be easily generated from a list of desired band gaps and center frequencies, meeting the ease-of-use requirement for the end user.

From this study, a software package was created that can serve as a framework for future phononic crystal neural network studies. The package currently offers multiple methods/options for customizing training data formatting (largest bandgap, full band structure, scalar DOS, binary DOS, include/exclude zero-bandgap designs, etc.). The package was built with an object-oriented design as such functionality should be easy to expand in the future.

A number of intuitions were also gained for successfully training neural networks with phononic crystal data. For example, in the formatting methods where multiple zero bandgap designs cause redundancies in the training data (largest bandgap, binary DOS) that these data points should be removed as they skew the predictive performance towards zero. Another example: it was generally found that final training loss needed to reach values around 0.001 or lower for the best predictive performance. This can be seen in the section 5.3 training/test split experiments where the predictive performance would degrade as the final loss increased past this value. Finally, Appendix B stores the neural network architecture parameters used for the experiments in this study. Though it was found that the exact parameters used do not have a large impact on the training process is will serve as a starting point for future work.

Finally, the training-test split experiment showed that for the case demonstrated, the amount of data necessary to reserve for training can stay low. This could signal that future data generation effort could require less data to be generated. This experiment however should be expanded to include other training datasets and formatting methods in the future for a more robust outlook.

6.1 Future Work

The work done in this study can be expanded on moving forward. This section lists some of the possible endeavors moving forward:

Though the scheme proposed in 5.2.3.3 does show promising results, it would be beneficial to take several geometry predictions and verify through COMSOL that the predicted designs match the desired bandgap and center frequency exactly. Once these results have been verified through simulation, fabrication and characterization of real PnCs can further verify the ML results.

Improving the scalar DOS method could also be a worthwhile endeavor. The scalar DOS method showed strong predictive performance but suffers from the ease-of-use problem. If the non-bandgap regions could be modeled successfully using some sort of stochastic process, then this method could replace binary DOS in the proposed scheme in 5.2.3.3. The higher amount of information offered by the scalar DOS method could be useful if, in future, more complex datasets, the scalar DOS method begins to fail.

Moving forward, it would be useful for the training datasets generated in the future to have parameterized geometry values. For example, the shape a hole in a unit cell could be modeled using a function in polar coordinates and the coefficients of that function could be ingested by the neural network. This would result in a larger number of possible novel crystal designs predictable by the neural network, instead of being limited by a single shape per dataset. Training data is the limiting factor when using neural networks and should be a careful consideration moving forward.

As there are many combinations of parameters to choose from when designing a neural network (number of layers, number of nodes per layer, optimizer, activation functions, batch size, learning rate, etc.) finding the optimal combination of parameters can be challenging and time consuming. It could be valuable to conduct more experiments to find the best possible training parameters for each of the of the training data formats discussed, especially the largest bandgap method which saw worse performance than the other methods. This would likely only result it slight performance gains, though.

Exploring more types of neural networks than MLP could also prove useful. For example, working with metasurfaces (both photonic and phononic crystals for example) comes with a "many-to-one" challenge. Multiple metasurfaces can results in the same response. This isn't a

problem when creating an RPN but is for the inverse problem/DPN. If the same response can correspond to multiple metasurfaces, the neural network can have difficulty converging. One solution to this is constructing what is known as a “tandem” neural network. The tandem neural net uses a “frozen” instance of a pre-created RPN to assist with DPN training and decision making. Exploring tandem neural networks could be worthwhile in the future, but also other types such as RNN or GAN could have their uses as well.

Another conceivable experiment would be to test interpolation vs extrapolation for these tests. Neural networks tend to be better at interpolation than extrapolation, but physics-based problems such as this one could show some ability to extrapolate. The neural network could be asked to predict values further and further outside the scope of the training data and compared to COMSOL simulation to get a measure of how far it can successfully extrapolate. It would also be useful to run more tests on the neural network’s ability to interpolate values given less and less samples in the training data space. A good starting point would be to extent the test done in 5.3 to more of the training data formatting methods.

Finally, there are many possible improvements that could be made to the software package in which the experiments were performed. One potential area of improvement would be Jupyter integration. Jupyter notebooks are Python scripts that are modular in nature; they are split into subsections that can be ran separately. Jupyter notebooks are especially useful when working with neural networks as the code that loads in the training data can be separated into its own block in the notebook. When using a normal Python script, the section of code that loads training data must be redundantly re-ran every time the neural network is trained. With Jupyter notebooks, the training data loading block can then be executed once and then the subsequent sections of code can be modified and re-ran without reloading the training data, saving time.

REFERENCES

[1] Y. Pennec, B. Djafari-Rouhani, H. Larabi and J. Vasseur, "Phononic crystals and manipulation of sound," *Current Topics in Solid State Physics.*

[2] Muhammad and C. Lim, "Phononic metastructures with ultrawide low frequency three-dimensional bandgaps as broadband low frequency filter," *Nature*, 2021.

[3] E. Tsymbal, "Section 5: Lattice Vibrations," University of Nebraska-Lincoln, [Online]. Available: https://unlcms.unl.edu/cas/physics/tsymbal/teaching/SSP-927/Section%2005_Lattice_Vibrations.pdf. [Accessed July 2022].

[4] N. Elabbasi, "Modeling Phononic Band Gap Materials and Structures," COMSOL, 10 February 2016. [Online]. Available: https://www.comsol.com/blogs/modeling-phononic-band-gap-materials-and-structures/#:~:text=What%20Is%20a%20Phononic%20Crystal,characteristics%20of%20mechanical%20wave%20propagation.. [Accessed July 2022].

[5] A. Fisher, "X-Ray Structure Determination of Proteins and Peptides," ResearchGate, [Online]. Available: https://www.researchgate.net/figure/Real-space-lattice-and-corresponding-reciprocal-space-lattice-a-Real-space-lattice_fig11_234265079. [Accessed July 2022].

[6] Y. Chen, J. D. Lee and A. Eskandarian, "Examining the physical foundation of continuum theories from the viewpoint of phonon dispersion relation," *International Journal of Engineering Science,* 2002.

[7] M. Dutta and M. A. Stroscio, "7.2 Elastic continuum model of phonons," in *Phonons in Nanostructures,* Cambridge, Press Syndicate of the University of Cambridge, 2001.

[8] M. A. Hopcroft, W. D. Nix and T. W. Kenny, "What is the Young's Modulus of Silicon?," *Journal of Microelectromechanical systems,* vol. 19, no. 2, 2010.

[9] A. Dertat, "Applied Deep Learning - Part 1: Artificial Neural Networks," Towards Data Science, 8 August 2017. [Online]. Available: https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6. [Accessed July 2022].

[10] S. Bahringer, "No Title," morioh.com, 2021. [Online]. Available: https://morioh.com/p/0be650ff3606. [Accessed July 2022].

[11] A. L. Chandra, "Learning Parameters, Part 1: Gradient Descent," Towards Data Science, 14 May 2019. [Online]. [Accessed July 2022].

[12] M. I. Hussein, "Reduced Bloch mode expansion for periodic media band structure calculations," 2009.

[13] A. Karttunen, "Phonon density of states with CRYSTAL and Phonopy," Aalto University, 7 July 2020. [Online]. Available: https://wiki.aalto.fi/display/IMM/Phonon+density+of+states+with+CRYSTAL+and+Phonopy. [Accessed July 2022].

[14] P. Schober, C. Boer and L. Schwarte, "Correlation Coefficients: Appropriate Use and Interpretation," Semantic Scholar, 1 May 2018. [Online]. Available: https://www.semanticscholar.org/paper/Correlation-Coefficients%3A-Appropriate-Use-and-Schober-Boer/b8d5dbafc507ec21221d523ae869fc14941f61ad. [Accessed July 2022].

APPENDIX

A. Example code for training DPN with full band structure data

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Feb 17 15:32:15 2022

@author: Drew Feltner
"""
import sys
sys.path.append('../../src')
import time
import pickle
import matplotlib.pyplot as plt
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense, Dropout
from tensorflow.keras.optimizers import Adam

# Custom imports
import training_data_formatter_banddiagram as tdfb
import scale_training_data as sd
import save_utils


# User input parameters
test_percentage = 0.2
calc_dos = False
num_epochs = 1000
training_data_dir = '/media/sf_shared/Thesis/save_folder/si_with_holes/'

# Begin tracking script run time
start_time = time.time()

nn_type = "DPN"

# Load data
print("Loading training data")
f = open(training_data_dir + "geometry.npy", 'rb')
geometry_data = pickle.load(f)
```

```
f = open(training_data_dir + "banddiagram.npy", 'rb')
banddiagram_data = pickle.load(f)

# Format and normalize training data between 0 and 1
x_data, y_data = tdfb.format_training_data(geometry_data,
                                           banddiagram_data,
                                           network_type="DPN",
                                           density_of_states=calc_dos)


# Split data into train/test
x_train, y_train, x_test, y_test = sd.scale_and_split_data(x_data, y_data,
test_percentage)

# Save the split training data for later evaluation step
save_utils.save_split_training_data(training_data_dir, nn_type, x_train, y_train,
x_test, y_test)

# Create neural net architecture
n = 1248 # 16 eigenmodes * 78 samples per eigenmode = 1248 nodes
input_design = Input(shape=n, name="designs")

dense_1 = Dense(n, activation="relu", name="dense_1")(input_design)
drop_1  = Dropout(0.2)(dense_1)
dense_2 = Dense(n, activation="relu", name="dense_2")(drop_1)
drop_2  = Dropout(0.2)(dense_2)
dense_3 = Dense(n, activation="relu", name="dense_3")(drop_2)
dense_4 = Dense(n, activation="relu", name="dense_4")(dense_3)
dense_5 = Dense(n, activation="relu", name="dense_5")(dense_4)
dense_6 = Dense(n, activation="relu", name="dense_6")(dense_5)

output_ref = Dense(6, activation="relu", name="gap")(dense_6)

dpn = Model(inputs=[input_design],
            outputs=[output_ref],
            name="dpn")

dpn.compile(optimizer=Adam(),
              loss="mean_squared_error",
              metrics=["accuracy"])

print(dpn.summary())
dpn_path = "../../src/net_designs/DPN"

# Create early stopping checks in case training loss starts to increase
```

```
checkpoint = ModelCheckpoint(dpn_path,
                             monitor="val_loss",
                             verbose=0,
                             save_best_only=True,
                             save_weights_only=True,
                             mode="min",
                             save_freq='epoch')

early_stop = EarlyStopping(monitor="val_loss",
                           min_delta=1e-7,
                           patience=100,
                           verbose=2,
                           restore_best_weights=True,
                           mode="min")

# Train network
dpn_fit = dpn.fit(x_train, y_train,
                  batch_size=100, epochs=num_epochs,
                  validation_split=0.1, shuffle=True,
                  verbose=1, callbacks=[checkpoint, early_stop])

# Plot training loss
dpn_hist = dpn_fit.history
fig, ax = plt.subplots()
ax.plot(dpn_hist.get('loss'), label='training', color='blue')
ax.plot(dpn_hist.get('val_loss'), label='validation', color='green')
ax.legend(loc='upper right')
ax.set_xlabel("Epoch")
ax.set_ylabel("Loss (MSE)")
ax.set_title("Training Loss")
plt.show()

# Save DPN weights
save_utils.save_neural_network(training_data_dir, nn_type, dpn)

end_time = time.time()
print("Ran in " + str(end_time - start_time) + " seconds")
```

B. Training Parameters Used

RPN: Largest Bandgap Method				
Layer Index	Layer Type	Number of Nodes	Activation Function	Drop Rate
0	Input Layer	N*	n/a	n/a
1	Dense Layer	Nx128	Relu	n/a
2	Dropout Layer	n/a	n/a	0.1
3	Dense Layer	Nx64	Relu	n/a
4	Dropout Layer	n/a	n/a	0.1
5	Dense Layer	Nx16	Relu	n/a
6	Dropout Layer	n/a	n/a	0.1
7	Dense Layer	Nx4	Relu	n/a
8	Dropout Layer	n/a	n/a	0.1
9	Output Layer	2	Relu	n/a

RPN: Full Band Structure Method				
Layer Index	Layer Type	Number of Nodes	Activation Function	Drop Rate
0	Input Layer	N*	n/a	n/a
1	Dense Layer	Nx2	Relu	n/a
2	Dense Layer	Nx4	Relu	n/a
3	Dense Layer	Nx8	Relu	n/a
4	Dense Layer	Nx8	Relu	n/a
5	Dense Layer	Nx16	Relu	n/a
6	Dense Layer	Nx32	Relu	n/a
7	Dense Layer	Nx64	Relu	n/a
8	Dropout Layer	n/a	n/a	0.2
9	Dense Layer	Nx128	Relu	n/a
10	Dropout Layer	n/a	n/a	0.2
11	Dense Layer	Nx256	Relu	n/a
12	Output Layer	1248	Relu	n/a

DPN: Largest Bandgap Method				
Layer Index	Layer Type	Number of Nodes	Activation Function	Drop Rate
0	Input Layer	2	n/a	n/a
1	Dense Layer	416	Relu	n/a
2	Dropout Layer	n/a	n/a	0.1
3	Dense Layer	320	Relu	n/a
4	Dropout Layer	n/a	n/a	0.1
5	Dense Layer	384	Relu	n/a
6	Dropout Layer	n/a	n/a	0.1
7	Dense Layer	224	Relu	n/a
8	Dropout Layer	n/a	n/a	0.1
9	Dense Layer	448	Relu	n/a
10	Dropout	n/a	n/a	0.1
11	Output Layer	2	Sigmoid	n/a

DPN: DOS Method				
Layer Index	Layer Type	Number of Nodes	Activation Function	Drop Rate
0	Input Layer	H**	n/a	n/a
1	Dense Layer	H	Relu	n/a
2	Dropout Layer	n/a	n/a	0.2
3	Dense Layer	H	Relu	n/a
4	Dropout Layer	n/a	n/a	0.2
5	Dense Layer	H/2	Relu	n/a
6	Dense Layer	H/2	Relu	n/a
7	Output Layer	N	Sigmoid	n/a

DPN: Full Band Structure Method				
Layer Index	Layer Type	Number of Nodes	Activation Function	Drop Rate
0	Input Layer	1248	n/a	n/a
1	Dense Layer	1248	Relu	n/a
2	Dropout Layer	n/a	n/a	0.2
3	Dense Layer	1248	Relu	n/a
4	Dropout Layer	n/a	n/a	0.2
5	Dense Layer	1248	Relu	n/a
6	Dense Layer	1248	Relu	n/a
7	Dense Layer	1248	Relu	n/a
8	Dense Layer	1248	Relu	n/a
9	Output Layer	N	Sigmoid	n/a

Training Data Split Test (Training/Validation and Training/Testing)				
Layer Index	Layer Type	Number of Nodes	Activation Function	Drop Rate
0	Input Layer	1248	N/A	N/A
1	Dropout Layer	N/A	N/A	0.2
2	Dense Layer	1248	Relu	N/A
3	Dropout Layer	N/A	N/A	0.2
4	Dense Layer	1248	Relu	N/A
5	Dense Layer	1248	Relu	N/A
6	Dense Layer	1248	Relu	N/A
7	Dense Layer	1248	Relu	N/A
8	Output Layer	6	Relu	N/A

* Number of nodes N in input layer equal to number of relevant input parameters. Number of nodes = 2 for individual circular hole training datasets (base height, circle radius), 5 for combined circular hole dataset (base height, circle radius, Si, SiC, W), and 3 for datasets with pillars (base height, pillar height, pillar radius).

** Number of nodes H is equal to the number of bins used to calculate DOS. For the studies in this paper, the number of bins was chosen to be 100, but the parameter is adjustable.