Evaluation of Deep Learning Algorithms in Predicting Seismic Response of a Reinforced Concrete Structure

by

Jacob A. Morgan

B.S. Civil Engineering
Rice University, 2023

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN CIVIL AND ENVIRONMENTAL ENGINEERING

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by:	Jacob A. Morgan Department of Civil and Environmental Engineering May 10, 2024
Certified by:	Oral Buyukozturk George Macomber Professor in Construction Management Professor of Civil and Environmental Engineering Thesis Supervisor
Accepted by:	Heidi Nepf Donald and Martha Harleman Professor of Civil and Environmental Engineering Chair, Graduate Program Committee

Evaluation of Deep Learning Algorithms in Predicting Seismic Response of a Reinforced Concrete Structure

by

Jacob A. Morgan

Submitted to the Department of Civil and Environmental Engineering
on May 10, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF ENGINEERING IN CIVIL AND ENVIRONMENTAL ENGINEERING

ABSTRACT

This thesis presents an evaluation of the performance of three well-established deep learning algorithms in predicting the response of a six-story instrumented reinforced concrete hotel in California to seismic excitation. Given the increasing availability of strong-motion data and expanded usage of deep learning in structural health monitoring, this thesis seeks to evaluate the predictions of purely data-driven and physics-informed architectures using processed instrumentation data in order to more accurately predict structural response for use in structural health monitoring and performance-based design applications.

By employing a variety of results metrics previously used in the literature, including correlation coefficients, normalized error distributions, and peak errors, this thesis examines different components of the models' capabilities to learn more about patterns in the data learned by the computational mechanisms of each architecture, and exploring the feasibility of a generalized approach for further application in structural response prediction.

Findings from the work show the data-driven Long Short-Term Memory (LSTM) network performing the most accurately, but not consistently outperforming the other algorithms. Some trends in the data could be evidence of how different architectures may be better equipped in predicting different mode shapes and frequency contents. For example, the data-driven and physics-guided LSTM models predicted the third floor's response more accurately than the roof, whereas the physics-guided convolutional neural network (CNN) was the opposite, showing a contrast between the two base architectures. This thesis also contributes to this growing field by documenting the experimental setup in detail to allow for the replication of results and for the facilitation of future application by structural engineers.

As structural engineering research in deep learning continues to gain popularity, this thesis provides an experimental basis of a case study that can be followed and replicated to motivate future experimentation, as well as offering compelling different directions that future work could be directed to further the usage of deep learning in structural response prediction and structural health monitoring as a whole.

Thesis supervisor: Oral Buyukozturk
Title: George Macomber Professor in Construction Management; Professor of Civil and Environmental Engineering

Acknowledgments

I would first like to thank Professor Oral Buyukozturk, my Thesis Supervisor, for his valuable guidance and support in shaping the direction of this work. When starting this project in September 2023 with no prior knowledge in deep learning or AI in general, Professor Buyukozturk's insightful questions dove to the root of the concepts and helped guide my learning to what I know today. The learning curve of this topic was not easy, but each of our meetings inspired me and gave me new ideas to troubleshoot issues during difficulties in the study, and I learned a lot of new skills in the process.

I would also like to thank Professor Kalil Erazo (Rice University) for his thorough and thoughtful feedback to improve this work, especially regarding figure design, as well as discussions on potential opportunities for future work. My interest in earthquake engineering began as an undergraduate under his instruction at Rice in a graduate-level course on the subject. This work would not have been possible had he not trusted in me to handle the rigor of his course.

I would like to acknowledge the supplementary information provided to me by Professor Ruiyang Zhang (Southeast University of China) and Professor Hao Sun (Renmin University of China) and thank them for their assistance. Their advice in troubleshooting the model training, particularly the PhyLSTM, and clarification that they used V2 data from the CESMD for their respective studies, made this study possible.

I would also like to acknowledge the assistance with VSCode and GitHub I received from Raphaël Trézarieu, another student in the Structural Mechanics and Design MEng program. His suggestion in using VSCode to commit to GitHub simplified my workflow in uploading my Python files so I can share my work with others.

Finally, I would like to express my gratitude for my family, whose hard work and commitment to my education made this opportunity possible. I am grateful for the love and support you have given me this academic year and I am excited for what the future holds.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Concrete is the most widely used building material in the world. The majority of buildings in the world are constructed of reinforced concrete and an estimated 70% of the world's population live in a building that has concrete components [1]. Recent earthquakes around the world have shown that sub-optimally designed and constructed reinforced concrete buildings catastrophically fail in major seismic events, which has resulted in unfortunate losses of life and damage to property. Thus, in order to prevent future structural failures, it is important to be able to quickly and accurately predict the seismic structural response of both newly designed and existing concrete structures.

Finite element modeling is currently one of the most widespread and powerful numerical methods for seismic structural response estimation [2]. However, due to the dynamic nature of strong seismic loading, structural response tends to be nonlinear in severe events. Modeling this nonlinearity requires the computation of structural behavior at very small time intervals through the analysis of many fine mesh elements. This results in a computationally expensive analysis that is not guaranteed to be accurate due to uncertainties in loading and model properties, and an analysis that can take several hours if not days to run [2].

In the past decade, due to advances in computing capabilities and data storage, structural engineering researchers and practitioners have expanded investigations into the usage of machine learning to revolutionize the industry, as observed in other fields like bioengineering, medicine, and advertising [3]. A variety of deep learning models have been proposed in the literature, showing promise across different aspects of structural health monitoring (SHM). Some deep learning models are applied to local SHM applications, like inspections and non-destructive testing (NDT) [4]. Other deep learning models are targeted at global SHM applications, sometimes referred to as vibration-based methods, like seismic response prediction [4].

Current seismic structural response prediction methods include model-based methods, which rely on both a finite element (FE) model of the structure and limited instrumentation data. An example of this is a model-based observer (MBO) which has demonstrated success [5]. While applicable at a building-level scale, when there is a lot of data known about the structure, model-based approaches are not easily scaled to the community or city level, due to having limited time available to create FE models and limited access to structural plans. Thus, data-driven methods like deep learning, which can also include physics-guidance to inform predictions, have the advantage to scale and simplify analysis.

This thesis will focus on vibration-based methods, like seismic response in particular. Several seismic response prediction deep learning models have been tested and validated numerically, through single-degree-of-freedom systems, structural metamodels, and Bouc-Wen hysteresis models, and some experimentally, like on an instrumented six-story reinforced concrete hotel in California [2], [6], [7]. The findings of this thesis will show the capabilities and limitations of popular deep learning models in predicting the response of an instrumented reinforced concrete hotel with limited training data, and motivate future improvement of neural network architectures to better accommodate physics constraints and nonlinearity in the analysis.

1.2 Problem Statement

The objective of this thesis is to evaluate the performance of deep learning algorithms on an instrumented structure using a variety of results metrics. First, a literature survey was conducted covering all facets of the work, including the fundamentals of machine learning and deep learning, how they are used in structural response prediction, and popular algorithms for which data and codes were readily available. Due to the data-driven nature of the project, it is also important to investigate the context of strong-motion instrumentation in California, where this study is focused on, and how acceleration data is acquired and processed in such a way that the outputs of the respective models have meaning in the context of structural behavior.

Following the literature survey, three popular algorithms (versions of Zhang et al.'s LSTM, PhyCNN, and PhyLSTM) were selected to investigate for this study and their architectures were examined [2], [7], [8]. Each architecture processes data differently through different computational mechanisms and layer structures, which could result in trends of strengths and weaknesses when evaluating different time histories and frequency compositions for varying structural types and systems.

Prior to beginning a new case study, the models, provided in repositories through Ruiyang Zhang's GitHub account[1], were first run using the provided data to recreate their initial study results. Following confirmation that the retrained results matched the results in the literature through qualitative and quantitative results metrics detailed in their respective studies, the originally-provided codes were modified as-needed to accommodate this case study.

[1]At the time of publication of this study, Ruiyang Zhang's GitHub account can be accessed at this link: https://github.com/zhry10.

Ground motion and structural response data were assembled and compiled for this new case study. The deep learning algorithms were retrained using the new data, and the performance of the algorithms was analyzed using quantitative statistical methods such as correlation analysis, confidence interval of normalized prediction errors, and peak errors to study the accuracy and effectiveness of the algorithms analyzed. The experimental setup is documented in detail throughout to allow for the replication of results and for the facilitation of future application by structural engineers.

Chapter 2

Background

2.1 Machine Learning Fundamentals

Artificial intelligence (AI), defined generally, is the development of computer systems able to perform tasks that normally require human intelligence. These tasks include, but are not limited to, learning, reasoning, problem-solving, perception, language understanding, and speech recognition. Though often incorrectly considered synonymous with AI, machine learning (ML) is in fact a subset of AI. ML focuses on the development of algorithms and statistical models that enable computers to perform a task without explicit instructions.

2.1.1 Machine Learning Paradigms

There are several learning paradigms within ML, with varying levels of human input. These consist of supervised learning, unsupervised learning, and reinforcement learning. Each of these learning paradigms has benefits, and the choice of the appropriate approach depends on the nature of the problem at hand and data availability.

Supervised Learning

In supervised learning, the machine learning model is trained on a labeled input dataset, known as training data. Training involves pairing the training data with corresponding labeled output data, which is sometimes referred to as target data. The goal of training is for the model to learn the mapping from the input data to the output data, so that predictions or classifications can be made based on new, unseen data, often referred to as test data. During the training process, the model adjusts its parameters based on the input-output pairs, minimizing the difference between its predictions and the actual labels. These differences in predictions and actual labels are called losses, and can be categorized by different equations and probabilities. Following the completion of training, which can be a certain number of epochs or iterations, or a convergence criterion, the model is then evaluated on a separate test dataset to assess its generalization performance.

Generally, structural seismic response prediction itself is a form of supervised learning, where the input/training data is labeled ground acceleration data and the output data is some kind of structural response, e.g. accelerations, velocities, and displacements, at several spatial locations throughout the structure.

Unsupervised Learning

Unlike supervised learning, unsupervised learning deals with unlabeled data, and the algorithm explores the inherent structure or patterns in the data without explicit guidance. The goal is generally to uncover hidden relationships or groupings within the data. Common techniques in unsupervised learning include clustering, where the algorithm groups similar data points together, and dimensionality reduction, which aims to simplify the dataset while retaining important information. An example of unsupervised learning is K-means clustering, which seeks to group data points into "clusters" based on similar characteristics.

Reinforcement Learning

Reinforcement learning is inspired by behavioral psychology. It involves training an agent, or decision maker, to take actions in a certain environment to maximize a reward. The agent learns by interacting with the environment, receiving feedback in the form of rewards or penalties for its actions. Some examples of reinforcement learning include autonomous vehicles and recommendation systems, which allow machines to learn in a trial and error format like humans.

2.2 Deep Learning Fundamentals

Deep learning (DL) is a subset of machine learning that focuses on neural networks with multiple layers, which are known as deep neural networks. A neural network is a computational model inspired by the way biological neural networks, like in the human brain, work. Neural networks consist of interconnected nodes, also known as artificial neurons or perceptrons, organized into layers. These layers typically include an input layer, one or more hidden layers, and an output layer. In deep learning, the neural networks are designed to automatically learn and represent features from data through the hierarchical arrangement of layers and mathematical computations. Deep learning excels at learning hierarchical representations of data, allowing it to automatically extract relevant features for a given task.

2.2.1 Traditional Neural Network Architectures

In order to solve different problems, neural networks are rearranged into different structures and configurations, which are called architectures. Some common neural network architectures, and their use cases are summarized below.

Multilayer Perceptrons (MLPs)

Multilayer perceptrons (MLPs) are also referred to as "dense" networks or "fully connected" networks, as neurons in one layer are connected to all neurons in the next layer [9]. MLPs are considered to have a simple network structure as they do not have specialized layers for handling spatial relationships or sequential dependencies. MLPs are known as universal approximators and are typically used for classification and regression across an entire set of data, i.e. globally [9]. They are capable of learning and representing complex, nonlinear relationships within sets of data. The simplicity of MLPs also causes it to have drawbacks. In an MLP, the input data is treated as an independent example at each iteration, due to the lack of a specialized layer to handle sequential dependency. This limits its ability to deal with time-series data, natural language processing, and other sequences in general. Additionally, since MLPs do not have any specialized layers for handling spatial relationships, they only recognize global patterns, so they may struggle to capture local patterns or spatial relationships that are important in tasks like image recognition and dealing with sequences. The computational expense of MLPs can also be a drawback as more layers and neurons are added, due to its dense/fully connected structure. Thus, more specialized architectures are used in the realm of structural seismic response prediction.

Convolutional Neural Networks (CNNs)

Convolutional neural networks are a neural network architecture inspired by the natural visual cortex of animals [10]. Although there are several different variations of CNN architectures employed in different use cases, there are three main types of layers that make up the network [10]. These three layers are known as convolutional layers, pooling layers, and fully connected layers.

Convolutional layers are the building blocks of CNNs, which are designed to detect and extract patterns in the input data. A convolutional layer consists of a set of filters, also referred to as kernels, that slide or convolve across the input data to detect local patterns or features. A convolution operation is shown in Figure 2.1.

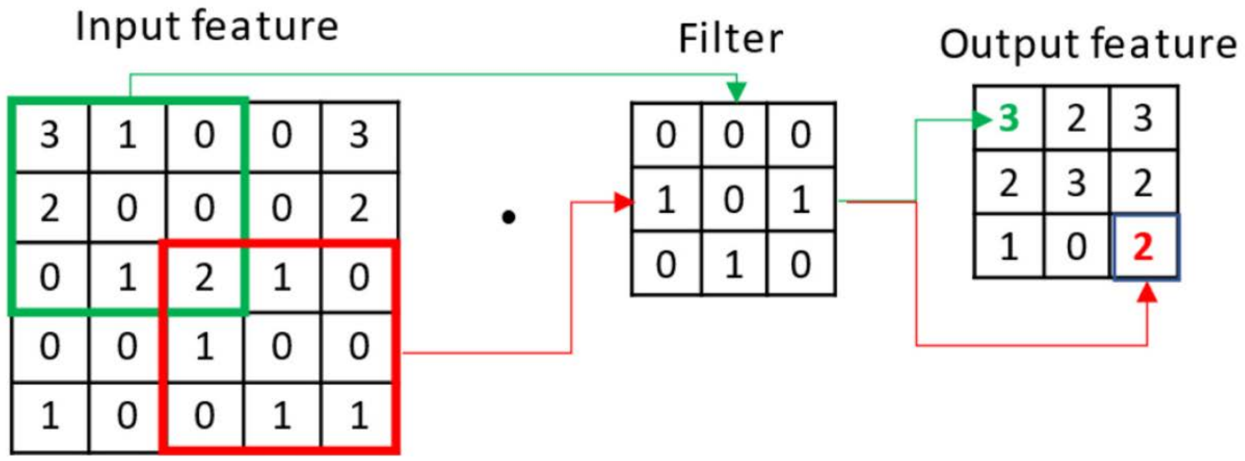


Figure 2.1: Example of a convolution. Reproduced from [4] and used with permission.

These local patterns or features are detected through element-wise multiplications of the filter weights with the input data and then summed to produce feature maps. The filter weights are learned by the model through training.

After the convolutions occur, pooling layers are used to downsample. There are several different types of pooling, but two common methods include max pooling, which takes the maximum value of the region, and average pooling, which takes the average value. These types of pooling are shown in Figure 2.2. Pooling after convolutional layers makes the patterns independent of scale, rotation, translation, and weight, making the network more robust, and also more computationally efficient due to the downsampling [10].

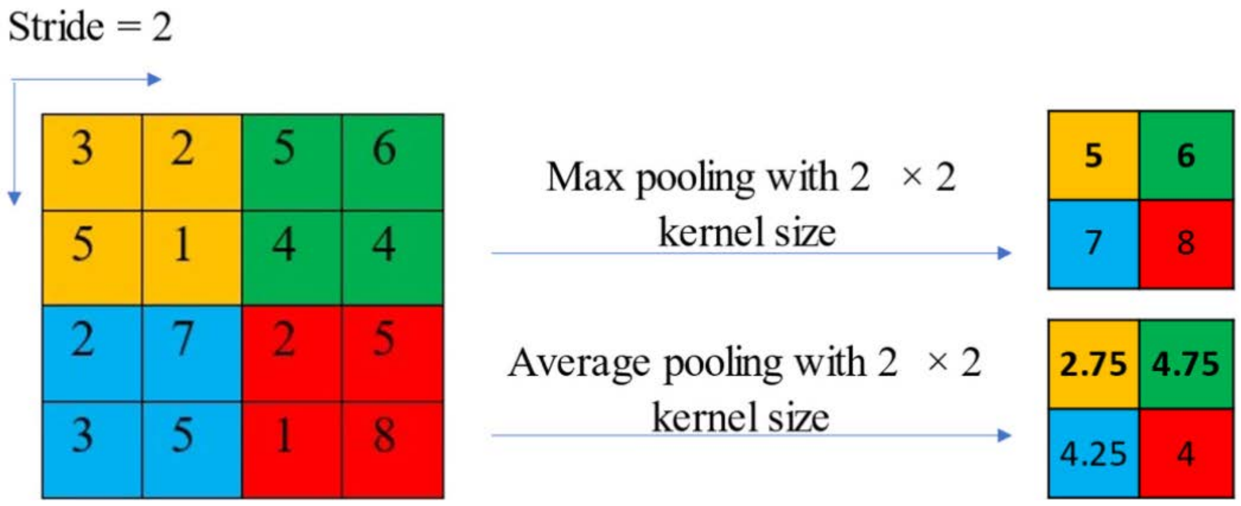

Stride = 2
3 2 5 6
5 1 4 4
2 7 2 5
3 5 1 8
Max pooling with 2 × 2 kernel size
5 6
7 8
Average pooling with 2 × 2 kernel size
2.75 4.75
4.25 4

Figure 2.2: Example of a pooling layer, showing max and average pooling. Reproduced from [4] and used with permission.

Finally, fully connected layers are traditional neural network layers, as in the case of MLPs above, where all of the neurons are connected. In the case of a CNN, fully connected layers are used at the end of the network to make predictions and to combine all of the feature maps together. Prior to being sent through a fully connected layer, the output of the last pooling layer is flattened into a vector. In the case of classification problems, the final fully connected layer has as many neurons as there are classes. In the case of multi-class classification, a softmax activation function, which is used to normalize predictions to a probability distribution, is often used. CNN architectures are also capable of handling regression problems, which Zhang et al. mention is often overlooked due to the primary usage of the architecture in classification [8].

Long Short-Term Memory Networks (LSTM)

Long short-term memory networks are a type of recurrent neural network (RNN) [2]. Recurrent neural networks are designed to process sequential data using connections that form a cycle from one entry to the next. This is unlike traditional feed forward neural networks, like the previously mentioned MLPs, that process input data independently and do not have "memory". Examples of a feed-forward network, like an MLP, and an RNN are shown in Figure 2.3. The cyclic connections of RNNs allow them to "remember" information over time, due to their ability to process sequences of inputs, where each input depends on both current inputs and previous inputs in the sequence.

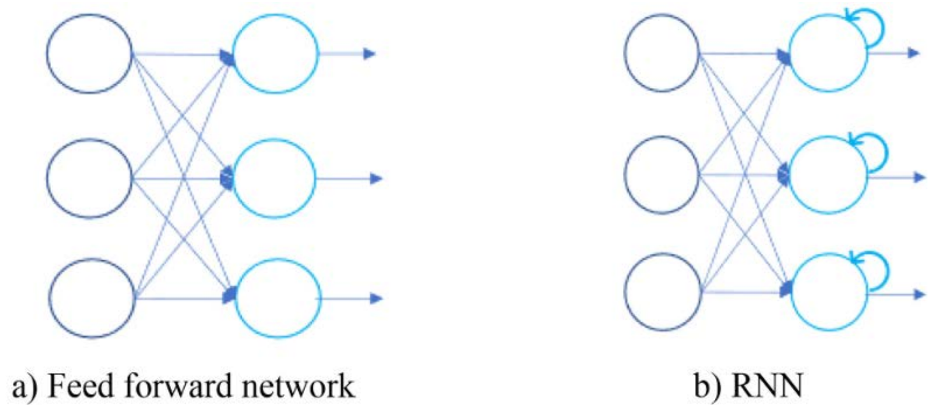


Figure 2.3: Example of a) a feed forward network and b) an RNN. Reproduced from [4] and used with permission.

To accomplish this, RNNs like LSTMs are made up of cells with different components that carry out different steps in the cycle of converting input data to output data [2]. Each LSTM cell consists of four interacting components: a cell state, a forget gate, an input gate, and an output gate. Each component has a different role in updating and controlling the flow of information, allowing the LSTM to keep or get rid of information over the sequence of the data [2].

The cell state acts as the memory of the network, which carries information throughout the sequence of the data. At each time step, it undergoes minimal changes and can "forget" irrelevant parts of the previous state, with new relevant information being added. The forget gate determines the information that is forgotten, or discarded, from the cell state by looking at the previous hidden state and the current input. The input gate controls the addition of new information from the current input and the previous hidden state to the cell state. The output gate determines what information from the cell state will be used to update the hidden state.

Temporal Convolutional Networks (TCNs)

Temporal convolutional networks are neural networks designed specifically for sequence modeling tasks like time-series analysis and natural language processing [11]. The architecture seeks to provide both the strength of LSTMs in sequentially modeling temporal dependencies and the strength of CNN in efficiency through convolutional operations, particularly in large datasets. Additionally, the architecture allows for the ability to use parallel computing, improving training efficiency, and they do not experience the "vanishing gradient problem" that other architectures like RNNs do, due to a backpropagation path different from the sequence's temporal direction [12].

TCNs use causal convolutions, which means that the prediction for a time step can only depend on past and present data, not future data [12]. This requires a long effective history, which means that a deep network with many convolution layers is necessary. This is resolved by dilated convolutions which skip input values at a predetermined rate, using larger filter sizes and increasing the dilation factor [12]. This increases the receptive field, the area of input data for a convolution, giving the causal convolutions the effective history it requires to make accurate predictions.

2.2.2 Physics-Informed Neural Networks (PINNs)

Although data-driven machine learning models have experienced advancements in recent years, limitations exist when deploying purely data-driven models for real-world applications, like in this case of structural response prediction [13]. Some of these limitations include the lack of "commonsense reasoning" and adherence to the laws of physics and physics constraints, which negatively impacts the model's robustness, which is the model's ability to maintain its performance under variations or noise in the input data [13]. To get around these limitations, several methods have been proposed to integrate physical knowledge with deep learning, resulting in Physics Informed Neural Networks (PINNs). These networks seek to use both empirical knowledge, as well as prior physical knowledge built into the architecture of deep learning models, to give more accurate and more reliable predictions than those of data-driven models [13].

Along with more accurate predictions in some applications, PINNs have other advantages over purely data-driven methods. The model training is more efficient and less dependent on having a large input dataset, since the physics guides the model toward more physically realizable solutions. This is valuable in the realm of structural response prediction as instrumentation on real-world structures is limited. Additionally, embedded physics can make the model's predictions more interpretable and more trustworthy, which is particularly important in structural engineering where safety and reliability are paramount [3]. This additional transparency would benefit the wider usage of deep learning in practice, resolving issues of interpretation and quality control, since deep neural networks are still known as somewhat of a "black box", even to leading experts in the field [3].

Depending on the architecture of the model and the problem at hand, prior physical knowledge can be incorporated in many ways. In the realm of structural response prediction, one common way is building the prior physical knowledge into the loss function to guide the training of the model [13]. This includes using the laws of dynamics, therefore considering the relationships between the acceleration, velocity, and displacement of the structural system and the physical properties of inertia, stiffness, and damping that govern the response at various time steps.

2.3 Earthquake Response Data and Strong-Motion Instrumentation

As this study is using earthquake response data, it is also important to understand where the data come from and the instruments that record it. Although earthquake detection devices have been around since the Ancient Chinese, strong-motion instrumentation in its current form, consisting of sensors that are capable of measuring and recording the ground motion of earthquakes, has taken off within the past century. After learning about the dangers of earthquakes from their Japanese counterparts at the World Engineering Congress in Tokyo in 1929, American engineers saw the need to develop instruments to record and monitor the responses of structures [14]. The seismograph, a device that had been around for four decades at the time, only measured ground displacement [15]. In 1931, the U.S. Congress set aside funds for the establishment of an engineering seismology program and for the creation of a national strong-motion network. The first accelerograph, adapted from the Wood-Anderson seismograph, was created by the National Bureau of Standards [14]. Soon after, the Long Beach earthquake hit Southern California in 1933. The successful recordings of the earthquake motivated the advancement of instrumentation and the expansion of strong-motion monitoring networks across the nation [14].

2.3.1 Accelerographs: Then and Now

The first accelerographs developed were optical-mechanical devices, recording ground accelerations onto photographic paper or film, and are commonly known as analog recorders [15]. Though state-of-the-art in their time, analog instruments have since been surpassed by digital instruments [15].

Digital accelerographs became more popular and available in the late 1970s and early 1980s, due to advances in recording technology and storage. Digital instruments provided solutions to the disadvantages of analog instruments. Digital instruments operate continuously, instead of relying on a trigger to activate them, as they have reusable storage. They also have a wider frequency range with a higher resolution than analog recording, and the instrument itself performs the digitization of the data, eliminating the additional digitization process necessitated by analog instruments [15].

Although this study uses completely digital data, analog data still have some use cases. More than half of the accelerograph records from the infamous 1994 Northridge earthquake came from analog instruments, and it took until the late 1990s for the prevalence of recordings from digital instruments to surpass those from analog ones [15]. However, despite the prevalence of digital instrumentation, Boore and Bommer believe that it will take several decades for the current strong-motion database to be updated with all-digital records and make the analog records redundant [15].

2.3.2 California Strong-Motion Instrumentation Program (CSMIP)

After the destructive 1971 San Fernando earthquake, the California Strong-Motion Instrumentation Program (CSMIP) was established by the California Geological Survey (CGS) in 1972 in order to collect more data to better understand how seismic events affected structures [16]. The program installs and maintains strong-motion instruments on selected structures and ground-response sites based on their potential to provide valuable data for informing design practices and safety. Representative buildings of all structural systems (frames, shear walls, base isolation) and materials (steel, concrete, masonry, and timber) are instrumented in locations where seismic events are mostly likely to occur [17]. Critical 'lifeline' structures like highway bridges, dams, power plants, ports, and air traffic control towers, were designed to specific criteria, unlike buildings, and are instrumented to better understand their seismic response and improve analysis procedures [17].

According to Huang and Shakal, 12 to 20 sensors, usually accelerometers, are installed on average in a building [17]. After studying the lateral systems of the structure in question, the sensors are laid out by CSMIP personnel in such a way to capture the structure's different vibration modes. Sensors are always placed at the base of a structure, which may be a ground floor or a basement level, the roof of a structure, and at intermediate levels as needed. The sensors are connected to a central recorder by low-voltage electrical cabling to efficiently collect, process, and store the data. The combined system allows for the instruments to record in a synchronized manner by sharing the same triggering mechanism, as well as simplifies maintenance and data retrieval processes [17].

2.3.3 Strong-Motion Data Types

Strong-motion data in both unprocessed and processed forms are offered for download by the Center for Engineering Strong Motion Data (CESMD). The CESMD is a collaboration between the CGS and the United States Geological Survey (USGS) [18]. The first form of data is the raw digitized output from the accelerograph, in the case of analog instruments, which is referred to as Volume 1 (V1), or Phase 1 data [19]. In the case of digital accelerographs, the data are transferred electronically from the machine's memory, thus eliminating the need for digitization [15]. The second form of data, Volume 2 (V2), or Phase 2 data, consists of processed acceleration, velocity, and displacement time histories obtained from the Volume 1 data through a series of baseline and instrument corrections and filtering [19]. The velocity and displacement time histories are obtained via numerical integration of the V2 acceleration data and receive additional filtering as necessary. The CESMD mentions that most earthquake engineering analysis and applications utilize the V2 data. The third form of data, Volume 3 (V3), or Phase 3 data, consists of spectral information computed from the accelerograph [19].

Chapter 3

Literature Review

3.1 Deep Learning Studies in Structural Response Prediction

3.1.1 Zhang et al. LSTM Study

Zhang et al. investigated the computational time and training accuracy of two different LSTM architectures when modeling the structural responses of numeric metamodels and a CSMIP-instrumented hotel (CSMIP Station 23287) [2]. The two investigated architectures were 1) a full sequence to sequence LSTM network, named LSTM-f, and 2) a stacked sequence to sequence LSTM network, named LSTM-s. The LSTM-f architecture modeled the pairwise input and output along the entire time history, while the LSTM-s architecture reorganizes the inputs into stacks. The LSTM-s architecture outperformed the LSTM-f in both computational time and in prediction accuracy of numerical test cases. The LSTM-s model successfully predicted the displacements of the 3rd floor and roof of the hotel [2].

3.1.2 Zhang et al. PhyCNN Study

Zhang et al. proposed a physics guided convolutional neural network (PhyCNN) to model structural response of a structure subjected to ground motion [8]. The model's performance was validated using numerical metamodels and the same instrumented San Bernardino hotel (CSMIP Station 23287) as the aforementioned LSTM study [2], [8]. To incorporate the prior physics knowledge into the model, the law of dynamics was built into the loss function, separately from the data loss calculated when training and validating the input data. A graph-based tensor differentiator was used to calculate the derivatives of the structural displacement, velocity, and normalized restoring force in order to update the loss at each epoch.

After comparing the numerical model prediction results of the PhyCNN to the predictions obtained by the traditional CNN, Zhang et al. found that the PhyCNN outperformed the traditional CNN without physics guidance. Just like the LSTM model, the PhyCNN model successfully predicted the displacements of the 3rd floor and roof of the hotel within a tight range [2], [8].

3.1.3 Zhang et al. PhyLSTM Study

Zhang et al. proposed two different physics-guided multi-LSTM network architectures that incorporate available physics knowledge into the LSTM networks to constrain and enhance the learning process. The architectures seek to better capture system nonlinearity from the scarce training datasets available and use linked LSTM networks to model various aspects of the structural response [7].

The first architecture, known as PhyLSTM2, consists of two interconnected LSTM networks, combined with a graph-based tensor differentiator, like the one used in the PhyCNN study, which incorporates physics into the architecture. The first LSTM network models the input-output relationship between ground motion to structural response variables, i.e. displacement, velocity, and the hysteretic parameter, r, which models the structure's energy dissipation behavior through inelastic deformation. The second LSTM model models the mass-normalized restoring force, which comes from the law of dynamics.

The second architecture, known as PhyLSTM3, is built on the PhyLSTM2 architecture but adds a third LSTM network dedicated to modeling the hysteretic parameter, r. The additional network dedicated to the hysteretic parameter theoretically allows the architecture to capture more complex nonlinearity.

3.1.4 Liu et al. PI-LSTM Study

Liu et al. proposed a physics-informed LSTM network inspired by Zhang et al.'s PhyCNN and PhyLSTM architectures [2], [6]–[8]. Its performance was showcased against these preexisting architectures, along with a purely data-driven LSTM model, in both numerical and experimental validation formats.

In the numerical validation of a single-degree-of-freedom system subjected to random ground motions, the PI-LSTM was compared to the PhyCNN and PhyLSTM in the prediction of displacement and velocity of the system, as well as the restoring force latent variable. The PI-LSTM outperformed the PhyCNN in predicting all three categories. The PI-LSTM performed similarly to the PhyLSTM, with the exception of predicting the restoring force, of which the PhyLSTM performed the worst of the three.

In an experimental validation, Liu et al. compared the performance of the PI-LSTM, PhyCNN, and a data-driven LSTM from the same 6-story hotel as previously mentioned in Zhang et al.'s studies [2], [6], [8]. In predicting the responses of the hotel's third story, the PI-LSTM performed similarly to Zhang et al.'s PhyCNN in predicting structural displacement, but outperformed the PhyCNN in predicting accelerations [6], [8]. In predicting the structural displacement and acceleration, the PI-LSTM more noticeably outperformed the PhyCNN, particularly in predicting acceleration [6], [8].

3.1.5 Günay et al. TCN Study

Günay et al. used a Temporal Convolutional Network (TCN), originally proposed by Lea et al., to predict structural responses of numerical systems and instrumented structures when subjected to ground motion [11], [20]. After validating the TCN's accuracy of modeling a linear elastic SDOF system, the TCN was used to predict the responses of three CSMIP-instrumented structures: the same 6-story hotel in San Bernardino that Zhang et al. and Liu et al. used in their studies (CSMIP Station 23287), along with a 4-story hospital in Hemet (CSMIP Station 12267), and a 54-story building in Los Angeles (CSMIP Station 24629) [2], [6], [8], [11].

Along with comparing the true and predicted time histories of acceleration, the frequency contents of the true and predicted acceleration time histories were also compared. This gives insights into the interpretability of the model, like how the model is learning and what patterns it is picking up, as well showing any limitations the model has, particularly when structures are experiencing nonlinear behavior, as indicated by period elongation with increasing shaking intensity [11].

Inherent characteristics of each structure's response to seismic events are ingrained into its accelerometer data, like natural periods, damping ratios, and the effects of higher mode contributions [11]. For the model to be able to learn all of these characteristics, the model needs enough training data. To find how little data was needed to provide what was deemed an accurate prediction, Günay et al. performed a parametric study using varying numbers of training records. It was found that a training set size of 10 motions was sufficient for predicting responses of the three buildings within the linear elastic range. Outside of the linear elastic range, more records are needed to predict the motion. The 6-story hotel in San Bernardino experienced nonlinearity in the north-south direction during one seismic event that was accurately predicted by the TCN model using a training set with 23 different ground motion records [11].

3.2 Research Gaps

This thesis seeks to go beyond previous studies in the literature by not only providing results and commentary from the experiments. As machine learning is not typically taught in a structural engineering curriculum, it is expected that there is a steep learning curve for unfamiliar students, researchers, or practitioners seeking to learn more about or to enter this area of research.

This thesis and accompanying GitHub repository[1] will provide the resources and information for students, researchers, or practitioners to replicate results and create new studies without needing any prior knowledge of deep learning or any advanced programming knowledge.

[1] This study's GitHub repository can be accessed via: https://github.com/jacobmorgan2023.

Chapter 4

Methodology

4.1 Selected Algorithms

Three different deep neural network architectures for structural seismic response prediction were selected to be used in this study: Zhang et al.'s LSTM-s, Zhang et al.'s PhyCNN, and Zhang et al.'s PhyLSTM3. For simplicity, these algorithms are referred to herein as LSTM, PhyCNN, and PhyLSTM. The original copies of the algorithms, along with input data, are available via Ruiyang Zhang's GitHub account[1] with an MIT License for reproduction, granting usage for commercial and non-commercial purposes. In addition to having input data and code being available on GitHub, these three studies have each been cited in hundreds of papers on the subject, showing their impact on the advancement in this field.

4.2 Experimental Testbed

4.2.1 Virtual Environment

Creating a virtual environment, a self-contained directory, is an important first step of a data science project to ensure that models have the dependencies to run properly. Anaconda, a popular distribution for Python, was chosen for this project, as it provides a powerful tool for managing these virtual environments. Anaconda offers a controlled space where dependencies (importable external components or libraries) can be isolated, as well as offering a robust package management system, allowing users to easily install, update, and uninstall packages within their virtual environments.

Managing dependencies is an important consideration for this project, as the three architectures are deployed using TensorFlow version 1.x, which is now recognized as legacy code due to its lack of support. Additionally, many cloud-based notebooks, like Google Colab, no longer offer native support for TensorFlow version 1.x. at the time of publication of this study. Details regarding the virtual environment and dependency versions are provided in

[1]At the time of publication of this study, Ruiyang Zhang's GitHub account can be accessed at this link: https://github.com/zhry10.

this study's GitHub repository[2].

4.2.2 Machine Specifications

All computations in this study have been performed by an off-the-shelf laptop with an Intel Core i7-9750H processor and an NVIDIA GeForce GTX 1650 graphics card. The machine used in this study is running Windows 10 with CUDA Toolkit V10.0, enabling TensorFlow to leverage GPU-assisted training. The usage of an off-the-shelf machine emphasizes the need for computationally efficient networks that can produce accurate predictions.

4.3 Algorithm Setup

Prior to beginning any new analysis, the LSTM, PhyCNN, and PhyLSTM needed to be configured to be able to run the data from this case study instead of the respective input data files that Zhang et al. posted on GitHub alongside each algorithm. As mentioned previously, LSTM and PhyCNN studies provided a field validation of the structural responses of a six-story hotel in San Bernardino (CSMIP Station 23287) subjected to various earthquakes, while PhyLSTM provided numerical validation of a Bouc-Wen hysteresis model [2], [7], [8]. High-level descriptions of the set-up process for each algorithm are summarized in the following subsections. Modified versions of the algorithms used in this study can be found in this study's GitHub repository.

4.3.1 LSTM Setup

The LSTM model algorithm, along with input data from the respective study, was accessed from Ruiyang Zhang's GitHub account. The LSTM-s algorithm set up to run the San Bernardino hotel data was downloaded to be used with this study. Minor syntax changes were made to the algorithm to accommodate library imports and changes in the window sizing, but the structure of the model was kept intact. The original algorithm was set up with an output window size of 2, which took in a time history input of 7200 data points to create a prediction time history with 3600 data points. To ensure that each study uses the same training data to level the playing field, the window size was changed to 1, so that a time history input of 3600 data points would produce a prediction time history of 3600 data points.

As in the original study, the LSTM was set up to train for 20000 epochs with a learning rate of 0.001. As the model trains, the model with the new best validation loss is saved as a Keras model. Keras is a high-level neural network library that runs on top of TensorFlow [21]. Following the completion of the LSTM model training, the best model is loaded from the training epoch with the lowest validation loss.

The LSTM architecture had a predetermined validation set, with training and validation indices provided in the input data, which corresponded to earthquakes in the input training motion dataset to be used for training and validation. The purpose of these indices is to

[2]This study's GitHub repository can be accessed via: https://github.com/jacobmorgan2023

initialize the training and validation sets. These sets are not static, however. As the model trains, the training and validation ground motions are randomly shuffled.

4.3.2 PhyCNN Setup

The PhyCNN model algorithm, along with input data from their respective study, was provided on GitHub by Ruiyang Zhang. The PhyCNN algorithm set up to run the San Bernardino hotel data was downloaded to be used with this study. Like with the LSTM, minor syntax changes were made to the algorithm to accommodate library imports, but the structure of the model was kept intact. As in the original study, the PhyCNN was set up to train for 20000 epochs with a learning rate of 0.001.

Since the PhyCNN network performs convolutions on the data, looking for patterns in small subsets of data, as opposed to sequential predictions, the input data is mapped to the output data differently than in an LSTM network. When setting up the training and prediction datasets, attention has to be paid to their respective dimensions. The output dataset from the prediction ground motions has to be the same dimension as the output dataset of the training ground motions, due to the nature of feature mapping of CNNs.

In the case of the provided input data, despite there being fewer earthquakes used in the prediction dataset input (six, instead of fifteen in the training input), the PhyCNN algorithm is set up to reformat the prediction input to be the same dimension as the training dataset input.

For example, in the provided input dataset, the input training data set is composed of a 15×3600 tensor of training ground motions, and a $15 \times 3600 \times 2$ tensor for each structural response i.e. displacement, velocity, and acceleration. The PhyCNN develops a feature map of convolving 15×3600 input data to $15 \times 3600 \times 2$ output data. However, the prediction ground motion dataset is 6×3600 and the prediction target structural response $6 \times 3600 \times 2$. Thus, in order to convolve from a 15×3600 tensor to a $15 \times 3600 \times 2$ tensor, the six prediction ground motions are concatenated as needed to fill in the remaining nine rows to be able to convolve correctly. Depending on the dimension of the prediction set relative to the dimension of the training set, the experimenter has to manually adjust the concatenation.

4.3.3 PhyLSTM Setup

Just like the LSTM and PhyCNN model algorithms, the PhyLSTM model algorithms were provided on GitHub by Ruiyang Zhang. along with input data from their respective study. However, unlike the LSTM and PhyCNN, the PhyLSTM study did not provide any experimental validation scenarios of an instrumented structure. The study focused on validating numerical scenarios, like a metamodel of a 3-story moment resisting frame and a Bouc-Wen hysteresis model. Data from the Bouc-Wen numerical validation was provided alongside the Python files containing the algorithms.

Although the PhyLSTM was not readily set up for a study of an instrumented structure, training the model with the same input data as the PhyCNN was straightforward, as the physics-guidance built into the PhyLSTM algorithm structure has similarities to the PhyCNN.

The PhyLSTM was set up to train for 5000 epochs with a learning rate of 0.001. Due to the computational expense of the PhyLSTM, training three separate LSTM networks, only one channel's structural response could be trained at a time on the machine's GPU. The computational expense of the PhyLSTM also limited the data that was able to be included in the prediction dataset to two ground motion records, instead of seven as was originally planned.

4.4 Case Study Formulation

In order to evaluate the performance of the LSTM, PhyCNN, and PhyLSTM in modeling structural response, a case study involving an instrumented structure is formulated. The chosen structure is a 6-story hotel in San Bernardino (CSMIP Station 23287), which was also investigated by Günay et al., Zhang et al., and Liu et al. [2], [6], [8], [11]. Time histories from east-west oriented instruments will be investigated.

4.4.1 Strong-Motion Data Selection

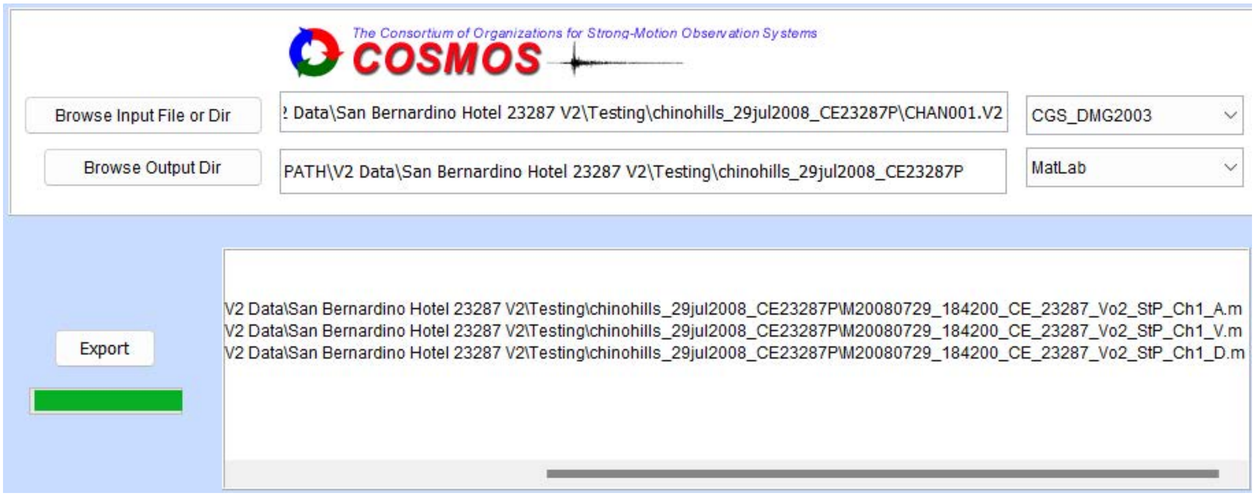
Data selection is an important step of any experiment, and is particularly important in data-driven experiments, like in the case of deep learning, in order to have good quality and consistent output data. In general, deep learning models aim to generalize patterns from the training data to make predictions on new, unseen data. If data is processed in a certain manner, it can have repercussions across the accuracy of a model's predictions.

CSMIP strong-motion data comes in the form of uncorrected acceleration (V1) data and processed (V2) data, which is processed using frequency bands. Since integrating acceleration records with noise results in errors in the velocity and displacement time histories, processed (V2) data has been chosen to be used in this study [2]. From personal communication with Ruiyang Zhang, it was learned that V2 data was originally used in the validation of their deep neural network architectures. Thus, V2 files from the structure, providing displacement, velocity, and acceleration time histories, will be used in this study.

4.4.2 Strong-Motion Data Acquisition

Strong-motion data are downloaded from the CESMD website [18]. The user can search for data using a variety of filters, including by specified station, or a specified earthquake. Additionally, the website offers a map view that shows the location of the station of interest, the type of station (ground, building, etc.), and what network that station is in (USGS, CGS, etc.) via color-coded shapes. The CESMD has thresholds for the inclusion of seismic data, which depend on whether the earthquake is international or domestic to the United States, as well as the seismicity of the region. In the United States, an earthquake is included if the magnitude exceeds 4.5 that has a peak ground acceleration (PGA) of 0.05g or greater [22]. This threshold is lowered to 4.0 in the central and eastern United States, as the these regions are less seismically active than the rest of the country.

When unprocessed V1 data and processed V2 or V3 data from CSMIP stations is downloaded from the CESMD, the data is in a CGS DMG 2003 file format and needs to be converted to a usable format. The COSMOS Converter tool[3], provided by the Consortium of Organizations for Strong Motion Observation Systems (COSMOS), is used to convert strong-motion data from different formats that different geological survey networks use, to usable formats like MATLAB files and comma separated variable files for use in data analysis applications. A screenshot of the GUI of the COSMOS Converter tool is shown in Figure 4.1, showing the conversion of a V2 file to a MATLAB file.


The Consortium of Organizations for Strong-Motion Observation Systems
COSMOS
Browse Input File or Dir
Browse Output Dir
Export

? Data\San Bernardino Hotel 23287 V2\Testing\chinohills_29jul2008_CE23287P\CHAN001.V2
CGS_DMG2003
PATH\V2 Data\San Bernardino Hotel 23287 V2\Testing\chinohills_29jul2008_CE23287P
MatLab

```
V2 Data\San Bernardino Hotel 23287 V2\Testing\chinohills_29jul2008_CE23287P\M20080729_184200_CE_23287_Vo2_StP_Ch1_A.m
V2 Data\San Bernardino Hotel 23287 V2\Testing\chinohills_29jul2008_CE23287P\M20080729_184200_CE_23287_Vo2_StP_Ch1_V.m
V2 Data\San Bernardino Hotel 23287 V2\Testing\chinohills_29jul2008_CE23287P\M20080729_184200_CE_23287_Vo2_StP_Ch1_D.m
```

Figure 4.1: An example of the COSMOS Converter converting a V2 file to MATLAB. Note the input and output file formats on the drop-downs, and the three .m files output from the conversion.

When the V2 file for a particular channel is converted to MATLAB, the displacement, velocity, and acceleration time histories are produced separately in the form of MATLAB .m files. By accessing the MATLAB .m files, the user can access the time history data of the chosen channel and measurement, and also learn about the array configuration of the instrumentation, the digital acquisition unit, the data series, the geolocation, processing information in the case of V2 and V3 data, sensor details, and other information.

Following conversion of the V2 files to MATLAB .m files, the MATLAB .m files need to be converted to MATLAB .mat files to be used as data for future analysis. A separate

[3]This tool is found at https://www.strongmotioncenter.org/COSMOSConverter.htm. The Converter relies on Java, so Java needs to be installed for the appropriate operating system prior to use at https://www.java.com/en/download/.

MATLAB script, provided in this study's GitHub repository, is used to automate the conversion of these .m files to .mat files by tokenizing the channel names and the category of measurement, i.e. based on displacement (D), velocity (V), acceleration (A). Following the file conversion to .mat format, these time histories are then combined and structured into tensors, which is described in more detail in the following section.

The goal at the end of this process is to maintain the highest sampling frequency possible in the data (i.e. having data at the smallest possible time increments). However, not every time history was originally generated at the same sampling frequency. Some events were posted to the CESMD with readings taken every 0.005 seconds (sampled at 200 Hz), while some had less frequent recordings, taken every 0.01 seconds (sampled at 100 Hz). Due to computational constraints, and observing past precedent studies like those of Zhang et al., data used in this study are downsampled to 0.02 seconds (sampled at 50 Hz), and the models make their predictions based on these downsampled data [2], [8].

4.4.3 Input and Output Data Structuring

LSTM

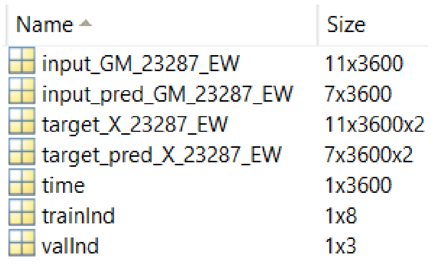
The network from the LSTM algorithm is not physics-based, so only structural displacement time histories are needed to train the model. Thus, a ground channel displacement time history is used as the input to train the displacement time histories from the upper floors of the structure.

For an experiment, an $M \times N$ tensor of training ground motions, consisting of M earthquakes and N evenly-spaced time steps, is constructed for a certain structure and orientation. Not every time history recording from every event has the same length, so records that are too short from a predetermined length by the experimenter are zero-padded at the end to reach N time steps. The training target displacement data consists of an $M \times N \times P$ tensor, where M corresponds to the same earthquakes as in the training ground motion tensor, N corresponds to the same time steps as in the ground motion, and P corresponds to the number of features under investigation, e.g. the number of floors.

In a similar manner, the prediction data are also used to construct tensors, resulting in an $A \times N$ tensor of A prediction input ground motions and an $A \times N \times P$ tensor of A prediction target displacement motions. The value of A does not necessarily have to equal M, as there can be fewer or greater prediction datasets than training datasets, depending on the nature of the study. An illustration of this is shown in Figure 4.2. As previously mentioned, training and validation indices are selected to initialize the training phase of the LSTM. These indices are randomly shuffled as the model trains.

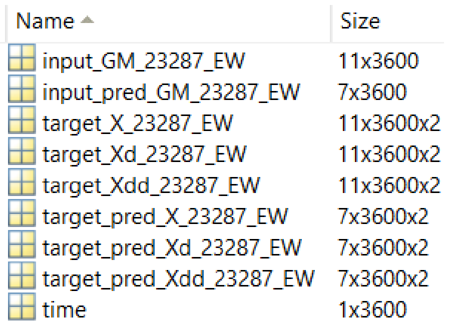
PhyCNN/PhyLSTM

Unlike the LSTM, the PhyCNN and PhyLSTM algorithms leverage physics-guidance. Due to the setup of their respective architectures, they use target velocity and acceleration time his-



Name	Size
input_GM_23287_EW	11x3600
input_pred_GM_23287_EW	7x3600
target_X_23287_EW	11x3600x2
target_pred_X_23287_EW	7x3600x2
time	1x3600
trainInd	1x8
valInd	1x3

(a)



Name	Size
input_GM_23287_EW	11x3600
input_pred_GM_23287_EW	7x3600
target_X_23287_EW	11x3600x2
target_Xd_23287_EW	11x3600x2
target_Xdd_23287_EW	11x3600x2
target_pred_X_23287_EW	7x3600x2
target_pred_Xd_23287_EW	7x3600x2
target_pred_Xdd_23287_EW	7x3600x2
time	1x3600

(b)

Figure 4.2: Example configurations of a) a LSTM algorithm input data file, with 11 training ground motions and 7 prediction ground motions, with 2 degrees of freedom under investigation, and b) a PhyCNN/PhyLSTM algorithm input data file, with 11 training ground motions, 11 target displacements, 11 target velocities, 11 target accelerations, 7 prediction displacements, 7 prediction velocities, 7 prediction accelerations, with 2 degrees of freedom under investigation.

tories for training and prediction. Target velocity and acceleration time histories for training are constructed into $M \times N \times P$ tensors just like their displacement time history counterparts are. Target velocity and acceleration time histories for prediction are constructed into $A \times N \times P$ tensors just like their displacement time history counterparts are.

In the case of the PhyCNN architecture, the input data and prediction data must be of the same dimensions, due to the convolutional operations on the data. Thus, the prediction output and training output are the same size despite there being different numbers of ground motions in each. This is due to the concatenation of the data, as previously demonstrated.

In the case of the PhyLSTM architecture, the analysis is more computationally expensive than the PhyCNN and LSTM, due to the number of networks. There are three interconnected networks, which increases the training time and the size of the data that has to be "remembered."

4.4.4 Result Analysis Metrics and Process

There are a few popular metrics in the literature in evaluating the prediction performance of a deep learning model in predicting seismic structural response. These include the Pearson's r correlation coefficient, the Normalized Error Percentage (NEP), which is assembled into a PDF, and the Peak Error Percentage (PEP).

Correlation Coefficient (r)

The correlation coefficient is a single scalar value calculated from each ground motion prediction's comparison to the ground truth. The correlation coefficient is calculated via the MATLAB 'corr' function, included in the Statistics and Machine Learning Toolbox. A correlation coefficient is calculated for each ground motion record for each degree of freedom, i.e. third floor and roof. Thus, from a training set of M ground motions for P floors of a structure, there are $M \times P$ correlation coefficients. In general, the correlation coefficient shows the linear relationship between the prediction and ground truth. A correlation coefficient closer to 1 indicates a positive relationship in the data and a correlation coefficient closer to -1 indicates an inverse relationship in the data.

Normalized Error Percentage (NEP)

The normalized error percentage is a comparison between the true and predicted values at each time step. Due to the wide-ranging scale of the true values of seismic-induced response for a given ground motion, the error percentage for a time step i is normalized. The greatest magnitude of displacement obtained from all of the true values of each respective quake is the denominator, unlike a regular error percentage calculation. A similar method was used by Zhang et al. and Günay et al. in their studies [2], [7], [8], [11]. The equation to calculate NEP is shown in Equation 4.1, where the i subscript corresponds to a time step along a single time history.

$$NEP_i = \frac{\left| y_i^{\text{pred}} - y_i^{\text{true}} \right|}{\max\left(\left|y_i^{\text{true}}\right|\right)} \times 100\% \tag{4.1}$$

Following the calculations of the NEP for each ground motion in a dataset (training or prediction), the NEPs from each ground motion are concatenated into a single vector of length $M \times N$, where M is the number of ground motions in the set and N is the number of time steps. The MATLAB function 'ksdensity', included in the Statistics and Machine Learning Toolbox, is used to calculate the kernel distributions based on a number of evenly distributed points. The square root rule, commonly used to determine the number of bins for a histogram, is used to determine the number of points. As the name implies, the number of points corresponds to the square root of the number of observations.

Peak Error Percentage (PEP)

The peak error percentage (PEP) is used to compare the single highest magnitude of a response along a ground truth and the highest magnitude in its corresponding prediction time history. Günay et al. used a variation of this metric to record the peak error for a given time history [11]. However, instead of leaving the error as a decimal form, it is converted to a percentage in this study. Equation 4.2 shows the calculation of the PEP as it is used in this study.

$$PEP = \frac{\max|y^{\text{pred}}| - \max|y^{\text{true}}|}{\max|y^{\text{true}}|} \times 100\% \tag{4.2}$$

Unlike the NEP, which is eventually assembled into a PDF to determine the accuracy of the entire time history, the PEP only evaluates a single maximum point for the measured time history and for the prediction time history for each degree of freedom. Similar to the correlation coefficient, for M time histories and P degrees of freedom, there are are $M \times P$ values of PEP.

Chapter 5

Case Study: 6-Story Hotel in San Bernardino

The structure of interest for the case study in evaluating the performance of these algorithms is the 6-story hotel in San Bernardino (CSMIP Station 23287) which was previously studied by Zhang et al., Günay et al., and Liu et al. [2], [6], [8], [11]. The hotel has a reinforced concrete shear wall (RCSW) system, which is a common and effective structural system for mid-rise buildings in seismic-prone regions [23]. Although the hotel was designed in 1970, it was not instrumented until 1976, following the establishment of CSMIP. The locations of the instruments inside the structure are shown in Figure 5.1. Sensors 1, 4, and 7, measuring east to west will be used in the case study. Sensor 1 is used for the ground motions, sensor 4 is used for the third floor readings, and sensor 7 is used for the roof readings. Their central location inside the structure gives a more complete idea of the structure's movements during these seismic events. Street-view and satellite imagery of the structure is shown in Figure 5.2 to give the reader a better idea of the geometry of the structure.

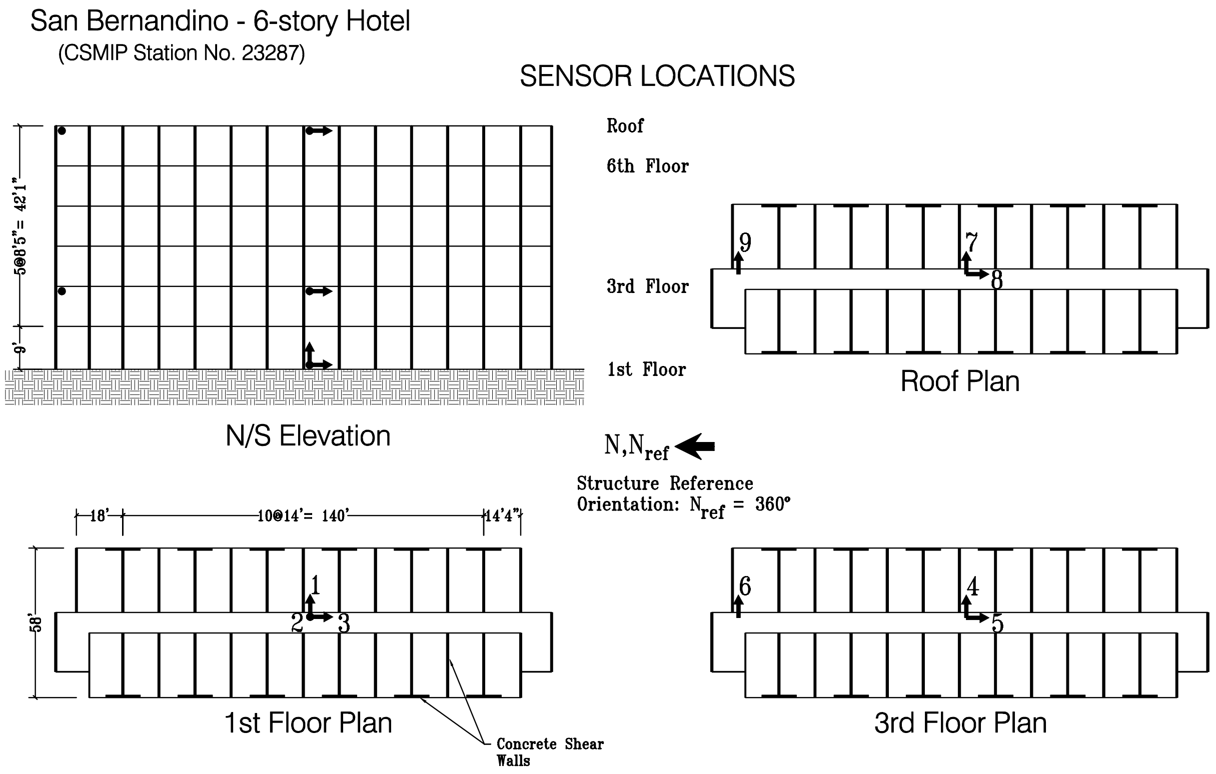


Figure 5.1: Sensor layout of the 6-story hotel, provided by CESMD.

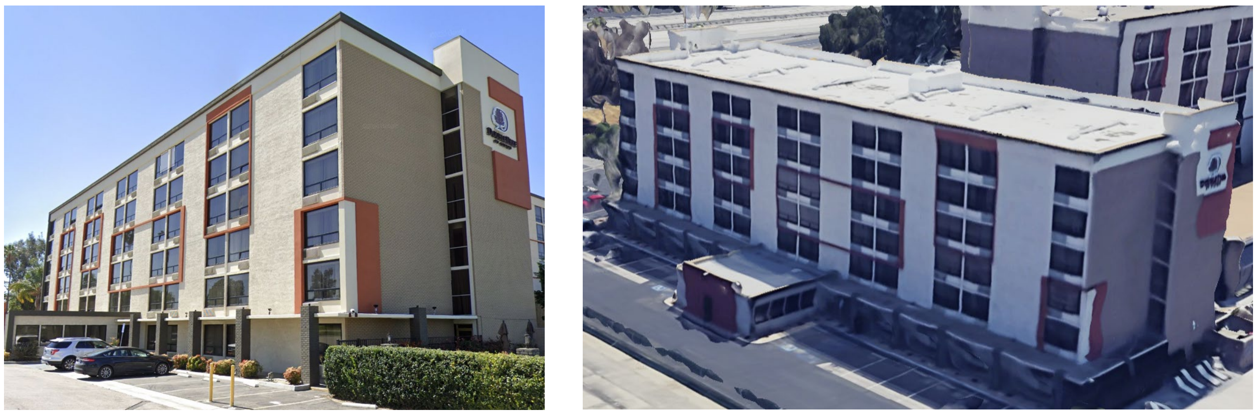


Figure 5.2: Isometric views of the 6-story hotel, provided by Google Maps.

5.1 Training and Prediction Record Selection

The earthquakes shown in Table 5.1 were chosen to be the training dataset and the earthquakes shown in Table 5.2 were chosen to be the prediction dataset. As Günay et al. demonstrated, the selected training responses cover the entire recorded range of shaking the building has experienced [11]. This is further illustrated by the differences in peak ground acceleration (PGA) and peak floor acceleration (PFA). These 11 training events were also part of the 15 training events used in Zhang et al.'s studies [2], [8]. Figure 5.3 shows the plot of PFA in the east-west (EW) direction with respect to the PGA in the east-west direction of each of the training and testing ground motions.

Table 5.1: Earthquake Training Records

#	**Earthquake Name and Date**	**PGA EW (g)**	**PFA EW (g)**
1	Borrego Springs Area Earthquake of 07 Jul 2010	0.024	0.045
2	Devore Earthquake of 29 Dec 2015	0.054	0.121
3	Fontana Earthquake of 15 Jan 2014	0.034	0.044
4	Inglewood Area Earthquake of 17 May 2009	0.008	0.016
5	Ocotillo Area Earthquake of 14 Jun 2010	0.006	0.014
6	San Bernardino Earthquake of 08 Jan 2009	0.094	0.219
7	Beaumont Earthquake of 14 Sep 2011	0.027	0.064
8	La Habra Earthquake of 28 Mar 2014	0.024	0.077
9	Loma Linda Earthquake of 13 Mar 2017	0.025	0.038
10	Ontario Earthquake of 20 Dec 2011	0.004	0.014
11	Yorba Linda Earthquake of 07 Aug 2012	0.003	0.010

Table 5.2: Earthquake Prediction Records

#	**Earthquake Name and Date**	**PGA EW (g)**	**PFA EW (g)**
1	Loma Linda Earthquake of 04 Mar 2013	0.012	0.032
2	Chino Hills Earthquake of 29 July 2008	0.036	0.117

Due to the computational constraints imposed by the PhyLSTM training, only two prediction records are used: the Loma Linda Earthquake of March 4, 2013 and the Chino Hills Earthquake of July 29, 2008. In their study, Günay et al. also used both of these records in their testing data, and out of that testing data set, these two records contained the two highest values of PGA and PFA among all floors in the east-west direction, which is the direction analyzed in this study [11]. Table 5.2 shows the PGA and PFA of these two events. Upon inspection of the frequency contents of the displacement time histories, it can be observed that these two events have different frequency characteristics. The Loma Linda earthquake, shown in Figure 5.4, tends to have higher frequency contents than the Chino Hills earthquake shown in Figure 5.5. Having prediction data with different frequency contents and different magnitudes of peak responses to test the accuracy of models will demonstrate the generalizability of the deep learning models, which is valuable if they are to be used in widespread applications.

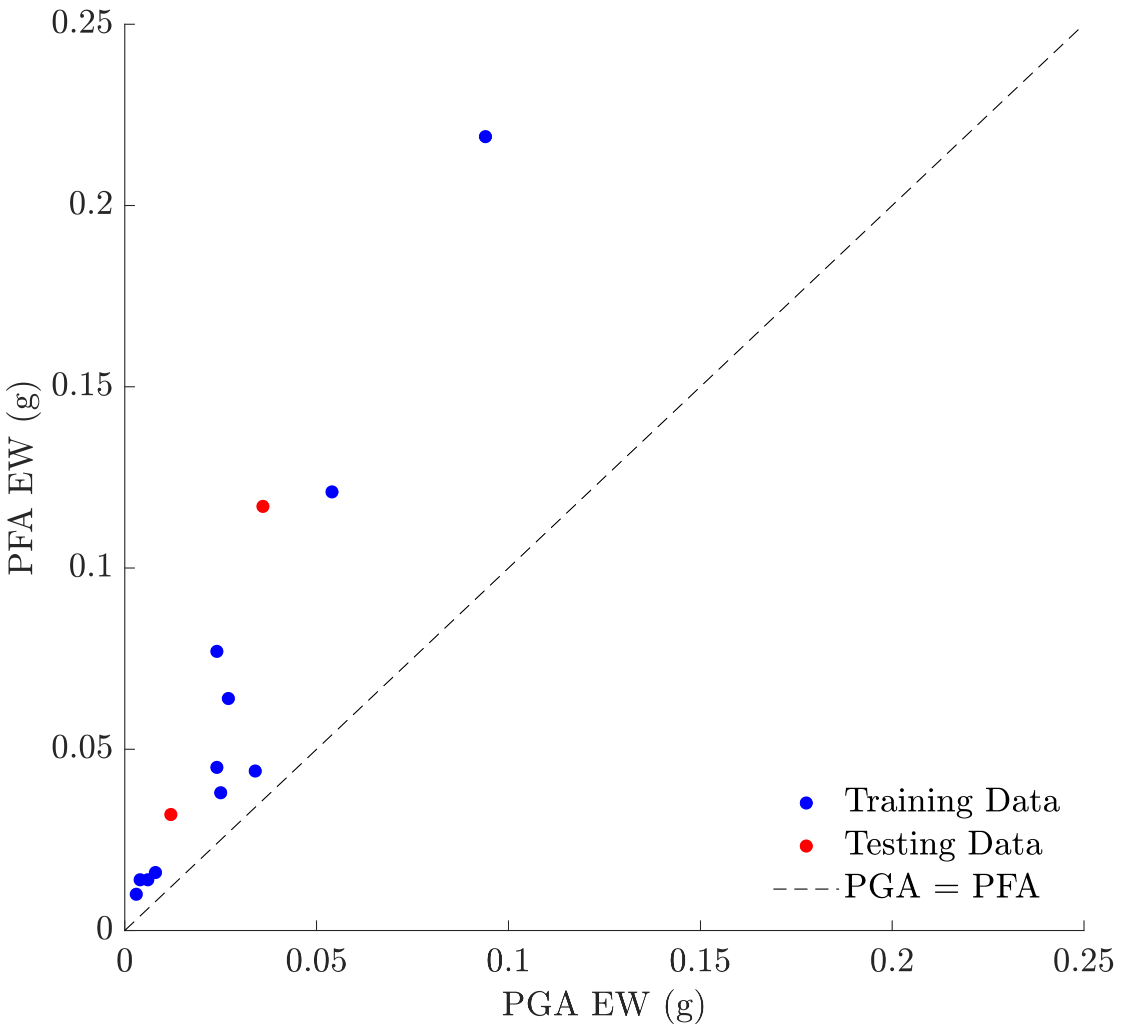
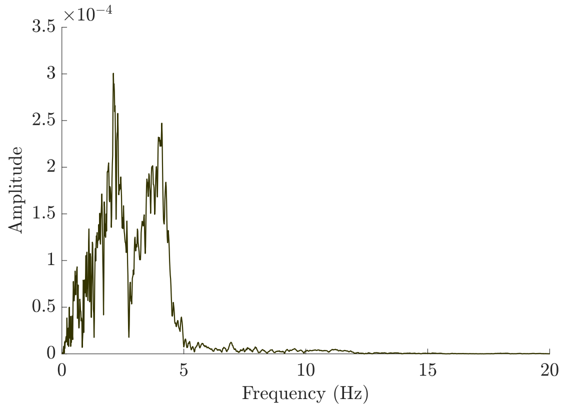
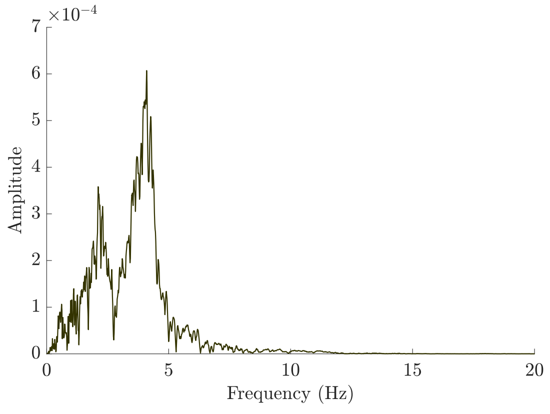


Figure 5.3: Training and testing records used in the case study. Note that PFA > PGA for every event.

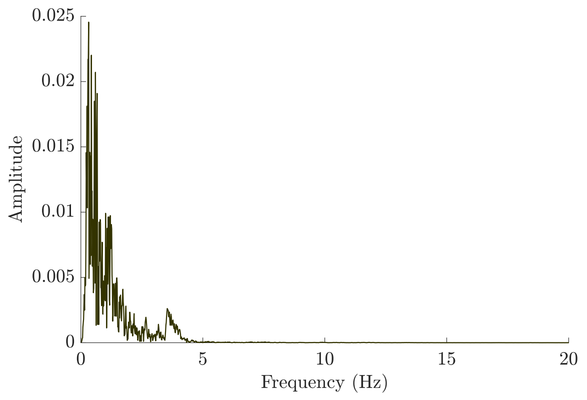


(a)

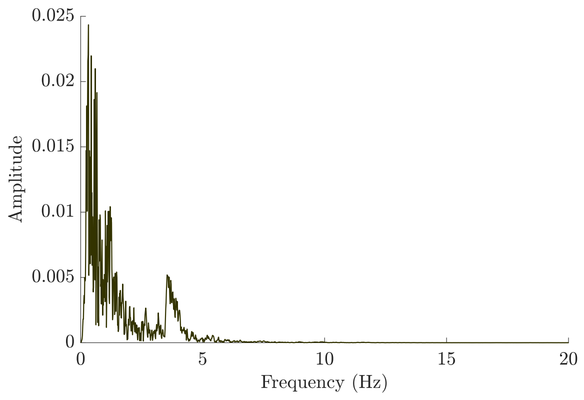


(b)

Figure 5.4: FFT results from the ground truth of a) the third floor displacement time history and b) the roof displacement history from the Loma Linda Earthquake of March 4, 2013.



(a)



(b)

Figure 5.5: FFT results from the ground truth of a) the third floor displacement time history and b) the roof displacement history from the Chino Hills Earthquake of July 29, 2008.

Chapter 6

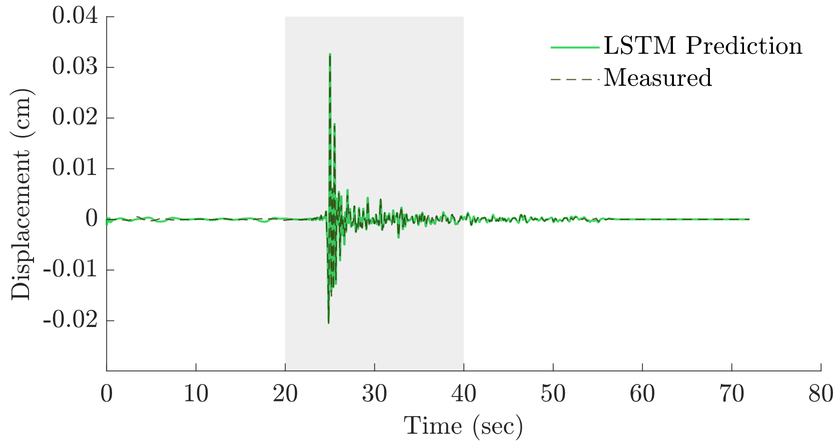
Results

6.1 Comparison of Third Floor and Roof Predictions

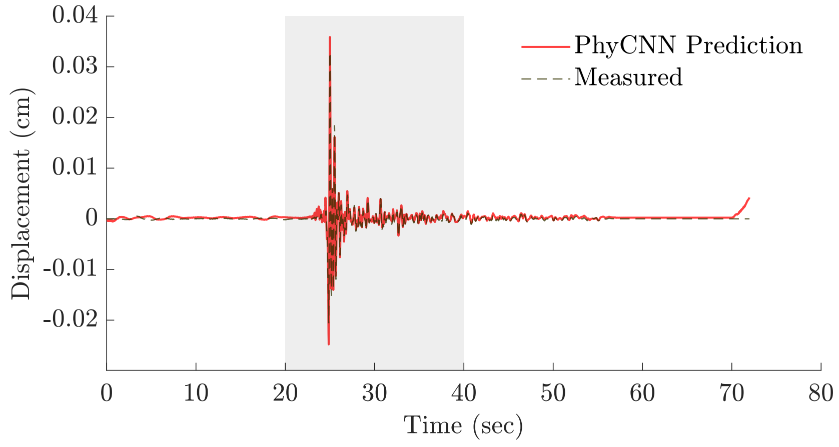
The predictions from each algorithm for both the third floor and roof responses from the two testing records, shown in Table 5.2, are plotted against their respective measured response values. For the third floor, displacement predictions from the Loma Linda earthquake of March 4, 2013 are shown in Figure 6.1 and the displacement predictions from the Chino Hills earthquake of July 29, 2008 are shown in Figure 6.3. For the roof, displacement predictions from the Loma Linda earthquake of March 4, 2013 are shown in Figure 6.5 and the displacement predictions from the Chino Hills earthquake of July 29, 2008 are shown in Figure 6.7.

For each of these figures (Figures 6.1, 6.3, 6.5, and 6.7) a gray region highlights a 20-second time window where the primary ground motion occurs: between 20-40 seconds for the Loma Linda earthquake, and between 15-35 seconds for the Chino Hills earthquake. These time windows aim to show a more focused view of the prediction accuracy of the LSTM, PhyCNN, and PhyLSTM. Predictions from these time windows are plotted separately in Figures 6.2 and 6.6 for the Loma Linda earthquake 3rd floor and roof displacement predictions, respectively, and Figures 6.4 and 6.8 for the Chino Hills earthquake for 3rd floor and roof displacement predictions, respectively. Qualitatively, good agreement can be found between the original measurements and the algorithms' predictions, which necessitates the need to use quantitative metrics to evaluate the accuracy of the predictions.

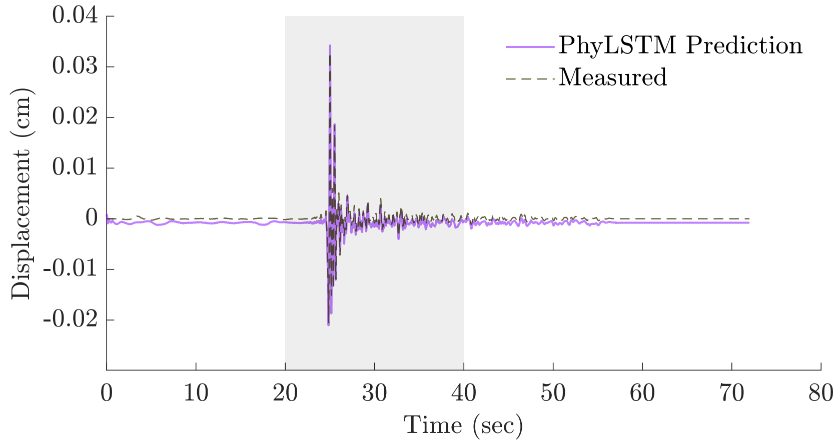
In both testing records, the third floor peak displacements were smaller than those of the peak roof displacements. This suggests that the structure could be in its fundamental mode shape throughout both events.



(a) LSTM Prediction

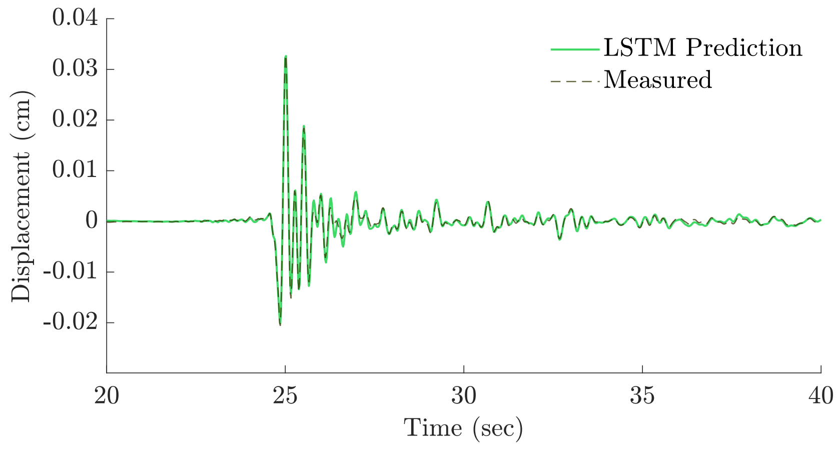


(b) PhyCNN Prediction

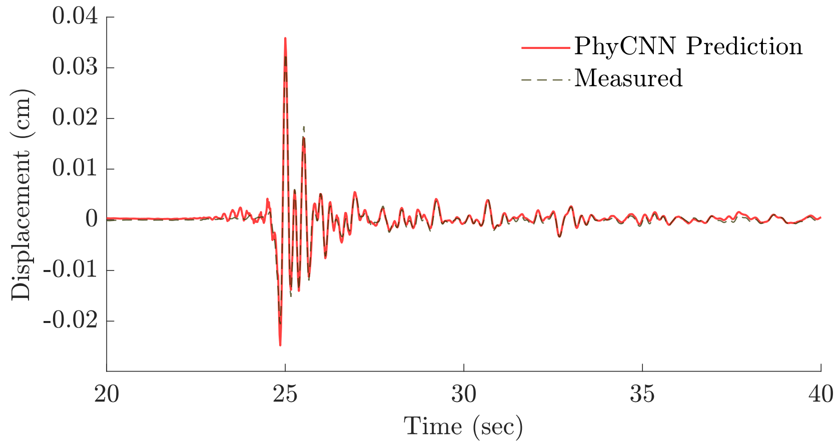


(c) PhyLSTM Prediction

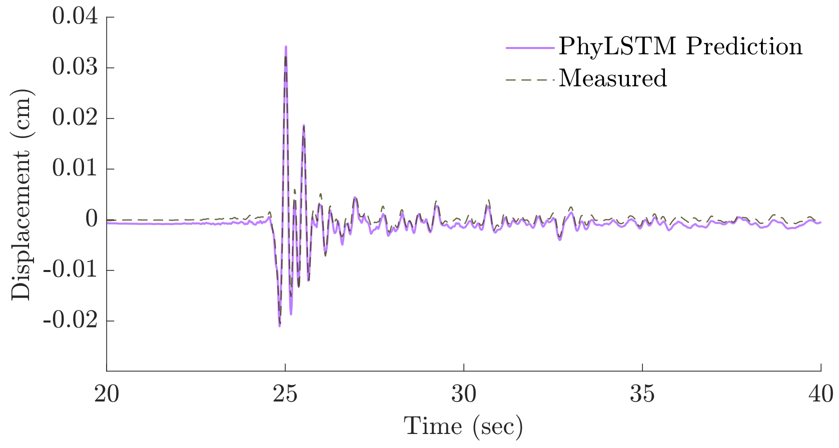
Figure 6.1: 3rd floor east-west displacement predictions for Loma Linda Earthquake of March 4, 2013.



(a) LSTM Prediction

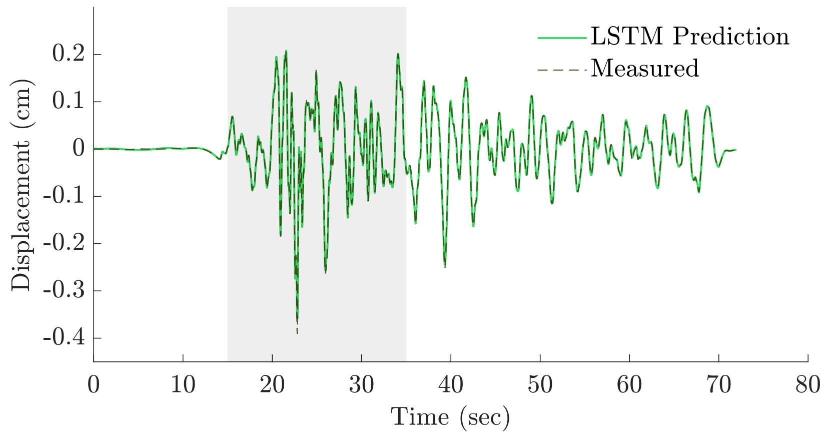


(b) PhyCNN Prediction

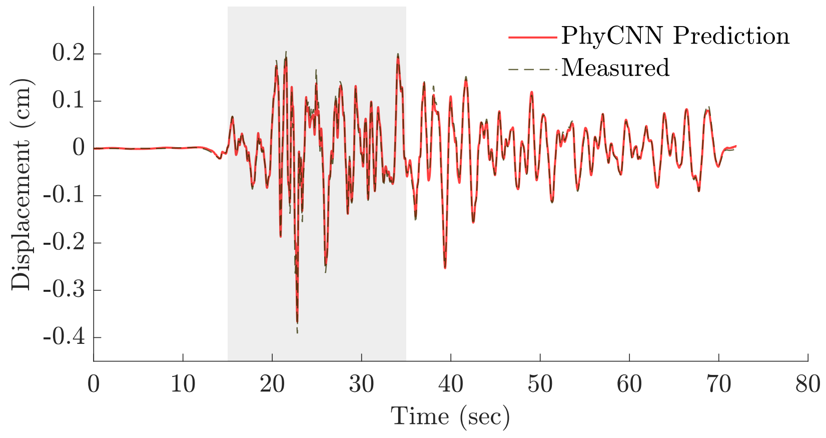


(c) PhyLSTM Prediction

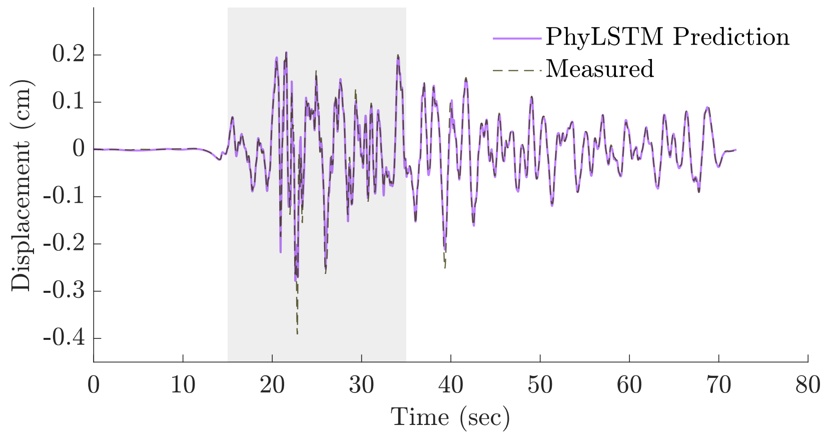
Figure 6.2: Time window of interest for 3rd floor east-west displacement predictions for Loma Linda Earthquake of March 4, 2013.



(a) LSTM Prediction

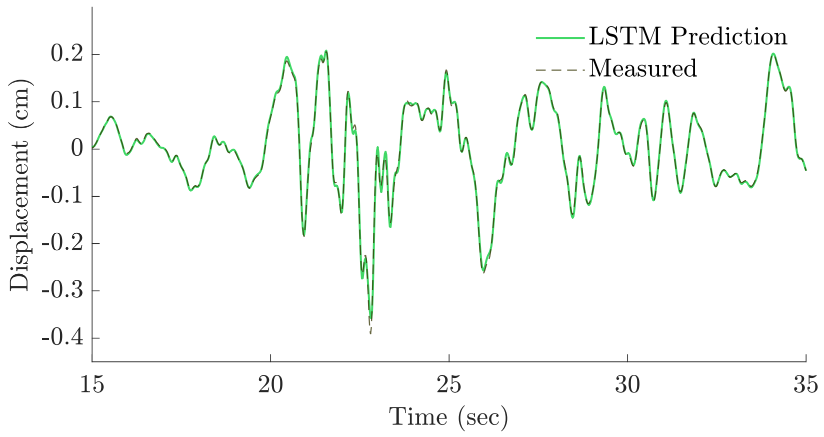


(b) PhyCNN Prediction

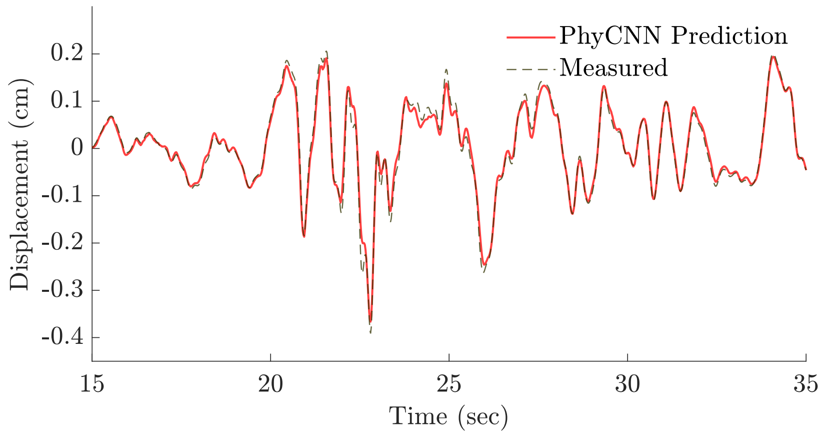


(c) PhyLSTM Prediction

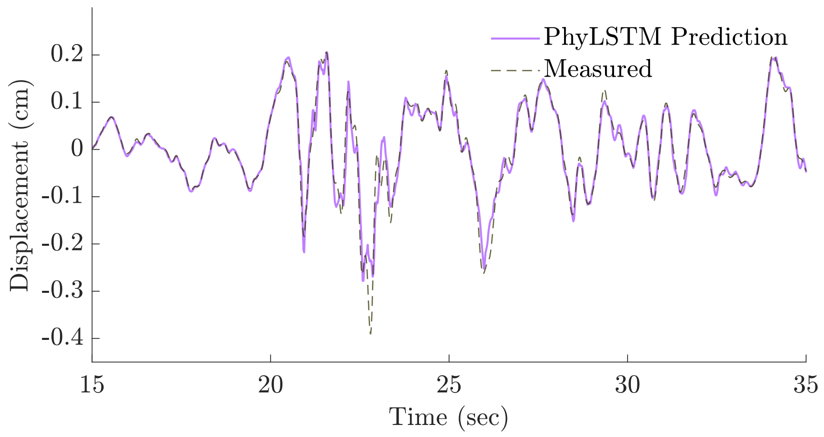
Figure 6.3: 3rd floor east-west displacement predictions for Chino Hills Earthquake of July 29, 2008.



(a) LSTM Prediction

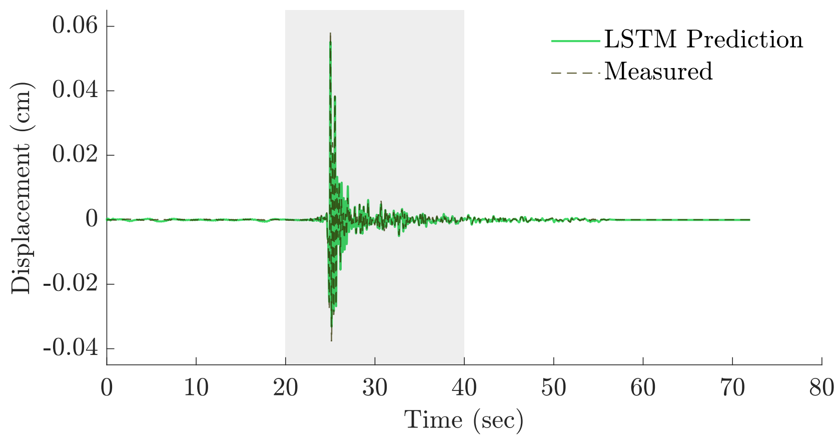


(b) PhyCNN Prediction

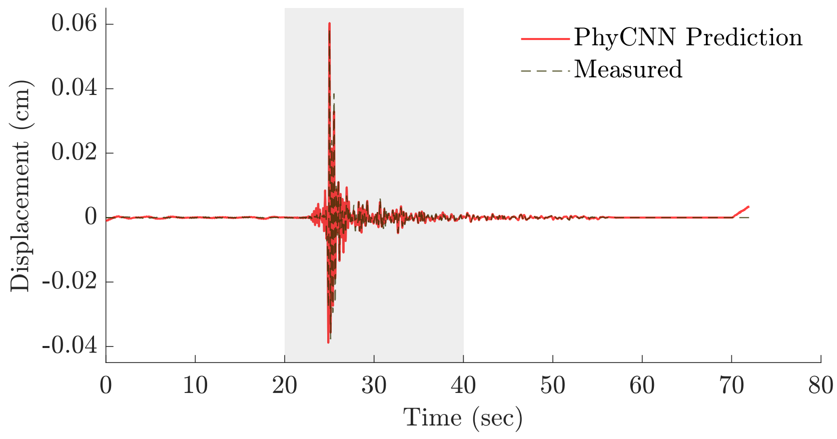


(c) PhyLSTM Prediction

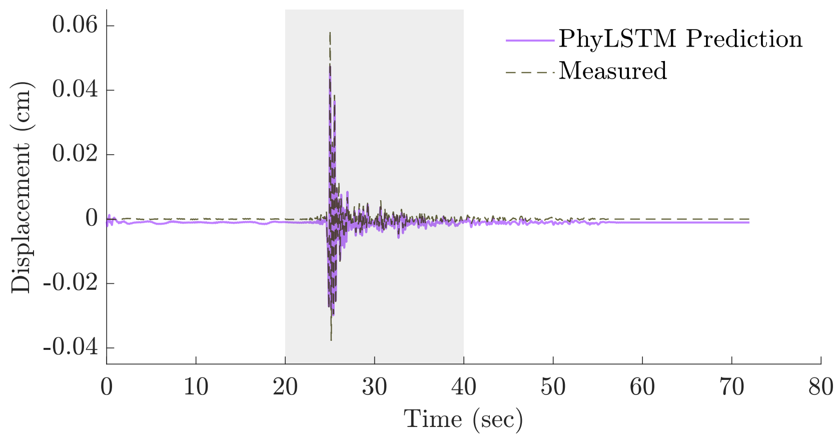
Figure 6.4: Time window of interest for 3rd floor east-west displacement predictions for Chino Hills Earthquake of July 29, 2008.



(a) LSTM Prediction

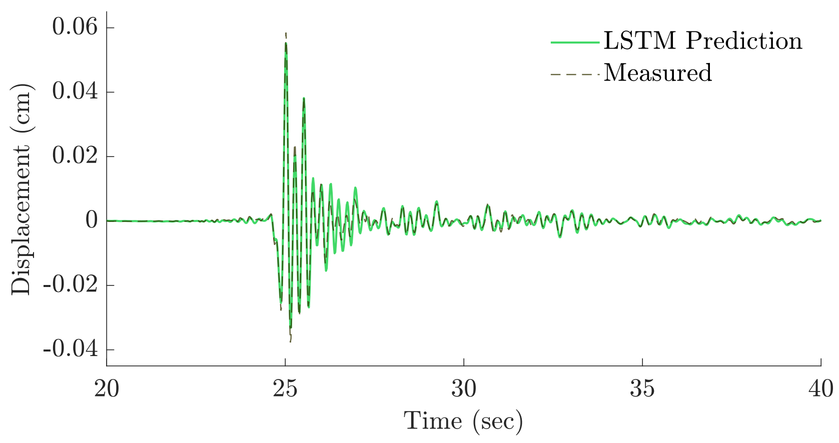


(b) PhyCNN Prediction

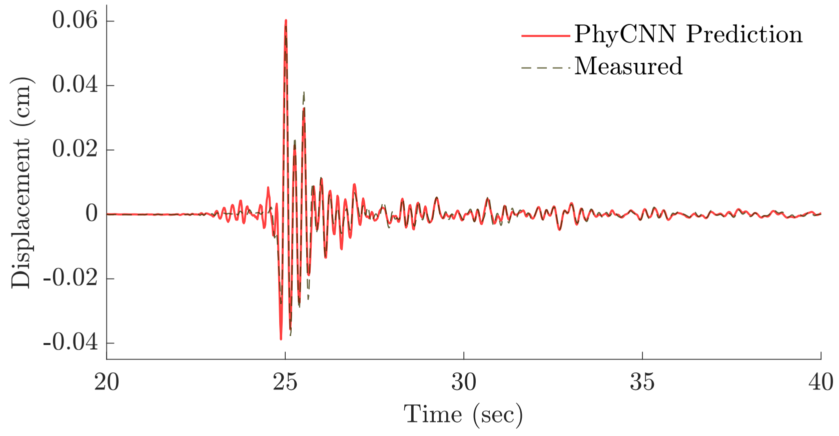


(c) PhyLSTM Prediction

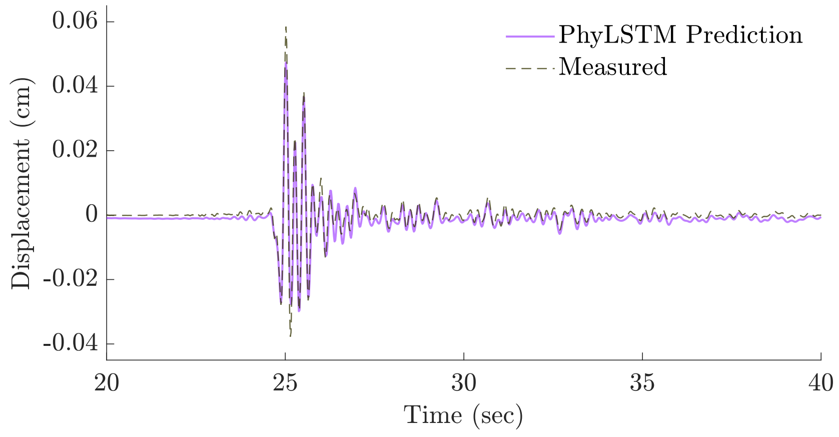
Figure 6.5: Roof east-west displacement predictions for Loma Linda Earthquake of March 4, 2013.



(a) LSTM Prediction

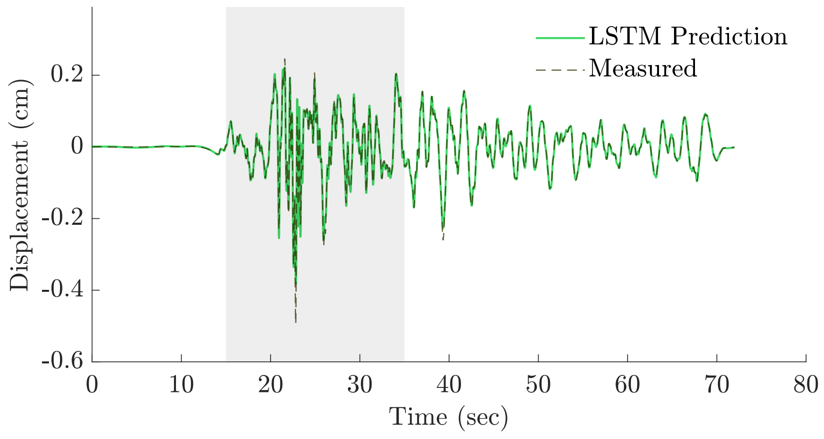


(b) PhyCNN Prediction

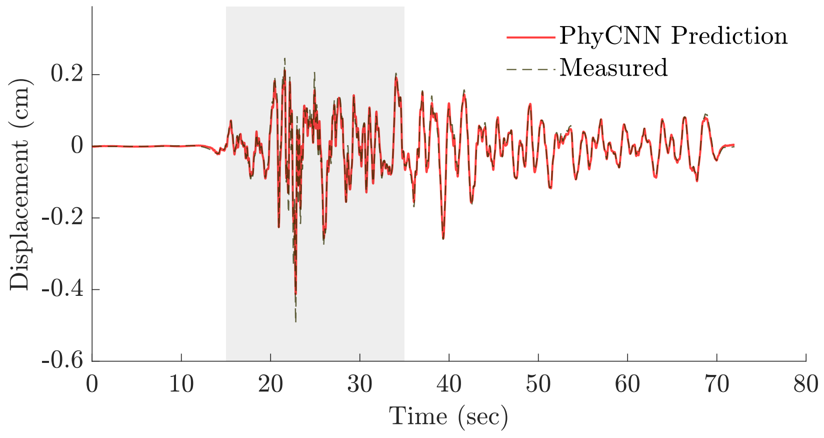


(c) PhyLSTM Prediction

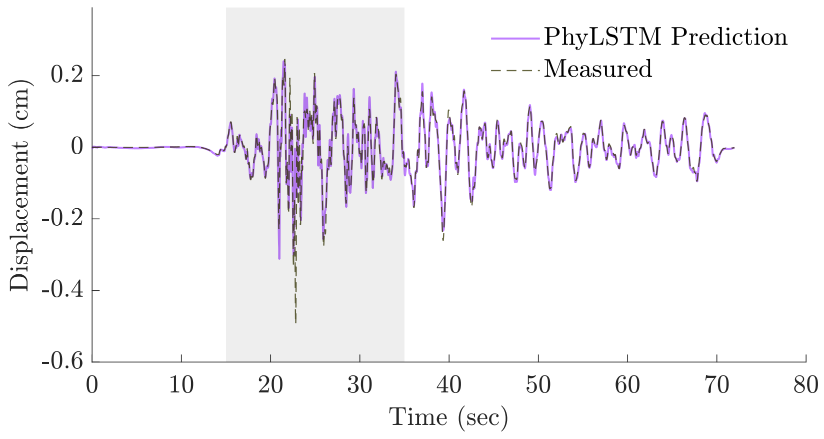
Figure 6.6: Time window of interest for roof east-west displacement predictions for Loma Linda Earthquake of March 4, 2013.



(a) LSTM Prediction

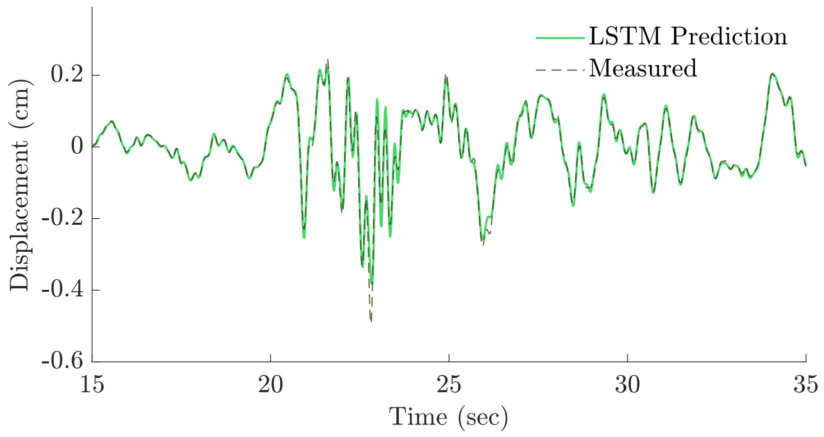


(b) PhyCNN Prediction

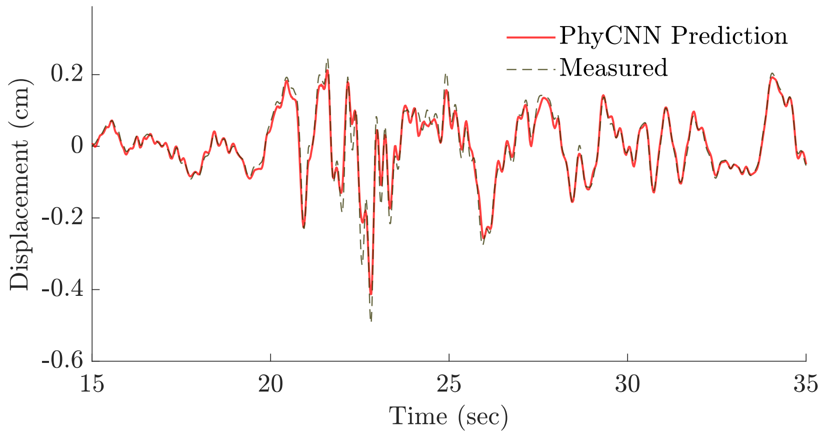


(c) PhyLSTM Prediction

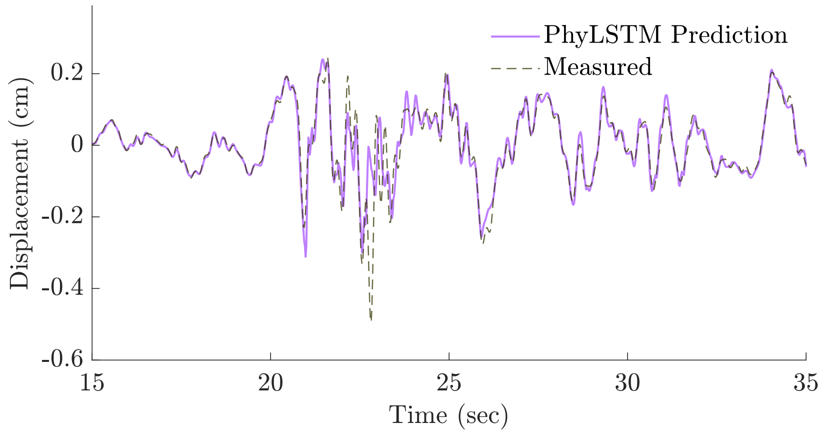
Figure 6.7: Roof east-west displacement predictions for Chino Hills Earthquake of July 29, 2008.



(a) LSTM Prediction



(b) PhyCNN Prediction



(c) PhyLSTM Prediction

Figure 6.8: Time window of interest for roof east-west displacement predictions for Chino Hills Earthquake of July 29, 2008.

6.2 Correlation Coefficients

Correlation coefficients were calculated for each training and testing record, for each floor. A total of 22 correlation coefficients were obtained from the training data for each architecture, 11 for the third floor and 11 for the roof. The means of these are shown in Table 6.1 and medians of these are shown in Table 6.2. A total of 4 were obtained from the testing data for each architecture, 2 for the third floor and 2 for the roof. Unlike with the training data, the results from the testing data are separated by particular event so that insights can be gained based on the characteristics of each of the testing records. Correlation coefficients for the test records are shown in Table 6.3.

For each of the test records at each respective floor, the LSTM model has better correlation coefficients than the other two algorithms. Günay et al. remarked in their study that the V2 data lost some of its physical meaning through the usage of data processing techniques, like baseline correction and filtering [11]. This result may be related to this fact. The physics-based models, though modeling the main event portion well from a qualitative standpoint by inspection of the plot, experienced difficulties at times in modeling the pre-event and zero-padded portions of time histories. The LSTM model's purely empirical predictions are not affected to as much of an extent, as differentiation is not built into its loss calculation. When the LSTM sees zero-padding for example, it appears to learn this pattern, unlike the PhyCNN predictions which experienced an upward curve at the end of the Loma Linda time histories. However, it is of note that the mean and median correlation coefficients of the PhyCNN's training data predictions for the roof are higher than those of both the LSTM and PhyLSTM.

The data are visualized in Figure 6.9 via box and whisker plots, showing the distribution of correlation coefficients from the training data, and points marking where each testing data point lies. A graphical trend that can be observed is the widening of the interquartile range (IQR) of the LSTM and PhyLSTM from the 3rd floor to the roof, and a narrowing of the IQR for the PhyCNN from the 3rd floor to the roof, reinforcing the argument that the PhyCNN may consistently predict the roof better than the 3rd floor. Another trend that the graph highlights is that the correlation coefficients for all of the Chino Hills earthquake predictions except in the case of PhyLSTM roof are higher than those of the Loma Linda earthquake, which has a larger share of higher frequency contents, as shown in the FFTs in Figures 5.4 and 5.5.

Table 6.1: Mean Correlation Coefficients of Training Data Predictions

	LSTM	PhyCNN	PhyLSTM
3rd Floor	0.9874	0.9744	0.9805
Roof	0.9791	0.9866	0.9790

Table 6.2: Median Correlation Coefficients of Training Data Predictions

	LSTM	PhyCNN	PhyLSTM
3rd Floor	0.9977	0.9958	0.9966
Roof	0.9958	0.9964	0.9949

Table 6.3: Correlation Coefficients of Test Data Predictions

	LSTM		PhyCNN		PhyLSTM	
	Loma Linda	Chino Hills	Loma Linda	Chino Hills	Loma Linda	Chino Hills
3rd Floor	0.9858	0.9991	0.9668	0.9938	0.9765	0.9845
Roof	0.9734	0.9930	0.9559	0.9873	0.9719	0.9398

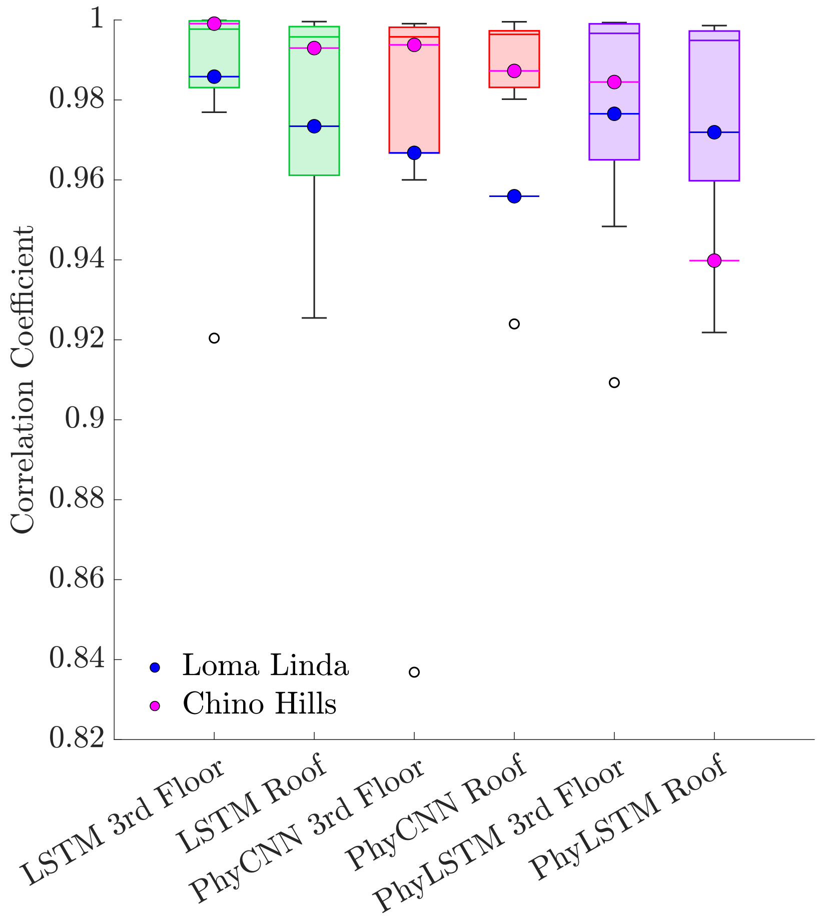
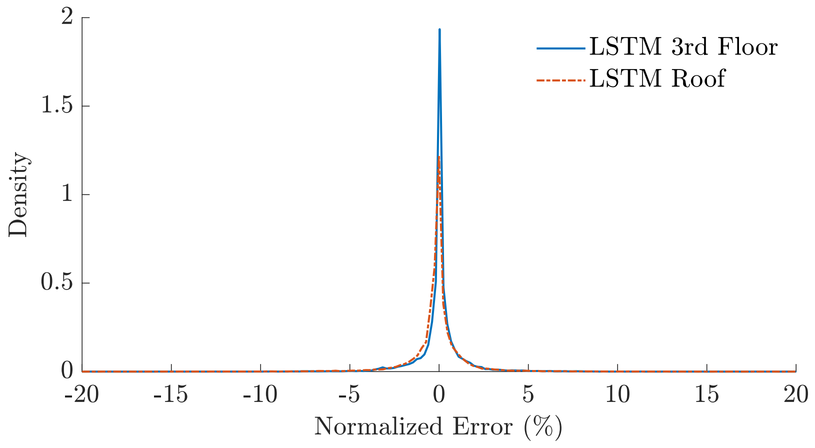


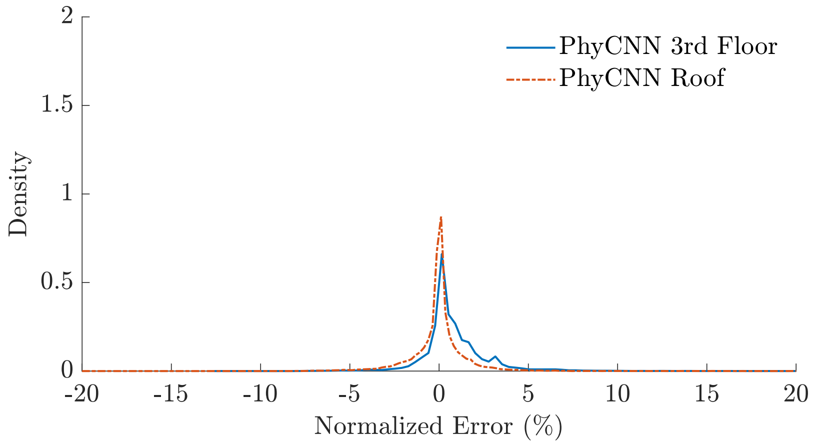
Figure 6.9: Box and whisker plots of correlation coefficients of training dataset predictions with test prediction results overlaid.

6.3 Normalized Error Percentages (NEP)

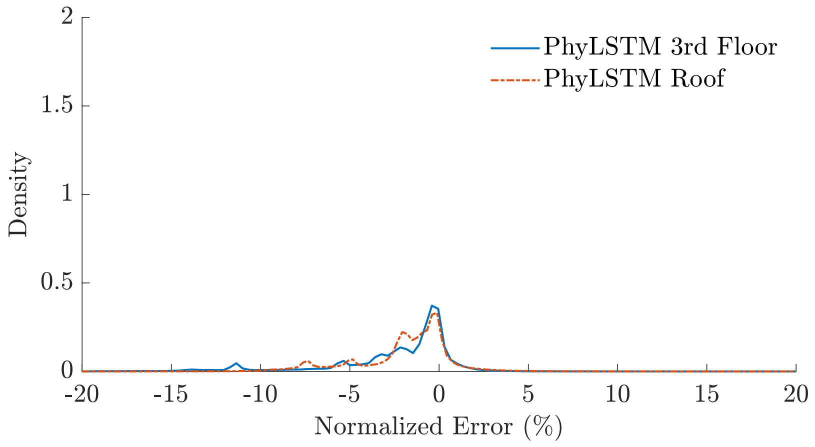
The NEPs from each time history were assembled into PDFs, with distributions shown in Figure 6.10 for the training data, and distributions shown in Figure 6.11 for the testing data. The purpose of the NEP metric is to show the general trend of how well the model predicts the magnitude of the data across each individual time history. In the training data case, it can be observed that the LSTM and PhyLSTM have more density centered at 0% error for the third floor than the roof, whereas this is the opposite in the case of the PhyCNN, of which the roof seems to be more dense at 0% error. Zhang et al. also achieved similar results in their LSTM and PhyCNN studies [2], [8]. Because of this reverse in trend for the PhyCNN, it may be related to the computational mechanics of how CNN layers process data as opposed to LSTM layers. In the testing datasets, as two ground motions were investigated, the effects of the patterns shown in the displacement prediction plots, like the PhyLSTM's negative offset, are visible in the NEP distributions. In both the training and testing datasets, the LSTM has the highest concentration of density at 0% error of the three algorithms for both the 3rd floor and roof.



(a) LSTM

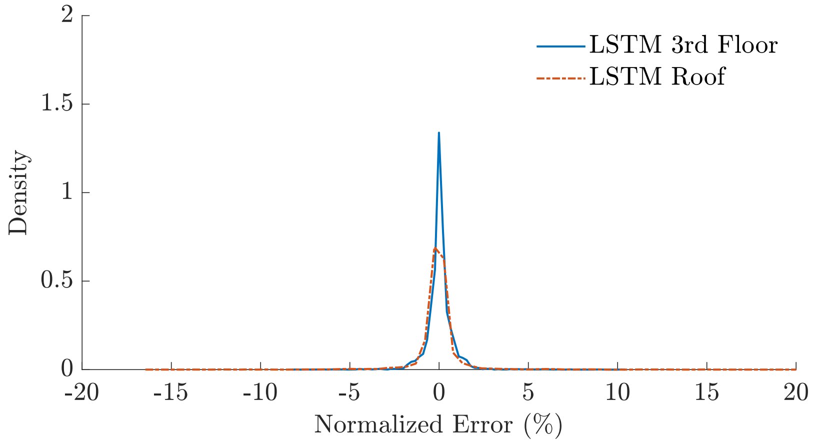


(b) PhyCNN

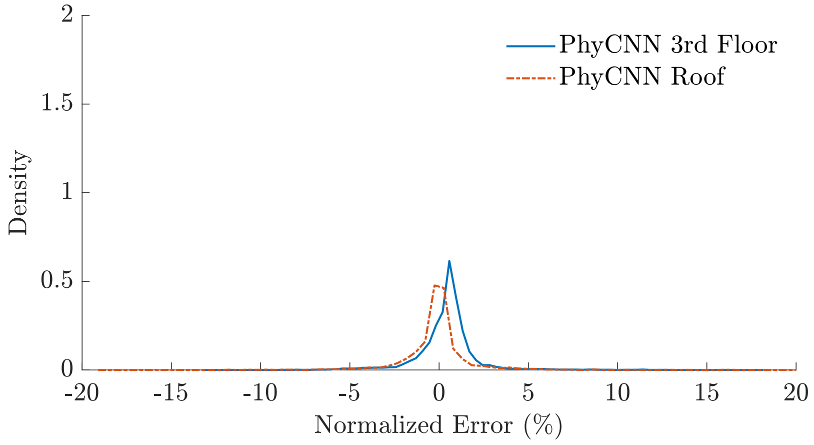


(c) PhyLSTM

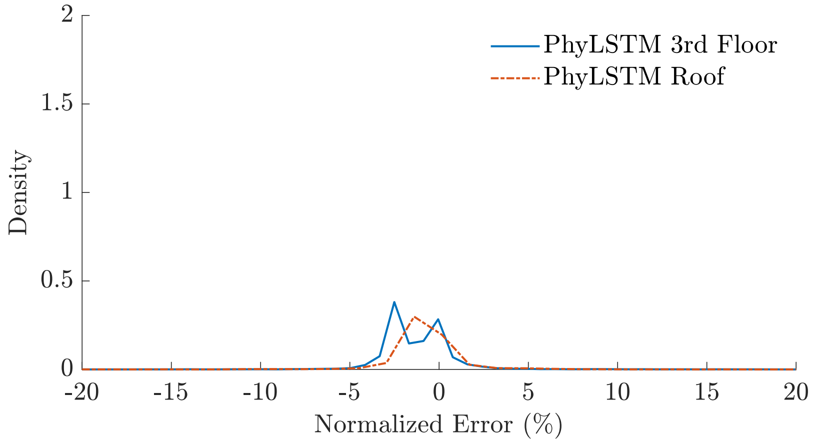
Figure 6.10: Normalized error distribution of all training datasets.



(a) LSTM



(b) PhyCNN



(c) PhyLSTM

Figure 6.11: Normalized error distribution of all testing datasets.

6.4 Peak Error Percentages (PEP)

Peak error percentages (PEP) were calculated for each training and testing record, for each degree of freedom. A total of 22 peak error percentages were obtained from the training data for each architecture, 11 for the third floor and 11 for the roof. The means of these are shown in Table 6.4 and medians of these are shown in Table 6.5. A total of 4 were obtained from the testing data for each architecture, 2 for the third floor and 2 for the roof. Unlike with the training data, the data from the testing data is separated by particular event so that insights can be gain based on the characteristics of each of the testing records, similar to how was carried out previously with the correlation coefficient section. Peak errors from the testing records are shown in Table 6.6.

The box and whisker plot in Figure 6.12 shows the distribution of the peak error percentages from the training data, and points marking where each testing data point lies. While the algorithms had larger correlation coefficients for the Chino Hills earthquake predictions relative to the Loma Linda earthquake predictions, the algorithms had more favorable peak errors for the Loma Linda earthquake than the Chino Hills earthquake. This differing trend in results could be related to the frequency content of the ground motions. As shown in the results tables, peak errors from the roof predictions tended to increase in magnitude relative to the peak errors from the third floor predictions, likely due to more severe and abrupt spikes than seen on the third floor. An example of this is particularly noticeable in Figure 6.7 with the Chino Hills earthquake, of which the magnitudes of all of the predictions underestimated the peak magnitude, as shown by the negative error.

Table 6.4: Mean Peak Error Percentage of Training Data Predictions

	LSTM	PhyCNN	PhyLSTM
3rd Floor	-1.29%	1.24%	3.39%
Roof	-3.80%	3.42%	-4.31%

Table 6.5: Median Peak Error Percentages of Training Data Predictions

	LSTM	PhyCNN	PhyLSTM
3rd Floor	-0.48%	1.70%	1.78%
Roof	-0.90%	3.06%	-2.88%

Table 6.6: Peak Error Percentage of Test Data Predictions

	LSTM		PhyCNN		PhyLSTM	
	Loma Linda	Chino Hills	Loma Linda	Chino Hills	Loma Linda	Chino Hills
3rd Floor	-0.42%	-8.52%	9.42%	-6.36%	4.39%	-28.93%
Roof	-5.68%	-22.59%	3.18%	-16.40%	-19.06%	-36.93%

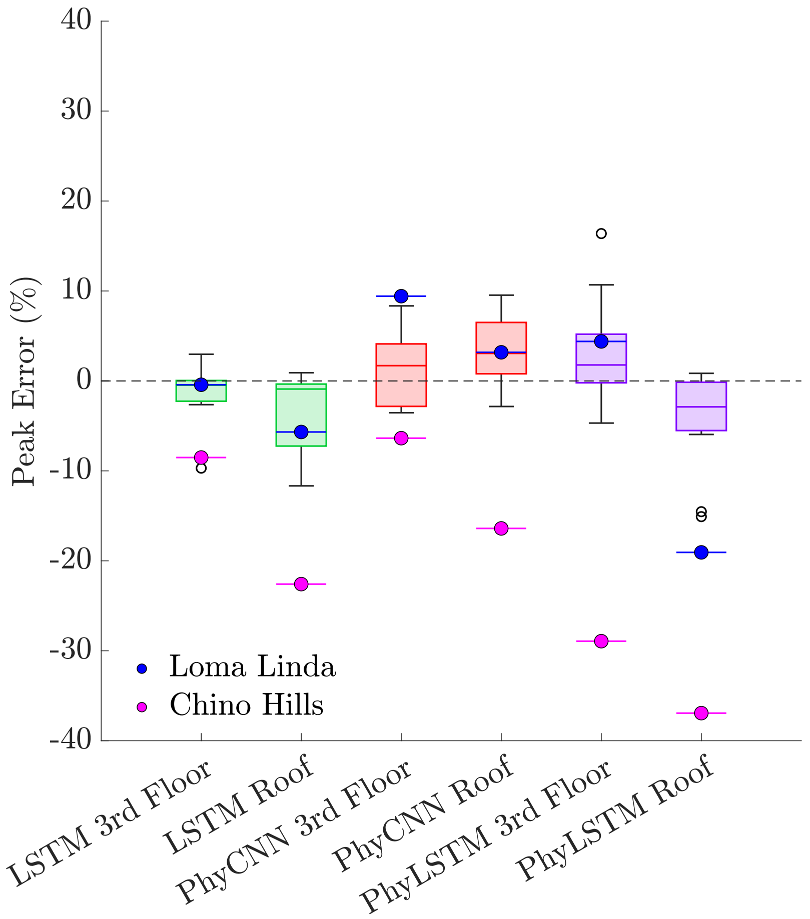


Figure 6.12: Box and whisker plots of PEPs of training dataset predictions with test prediction results overlaid. Note that negative error indicates an underestimation of the magnitude of the peak, whereas a positive error indicates an overestimation of the magnitude of the peak.

Chapter 7

Conclusions

7.1 Summary of Contributions

This work compared the performance of three popular deep learning networks in predicting the seismic response of an instrumented mid-rise reinforced concrete shear wall hotel using three different results metrics: correlation coefficients, normalized error percentages, and peak error percentages. Correlation coefficients measured the linear relationship between the prediction and ground truth; normalized error percentages measured the differences at each time step, normalized by the maximum value of that time history, and assembled into a PDF; and the peak error percentage measured the percent error between the peak value of the prediction and ground truth. These metrics were chosen due to prior usage in studies evaluating prediction accuracy of deep learning networks.

As deep learning becomes more and more common in structural engineering, academics and practitioners using deep learning will need to have an understanding of how the data is processed in deep learning algorithms, in addition to needing an understanding of structural dynamics. This understanding of how the data is processed begins even before the deep learning architecture of choice is deployed. In reality, it starts with the data used in the model, and the processing, if any, it has undergone.

7.2 Study Difficulties and Limitations

7.2.1 Study Difficulties

As this study was conducted, there were several difficulties encountered, particularly when setting up the virtual environment and successfully running the algorithms with the provided data. It is hoped that the inclusion of external resources on GitHub can alleviate some of these difficulties in future experiments with these particular algorithms.

In order to create the virtual environment, the dependency versions were determined by trial and error. Although redeploying the models using a more updated framework e.g. TensorFlow 2.x or PyTorch was considered, it was determined to be out of scope and can be future work. Replicating the experimental methodology also came about via a trial and error approach, as the workflow in accessing the CESMD website, downloading and converting the strong-motion data to a usable format is not well-published in the literature alongside other deep learning studies in seismic response prediction.

7.2.2 Limitations

A limitation of this study is that one experiment, with one corresponding set of training and testing data for a single structure, was conducted. Usage of different ground motion records for training may affect the prediction accuracy of each deep learning model. For example, if the training data of an experiment is only composed of linear structural responses, and the deep learning model is expected to predict a nonlinear structural response, the data-driven and physics-guided methodologies may have different accuracy than this study which used testing data with lower PGA and PFA than some of the training records.

Another limitation of this study is that the LSTM, PhyCNN, and PhyLSTM algorithms were compiled using TensorFlow 1.x, which at the time of their respective publication, was state-of-the-art, but that version is now outdated. In the future, deep learning networks for widespread usage will need to be deployed using up-to-date frameworks. As machine learning, particularly deep learning, is a popular research area, it is anticipated that there will continue to be advancements in novel architectures and computational efficiency.

The computational power of the off-the-shelf laptop used in the study was limited. Due to these limits, the sampling frequencies and training and testing data sizes were limited, particularly for the PhyLSTM, and the sampling frequencies for each time history had to be downsampled from the provided CESMD data.

7.3 Findings and Potential Impact

From examining the results of these three models, there is no clear winner or standard approach to achieve the best results across the three results metrics. In terms of NEP, the LSTM performed the most consistently in this experiment due to its higher concentrations of density at 0% error as shown in Figures 6.10 and 6.11, but it does not consistently outperform the other algorithms in other metrics, like the peak error. Metrics like the peak error are also valuable to consider, as they are applicable to performance-based seismic design applications, like determining lateral drift. The LSTM's predictions mostly underestimated the magnitude of the peak response, whereas the PhyCNN and PhyLSTM occasionally overestimated the magnitude of the peak response.

In this experiment, training records with values of PGA and PFA higher than the testing records were used. Training the model in this manner showed a wider variety of the structure's behavior for the LSTM to learn from. With a different training/testing data configuration, the algorithms will likely perform differently, like if they are trying to predict large intensity motions from small intensity training data. Due to the unpredictability of nonlinearity, it is possible that the data-driven LSTM will further underestimate the response of the structure during damaging events, if it has not previously seen similar behavior in its training data. The physics-guided algorithms may not have this issue, but further investigation is needed.

Acceleration time history data are filtered and double integrated to go from the raw digitized V1 data to filtered V2 data, which could've lost physical meaning as they were filtered [11]. This loss of physical meaning can be a limitation of architectures with physics-informed loss functions [4]. In spite of this limitation, algorithms will have to leverage these data efficiently and effectively to make predictions if there is any chance of widespread deep learning applications in structural engineering. In this experiment, out of these three algorithms in their current configurations, the data-driven LSTM performed the best, having the most consistent correlation coefficients, as well as lowest peak and normalized error for most of the records. However, its predictions were not entirely consistent and it was outperformed in some instances, meaning it is not a standard or unified approach, and further investigation of other methods is needed.

7.4 Intellectual Contributions and Broader Implications

This thesis contributes an experimental setup for a validation of existing popular deep learning methods in predicting structural response as well as context about structural instrumentation, with a focus towards California and CSMIP. If deep learning is to be used widespread in industry or academia, it is important for researchers and practioners to understand the capabilities and limitations of the equipment that supply the data.

The majority of the world, 70%, lives in structures with concrete components [1]. It is difficult to model existing structures at a large scale due to the time and information required, however the increasing affordability of accelerographs and the prediction accuracy from deep learning approaches offer the potential to contribute to the safety of structures across the world. This thesis and accompanying material on GitHub[1] will enable researchers and practitioners with off-the-shelf laptops and who may not have a computer science background to be able to use these deep learning methodologies for their own experiments. This increased access to deep learning across the structural engineering domain will lead to increased experimentation and contribute greatly to future research in the field.

7.5 Future Work

Deep learning in the realm of structural engineering provides many different avenues for future research, including the investigation and comparison of new and different architectures, the usage of results metrics more oriented toward a structural health monitoring perspective, the usage of different types of response data to train the deep learning models, and the

[1]This study's GitHub repository can be accessed via: https://github.com/jacobmorgan2023

investigation of different structure types.

The comparison of additional types of deep learning architectures on the same training and testing data, like the TCN proposed by Lea et al., could provide further insight into what computational mechanisms result in the most accurate predictions of structural response [20]. As was illustrated in this study, the LSTM-based and CNN-based networks showed different trends in third floor and roof predictions.

Deep learning is increasingly being used for structural health monitoring applications. Within structural health monitoring, there are several global metrics regarding structural behavior that are investigated and could be predicted through deep learning methods. Cumulative absolute velocity (CAV), proposed by Muin and Mosalam, for example, can be predicted via data-driven models and has been previously used in the literature as a model prediction accuracy metric [11], [24].

With these three algorithms in particular, future work could entail expanded experimentation with different types of data. Truer data, like displacements obtained from GPS instruments, and displacements obtained from double integration of more precise (less-noisy) instruments, theoretically conform better to the laws of physics than relying on numerically integrated and processed acceleration records obtained from noisy field instruments. This could potentially better showcase the accuracy of physics-guided approaches. Additionally, shake table tests, like those shown in Oh et al., also provide extra levels of control to the experiment [25]. The input ground motion is known and controlled by the experimenters, specimens under investigation are typically more densely instrumented than civil structures, and structural damage is typically documented at each stage of the experiment. Knowing the behavior of the structure at each stage of a shake table experiment could inform the selection of training and testing data for a deep learning study to achieve different experimental aims based on the behavior of the structure during a particular ground motion.

Finally, comparing the prediction accuracy of deep learning models applied to different types of structures, like high-rise buildings and bridges, could be future work, since these structures react in different ways to different frequency compositions. Some neural network architectures may be better at predicting high frequency oscillation over low frequency oscillation of these structures, or vice versa.

References

[1] C. R. Gagg, "Cement and concrete as an engineering material: An historic appraisal and case study analysis," *Engineering Failure Analysis*, vol. 40, pp. 114–140, 2014. DOI: 10.1016/j.engfailanal.2014.02.004.

[2] R. Zhang, Z. Chen, S. Chen, J. Zheng, O. Büyüköztürk, and H. Sun, "Deep long short-term memory networks for nonlinear structural seismic response prediction," *Computers & Structures*, vol. 220, pp. 55–68, 2019. DOI: 10.1016/j.compstruc.2019.05.006.

[3] H. Burton and M. Mieler, "Machine learning applications in structural engineering: Hope, hype, or hindrance," *STRUCTURE Magazine*, pp. 16–20, Jun. 2021.

[4] Y.-J. Cha, R. Ali, J. Lewis, and O. Büyüköztürk, "Deep learning-based structural health monitoring," *Automation in Construction*, vol. 161, p. 105 328, 2024, ISSN: 0926-5805. DOI: 10.1016/j.autcon.2024.105328. [Online]. Available: https://doi.org/10.1016/j.autcon.2024.105328.

[5] K. Erazo and E. M. Hernandez, "High-resolution seismic monitoring of instrumented buildings using a model-based state observer," *Earthquake Engineering and Structural Dynamics*, vol. 45, pp. 2513–2531, 2016. DOI: 10.1002/eqe.2781.

[6] F. Liu, J. Li, and L. Wang, "PI-LSTM: Physics-informed long short-term memory network for structural response modeling," *Engineering Structures*, vol. 292, Oct. 2023. DOI: 10.1016/j.engstruct.2023.116500.

[7] R. Zhang, Y. Liu, and H. Sun, "Physics-informed multi-LSTM networks for metamodeling of nonlinear structures," *Computer Methods in Applied Mechanics and Engineering*, vol. 369, 113226, 2020. DOI: 10.1016/j.cma.2020.113226. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045782520304114.

[8] R. Zhang, Y. Liu, and H. Sun, "Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling," *Engineering Structures*, vol. 215, 110704, 2020. DOI: 10.1016/j.engstruct.2020.110704.

[9] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, no. 7, 2009.

[10] J. Gu, Z. Wang, J. Kuen, *et al.*, "Recent advances in convolutional neural networks," *Pattern Recognition*, vol. 77, pp. 354–377, 2018. DOI: 10.1016/j.patcog.2017.10.013.

[11] S. Günay, I. K. Pang, and K. M. Mosalam, "Structural response using deep neural networks," in *Proceedings of the SMIP23 Seminar on Utilization of Strong-Motion Data*, Oct. 19, 2023, pp. 1–22.

[12] S. Bai, J. Z. Kolter, and V. Koltun, *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling*, Apr. 19, 2018. arXiv: 1803.01271[cs].

[13] Z. Hao, S. Liu, Y. Zhang, C. Ying, Y. Feng, H. Su, and J. Zhu, *Physics-informed machine learning: A survey on problems, methods and applications*, 2023. arXiv: 2211.08064[cs,math].

[14] U.S. Geological Survey, *Nsmp history*, 2024. [Online]. Available: https://earthquake.usgs.gov/monitoring/nsmp/history/.

[15] D. M. Boore and J. J. Bommer, "Processing of strong-motion accelerograms: Needs, options and consequences," *Soil Dynamics and Earthquake Engineering*, vol. 25, no. 2, pp. 93–115, 2005. DOI: 10.1016/j.soildyn.2004.10.007.

[16] M. Huang and A. Shakal, "Strong motion instrumentation of seismically-strengthened port structures in california by csmip," in *TCLEE 2009: Technical Council on Lifeline Earthquake Engineering Conference*, American Society of Civil Engineers, Oakland, California, United States, 2009, pp. 1–9. DOI: 10.1061/41050(357)99.

[17] M. Huang and A. Shakal, "Structure instrumentation in the california strong motion instrumentation program," in *Strong Motion Instrumentation for Civil Engineering Structures*, M. Erdik, M. Celebi, V. Mihailov, and N. Apaydin, Eds. Dordrecht: Springer Netherlands, 2001, pp. 17–31. DOI: 10.1007/978-94-010-0696-5_2.

[18] H. Haddadi, A. Shakal, C. Stephens, W. Savage, M. Huang, W. Leith, R. Borcherdt, and J. Parrish, "Center for engineering strong-motion data (cesmd)," in *14th World Conference of Earthquake Engineering*, Beijing, China, 2008.

[19] A. Shakal and M. Huang, "Standard tape format for csmip strong-motion data tapes," California Department of Conservation, Division of Mines and Geology, Office of Strong-Motion Studies, Tech. Rep. OSMS 85-03, 1985. [Online]. Available: https://www.conservation.ca.gov/cgs/pages/program-smi/reports/other/osms85-03.aspx.

[20] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI: IEEE, Jul. 2017, pp. 1003–1012. DOI: 10.1109/CVPR.2017.113.

[21] F. Chollet *et al.*, *Keras*, 2015. [Online]. Available: https://github.com/fchollet/keras.

[22] *Update and New Features of the Center for Engineering Strong Motion Data (CESMD)*, SMIP09 Seminar Proceedings, California Geological Survey and U.S. Geological Survey, California, USA, Nov. 2009.

[23] M. C. Porcu, J. C. Vielma Pérez, G. Pais, D. Osorio Bravo, and J. C. Vielma Quintero, "Some issues in the seismic assessment of shear-wall buildings through code-compliant dynamic analyses," *Buildings*, vol. 12, no. 5, p. 694, 2022. DOI: 10.3390/buildings12050694.

[24] S. Muin and K. M. Mosalam, "Cumulative absolute velocity as a local damage indicator of instrumented structures," *Earthquake Spectra*, vol. 33, no. 2, pp. 641–664, May 2017. DOI: 10.1193/090416EQS142M.

[25] B. K. Oh, Y. Park, and H. S. Park, "Seismic response prediction method for building structures using convolutional neural network," *Structural Control and Health Monitoring*, vol. 27, no. e2519, 2020. DOI: 10.1002/stc.2519.