SAND2017-5165C

Machine Learning Models of Errors in Large Eddy Simulation Predictions of Surface Pressure Fluctuations

Matthew F. Barone * Jeffrey A. Fike † Kamaljit S. Chowdhary ‡

Warren L. Davis IV § Julia Ling ¶

Shawn Martin ‖

Sandia National Labs, Albuquerque, NM, 87123, and Livermore, CA, 94450 USA

We investigate a novel application of deep neural networks to modeling of errors in prediction of surface pressure fluctuations beneath a compressible, turbulent flow. In this context, the truth solution is given by Direct Numerical Simulation (DNS) data, while the predictive model is a wall-modeled Large Eddy Simulation (LES). The neural network provides a means to map relevant statistical flow-features within the LES solution to errors in prediction of wall pressure spectra. We simulate a number of flat plate turbulent boundary layers using both DNS and wall-modeled LES to build up a database with which to train the neural network. We then apply machine learning techniques to develop an optimized neural network model for the error in terms of relevant flow features.

I. Introduction

Unsteady aerodynamic pressure loads resulting from turbulent flow provide a key design environment for many aerospace structures. Examples include turbulent boundary layer pressure fluctuations on atmospheric re-entry vehicles,[1] and flow within aircraft bays.[2] High-fidelity computational fluid dynamics models, including Large Eddy Simulation (LES), offer the potential to directly predict unsteady loading environments of this kind. However, turbulence modeling, particularly in the near-wall region, inevitably results in poorly understood *model form* errors that can lead to unquantified uncertainties for the predictions. Here, a model form error is defined to be an error due to an incorrect mathematical representation of a term, or terms, in the governing equations. In the context of LES, model form error is introduced in phenomenological modeling of the sub-filter turbulent stress and in application of approximate turbulent stress boundary conditions at solid surfaces.

Many previous assessments of prediction of wall-bounded turbulent flows have focused on accuracy of predicted velocity field statistics, including mean velocity field and Reynolds stress fields. There has been a corresponding focus on model development to accurately predict these quantities of interest. There has been much less focus on accuracy of prediction of surface pressure fluctuations, with some recent exceptions.[3,4]

In this work, we bring two tools together to attempt to increase our understanding of near-wall turbulence model form error. Direct Numerical Simulations (DNS), which exactly resolve the near wall turbulent flow, will be used to investigate the sources of errors in these turbulence models. However, extracting sensitivity information from the massive amounts of data generated by DNS is an enormous challenge. To tackle this challenge, we will leverage machine learning methods.

Machine learning is a set of data-driven algorithms, including regression, classification, and clustering techniques. Recently, there has been increasing interest in applying machine learning methods to turbulence

*Principle Member of the Technical Staff, Aerosciences Department, AIAA Associate Fellow
†Post-doctoral Researcher, Aerosciences Department, AIAA Member
‡Extreme Scale Data Science and Analytics Department, AIAA Member
§Principle Member of the Technical Staff, Scalable Analysis and Visualization Department
¶Harry S. Truman Fellow, Thermal/Fluids Science and Engineering Department
‖Senior Member of the Technical Staff, Software Systems R&D Department

modeling. Tracey et al.[5] used neural networks to try to predict the source terms from the Spalart-Allmaras Reynolds Averaged Navier Stokes (RANS) model. Duraisamy et al.[6] used neural networks and Gaussian processes to predict RANS source terms as well as turbulent transition intermittency factors. Parish and Duraisamy[7] also suggested combining an inversion step with the machine learning step to determine the optimal turbulence model closures for a given RANS model. Ling et al.[8] demonstrated that random forest regressors could be used to predict the Reynolds stress anisotropy invariants more accurately than conventional RANS linear eddy viscosity models. Ling et al.[9] also showed that neural networks could embed Galilean invariance into a model for the full Reynolds stress anisotropy tensor. Sarghini et al.[10] have also investigated neural network models for LES subgrid scale closures. These studies demonstrate the significant and growing interest in applying data-driven machine learning methods to turbulence modeling applications. In the context of this paper, we will discuss how machine learning regression algorithms can be used to map between LES and DNS wall pressure spectrum predictions.

Much of our work to date has focused on the non-locality of pressure fluctuations within a compressible boundary layer, and the importance of including non-local information as inputs to a machine learning model for wall pressure fluctuations.[11] Deep neural networks have been used to analyze pressure and velocity power spectral densities (PSDs) within a turbulent boundary layer field (produced by DNS). The present study aims to apply neural networks to create models that map local flow-field information, such as nearby velocity and pressure statistics, to errors in wall pressure PSD predictions from an LES simulation. To accomplish this, we apply DNS of compressible flat plate boundary layers at several supersonic Mach numbers, along with corresponding wall-modeled LES simulations of the same flow conditions to build a database of model errors in wall pressure fluctuation PSDs.

Section II presents the computational set-up of the DNS and wall-modeled LES along with some DNS results. Section III explains the neural network implementation and toolset. Section IV describes how these tools and data sets are used to build machine-learned models of wall-pressure fluctuation errors. Section V presents some details of the DNS and WMLES database. In Section VI, we provide results from the application of neural networks to the data, and in Section VII we provide some additional discussion and conclusions.

II. Simulation Method

Both the DNS and wall-modeled LES simulations are performed using SIGMA-CFD, a multi-block, structured grid, finite volume code.[12] The code solves the compressible Navier-Stokes equations using a low-dissipation, fifth order upwind biased flux-reconstruction scheme. The time integration scheme is a fourth-order Runge-Kutta method.

For the wall-modeled LES, a sub-grid stress (SGS) model and a wall-layer model are required. For this work, we employ the standard Smagorinsky SGS model.[13] For the near-wall "model," we simply apply a no-slip boundary condition. This leads to severe under-resolution of near-wall turbulent eddies, leading to well-known solution deficiences. However, the purpose of the present study is to test the ability of machine learning algorithms to represent the error in near-wall pressure fluctuations, so the choice of near-wall treatment is not particularly important.

We simulate a spatially-developing turbulent boundary layer flowing over an adiabatic wall at several supersonic Mach numbers. Transition of the boundary layer to a turbulent state is accelerated using the inflow forcing technique of Li and Coleman.[14] Mesh sensitivity studies have been performed to ensure adequate resolution of the DNS mesh. A snapshot of the Mach 2.0 DNS solution for density gradient magnitude is shown in Figure 1. This image shows the full domain, including the transitional portion of the boundary layer and the outflow sponge region, within which the turbulent fluctuations diminish before reaching the outflow boundary of the computational domain.

The computational domain used for the DNS simulations is $165\delta_0 \times 16\delta_0 \times 6\delta_0$, where δ_0 is the initial boundary layer thickness, determined by selecting an appropriate profile from a 2D RANS simulation. The mesh resolution at the wall is such that in the domain of interest $\Delta y^+ < 0.6$, $\Delta x^+ < 5.0$, and $\Delta z^+ < 4.0$. Solution verification studies on coarser and finer meshes showed this level of resolution was adequate to ensure numerically converged results for velocity and pressure statistics. The sizes of the meshes used for each Mach number are given in Table 1. The domain is periodic in the z-direction. An absorbing sponge layer is applied from $150\delta_0 \leq x \leq 165\delta_0$ to prevent any issues with the outflow boundary condition. A

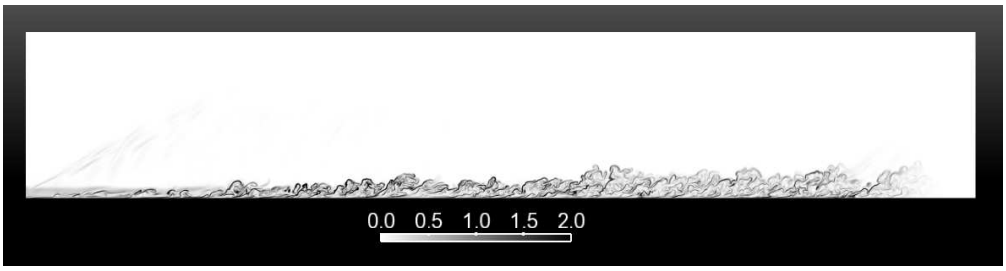

Figure 1. Schematic of DNS flow configuration showing contours of instantaneous density gradient magnitude for Mach=2.0.

Mach Number	# Cells in x	# Cells in y	# Cells in z	Total Cells
2.0	4704	193	215	195,192,480
2.5	3980	175	182	126,763,000
3.0	3435	162	157	87,365,790
3.5	3034	150	139	63,258,900

Table 1. Mesh sizes for the DNS runs at each Mach number.

Mach Number	Inflow Thickness δ_0	Van Driest Thickness
2.0	$1.8651 \cdot 10^{-4}$ m	$3.2340 \cdot 10^{-5}$ m
2.5	$1.7203 \cdot 10^{-4}$ m	$3.1339 \cdot 10^{-5}$ m
3.0	$1.6748 \cdot 10^{-4}$ m	$3.1993 \cdot 10^{-5}$ m
3.5	$1.6930 \cdot 10^{-4}$ m	$3.3761 \cdot 10^{-5}$ m

Table 2. Boundary layer thicknesses used for the inflow boudary condition forcing.

sponge layer is also applied from $6\delta_0 \leq y \leq 16\delta_0$ to prevent pressure wave reflections from the top of the domain. The inflow boundary condition uses the 2D RANS profile and applies the forcing technique of Li and Coleman. The inflow forcing applies modal perturbations which are a function of height normalized by the Van Driest thickness. For these runs, the Van Driest thickness for each Mach number was determined from a theoretical boundary layer profile. These thicknesses are given in Table 2.

The DNS simulations were run using fourth-order Runge-Kutta time integration. A constant time step of $\Delta t = 7.0 \cdot 10^{-10}$ was used for each Mach number resulting in a maximum CFL number of under 0.25. The DNS simulations were broken into three phases. In the first phase, the simulations were run for at least three flow-through times to get beyond any initial transient effects. In the second phase, each Mach number was run for between 22 and 30 flow-through times to collect statistics on the boundary layer flow. The end of the second phase was chosen to ensure that a fully-developed turbulent boundary layer was achieved. The properties of the boundary layer at the end of the second phase were used to determine the locations where time histories of the flow variables were recorded for use in computing PSDs. Specifically, this information was collected every 20 cells from $60\delta_0 \leq x \leq 145\delta_0$ and at 12 equally-spaced locations in the z-direction. For each of these locations, the values of pressure, density, and velocity were recorded at six locations through the boundary layer: at the wall, at $y^+ = 10$, at $y^+ = 25$, at $y^+ = 50$, at $y^+ = 100$, and at $y^+ = 200$. The third phase was run for 15 flow-through times to record the pressure, density, and velocity time histories at the selected locations.

The LES simulations were run using the same flow conditions and numerical scheme as for the DNS. The LES mesh was comprised of 505 cells in the x direction, 66 cells in the y direction, and 34 cells in the z direction, for a total of $1,233,220$ cells. The domain size was identical to the DNS domain, except that the streamwise length was increased to $180\delta_0$. This was because the boundary layer transition zone was longer for the LES, possibly due to the much coarser mesh resolution. The longer LES domain allowed for

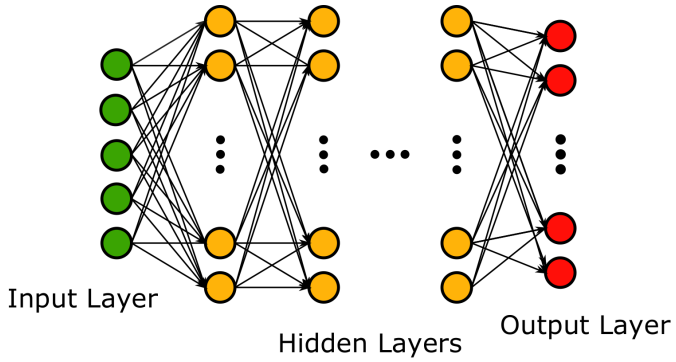

Figure 2. Schematic of multilayer perceptron neural network.

a significant portion of overlap where the LES and DNS boundary layer thicknesses could be matched.

III. Neural Network Methodology

Neural networks are a type of machine learning model in which the inputs are transformed through successive non-linear interactions. Figure 2 shows a schematic of a neural network. The input features enter the network through the input layer. The following layers are called *hidden layers*. Each node in the hidden layer is connected to every node in the following hidden layer. This type of densely connected network is named a *multilayer perceptron*. At each node in the hidden layers, the output of the node is $f(w^T x)$, where $f(x)$ is an activation function, w is a weight vector, and x is the vector of inputs to that node (i.e. the outputs from all the nodes in the previous layer.) Common activation functions include sigmoid functions, hyperbolic tangent functions, and rectified linear functions. The leaky rectified linear function (i.e. $f(x) = max(0.01x, x)$) was used in this study because of its strong performance with deep neural networks.[15] The weight vectors w for all nodes in the hidden layers are determined during the network training process. The output layer yields the final network prediction. For training, we used a learning rate of .01, and a maximum of 10,000 epochs.

There are two main phases to developing a neural network model. The first phase is the *training phase* in which the network weights are set. During this phase, the training data, a subset of the total data set, are used to calibrate the model. This is done by minimizing the mean squared error between the network predictions and the data labels. The minimization proceeded through back propagation using the Adam optimizer.[16] The second phase is the *validation phase* in which the network makes predictions on the validation subset of the data. Importantly, the validation data should be separate from the training data so that the network performance is evaluated on data it has not seen before. The network predictions on this validation data can be compared to the data labels to determine the accuracy of the network predictions.

In this paper, we describe the results of neural networks that were trained to predict either the wall pressure PSD, or the difference between the true wall pressure PSD from DNS and the LES prediction. The inputs to the neural network include the LES predictions of the discrete pressure PSD at the wall as well as other LES pressure spectra at specified distances away from the wall. The data label is the difference between the LES and DNS discrete wall pressure PSD. Therefore, for a given flow field, each point along the wall represents a distinct data point at which we would like to be able to correct the LES prediction of the pressure PSD. The networks were implemented using the Lasagne[17] python package, which is built on Theano.[18]

For our experiments, we created three different types of networks. The first is a standard multi-layer perceptron (MLP). There are 159 input nodes, corresponding to the dimensionality of the input PSD feature vectors. The first full-connected hidden layer is 795 nodes, followed by 583 nodes on the second, 371 nodes on the third, and ending with an output layer of 159 nodes, corresponding to the dimensionality of the output PSD feature vectors.

The second network was a convolutional network (CNN). CNNs were inspired by the architecture of the

cat and monkey visual cortex,[19] where the *visual field* is divided into various *receptive fields*, areas that correspond to different neural activations. Although the receptive fields focus on small portions or aspects of the visual field, collectively they span the entirety of the visual field. Similarly, in CNNs, the receptive fields are implemented as filters that apply to a smaller window of the input space, and are convolved across that space. During the learning process, useful values of the filter are learned through backpropagation as with the standard MLP layers. The output of these convolution layers are then usually sent through a pooling layer, which reduces the dimensionality of the output, producing a summary of what the filter has detected, and providing translation invariance. Max pooling, which takes the maximum output of a convolution layer within a specified pool size, is one of the more popular pooling methods. For this research, we implemented a CNN with 159 input nodes, followed by a one-dimensional convolutional layer with a window size of 3, and then a max pool layer of size 2. This was succeeded by two fully-connected output layers of 159 nodes each.

Lastly, we implemented a convolution-deconvolution network (CDNN). The architecture for this "mirrors" the convolutional network. As with the CNN, there were 159 input nodes, followed by convolutional, max pool, and fully-connected layers with the same parameters. However, these are succeeded by an unpooling layer, and deconvolutional layer, and finally an output layer of 159 nodes. This takes the form inspired by the work of Hinton et al. on autoencoders for learning compressed data representations.[20]

In all networks, we used 80% training data, and 20% testing data. Of the training data, 20% was used as validation data to avoid overfitting. We used several different methods of splitting the data along the streamwise direction. The first was using a random selection of data. However, there was a concern that partitioning the data in this way would allow spatially contiguous points to be used for training and testing. If the contiguous points were similar enough to share physical dynamics, then we could be unintentionally overfitting. So we implemented a sequential split, which took the first part of the data to use as training, and the second part for testing. We introduced a buffer of two data points between the training and testing data to help reduce overfitting between the edge points. However, there was a concern that downstream dynamics may not be captured by training only on the upstream data. So lastly, we implemented a bracketing partition, where training data came from both the upstream and downstream data, and testing data was extracted from the middle of the flow. A buffer of two data points was added both before and after the test set. In the sequential and bracketing partitioning techniques, the same partitioning was also enforced between the training and validation data.

IV. ML Analysis of Model Errors

The overall objective of this work is to create, using ML methods, a model that maps LES state variables to errors in prediction of surface pressure PSDs. Figure 3 illustrates the proposed framework for creating this model, or "map". During the training process, both DNS and wall-modeled LES simulations are used to produce surface pressure fluctuation statistics. The DNS data, D, are the truth against which the wall-modeled predictions, P, are assessed. Various candidate LES flow-field features are also saved, such as local state quantities, local state statistics, or non-local distributions of states or statistics. The candidate feature set is generated *a priori* using domain expertise and understanding from turbulent flow theory. The data D, predictions P, and feature set $\{F*\}$ are used to train a machine learning map that takes as inputs a subset of $\{F*\}$ and outputs a model error, a measure of the discrepancy between prediction and truth. Feature-selection algorithms may also be used to extract the most useful flow-field features for prediction of model error.

The present study is the first exploration of the feasibility of the proposed framework. Our features are chosen to be simply pressure PSDs at points within the boundary layer above the prediction point on the wall. Note that other relevant flow statistics can be added to the candidate feature set, such as the wall-normal Reynolds stress, a quantity which turbulence theory tells us is strongly related to boundary layer pressure fluctuations. We plan to explore other such features in future studies.

V. Flow Simulation Database

The first step in the workflow is the generation of a quality DNS database for use as input into the ML machinery. Towards this end, we performed DNS simulations of turbulent boundary layers at Mach 2.0, 2.5, 3.0 and 3.5.

In Figure 4, we compare mean and RMS velocity profiles to previously published DNS results for a Mach

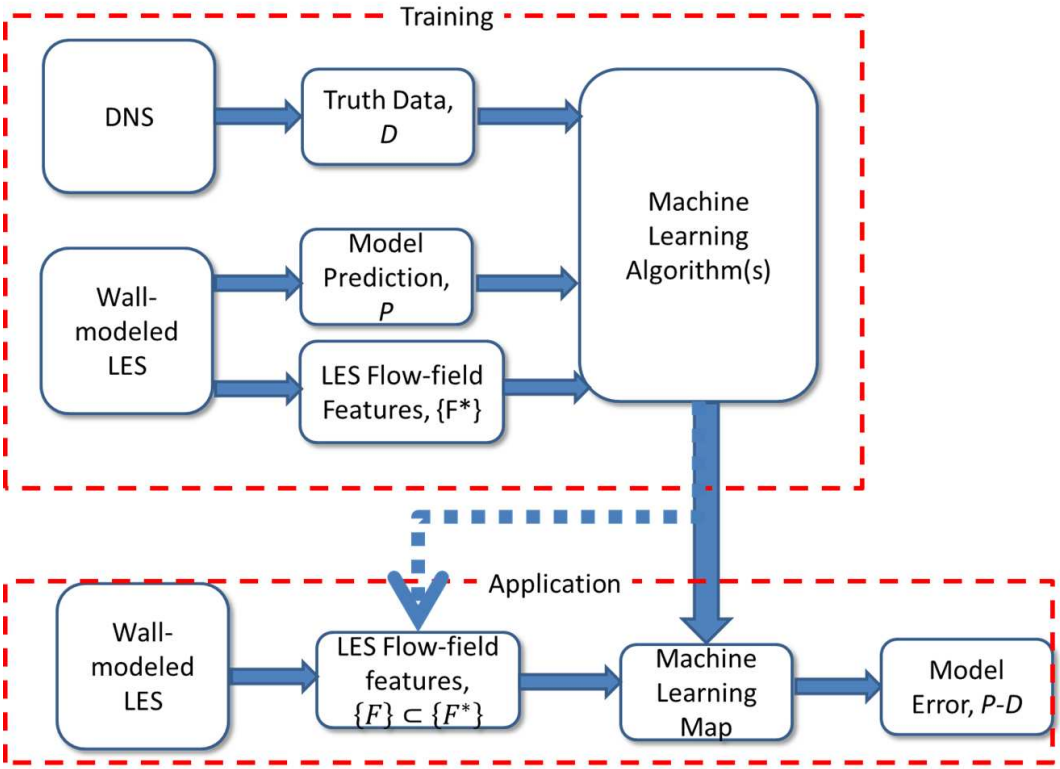

Figure 3. Method for creating a machine learning model that maps LES flow-field features to model error.

2.0 boundary layer.[21] Figure 6 shows comparisons for a Mach 3.0 boundary layer. In Figure 5, we compare the wall pressure PSD from the Mach 2.0 DNS with published results. These comparisons show excellent agreement, providing confidence in the current DNS database as a useful surrogate for providing the desired true solution.

We then ran wall modeled LES (WMLES) simulations, matching the free stream conditions from the DNS cases. Figures 7 and 8 compare mean and RMS velocity profiles from the WMLES with the DNS results shown previously at Mach 2.0 and Mach 3.0, respectively. In order to present this comparison, we chose stream-wise locations in order to match the values of Re_θ from the DNS with those from the LES. Substantial error is seen in both the mean and RMS velocity profiles; the wall shear is under-predicted, resulting in an upward shift of the non-dimensional mean velocity profile. The peak RMS velocity fluctuations are over-predicted, a typical behavior that results from poor near-wall resolution and an inadequate near-wall turbulence model.

Figure 9 shows pressure PSDs for the Mach 2.0 DNS and WMLES runs at several heights in the boundary layer. There are significant differences between the DNS and WMLES spectra. At lower frequencies, the LES over-predicts the pressure spectra, while at higher frequencies the LES spectra diminish very rapidly with frequency compared with the DNS. This behavior is fairly representative across the various Mach numbers that were simulated.

VI. Machine Learning Results

We have created and evaluated several neural networks. For each of the four Mach numbers for which we have simulation data, we evaluated three neural network architectures: MLP, CNN, and CDNN. For each of these cases we also investigated two approaches to splitting the data into training sets and testing sets: sequential and bracketed, discussed in Section III. We also consider input data from five nominal heights in the boundary layer (listed in section II). Due to the large number of permutations, we present representative comparisons.

In order to test the accuracy of our machine learning approaches, we first attempt to predict the spectra

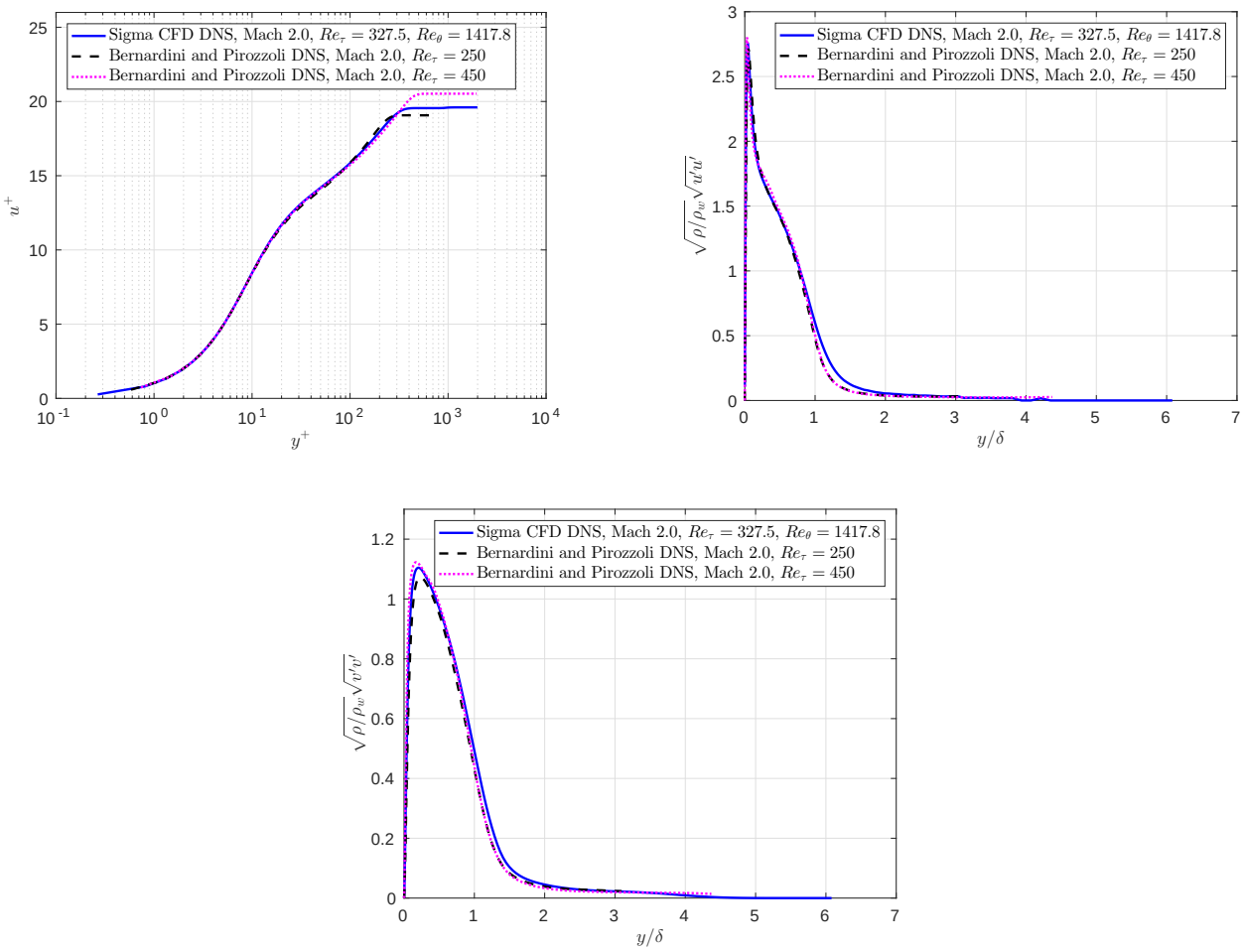

Figure 4. Comparisons of mean streamwise velocity and density-scaled RMS velocity fluctuations with previous DNS results of Bernardini and Pirozzoli.[21]

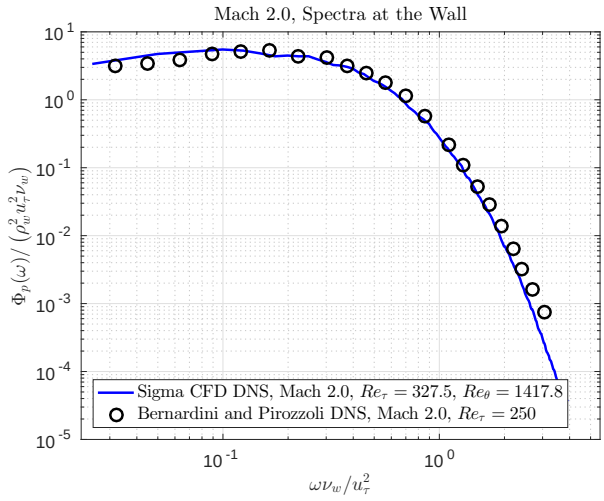

Figure 5. Comparisons of simulated wall pressure PSD with previous DNS results of Bernardini and Pirozzoli.[21]

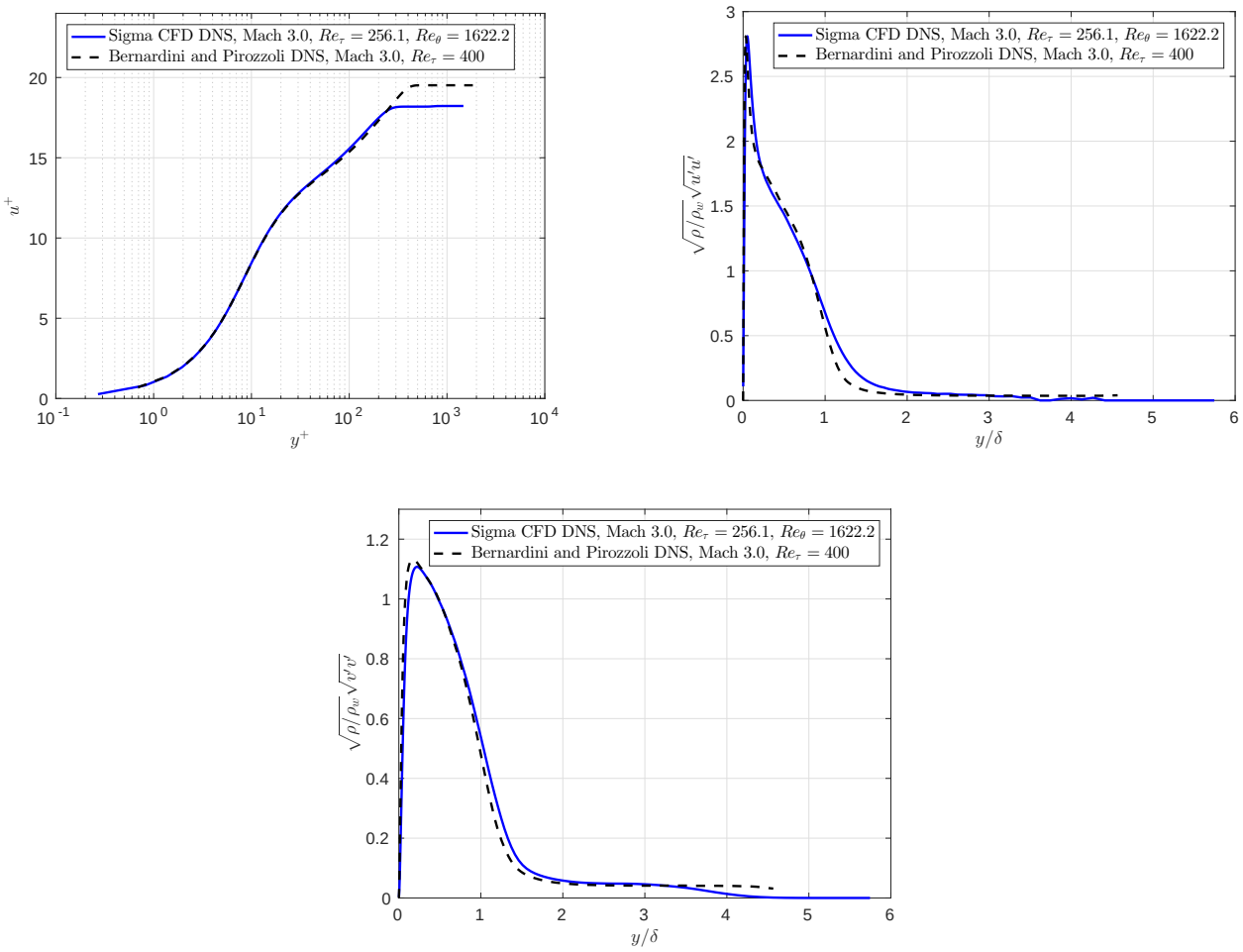

Figure 6. Comparisons of mean streamwise velocity and density-scaled RMS velocity fluctuations with previous DNS results of Bernardini and Pirozzoli.[21]

at the wall using spectra at various heights in the boundary layer using only our DNS data. Table 3 provides information on the training and testing sets used for these DNS to DNS comparisons. This table provides the range of Reynolds numbers and the number of points in the training and testing sets for each Mach number and data splitting approach.

For these DNS to DNS comparisons, we considered two sets of training data. The first set of data consisted of PSDs in units of Pa^2/Hz. The second set of data consisted of PSDs in decibel units of dB/Hz. Figure 10 shows the predicted spectra for these two sets of training data.Both plots are presented in units of dB/Hz to provide a common basis for comparison. This figure is for Mach 2.0 DNS using the MLP NN to predict the spectra at the wall using the spectra at $y^+ = 50$.

In order to provide a more quantitative comparison, we compute the ML prediction error as the difference between the predicted spectra and the true spectra. This error is computed after the predicted spectra have been converted to dB/Hz. Figure 11 shows the ML prediction error for the two sets of training data. The conclusion that can be reached from this comparision is that at high frequencies the error is lower when the training data is in units of dB/Hz. An explanation for this behavior is that when the training data is in units of Pa^2/Hz the values of the PSD at high frequencies are several orders of magnitude smaller than the values of the PSD at low frequencies. The NN training thus applies a higher importance to the lower frequency values because the high frequency values are very small in magnitude. This results in a NN that places emphasis on the accuracy of the lower frequency values. By converting the training data to dB/Hz, which involves taking a logarithm, the values of the PSD at both high and low frequencies have the same order of magnitude. The resulting NN therefore applies a more equal weight to all frequencies. This is a representative case and similar results have been observed for the other permutations of Mach number, NN

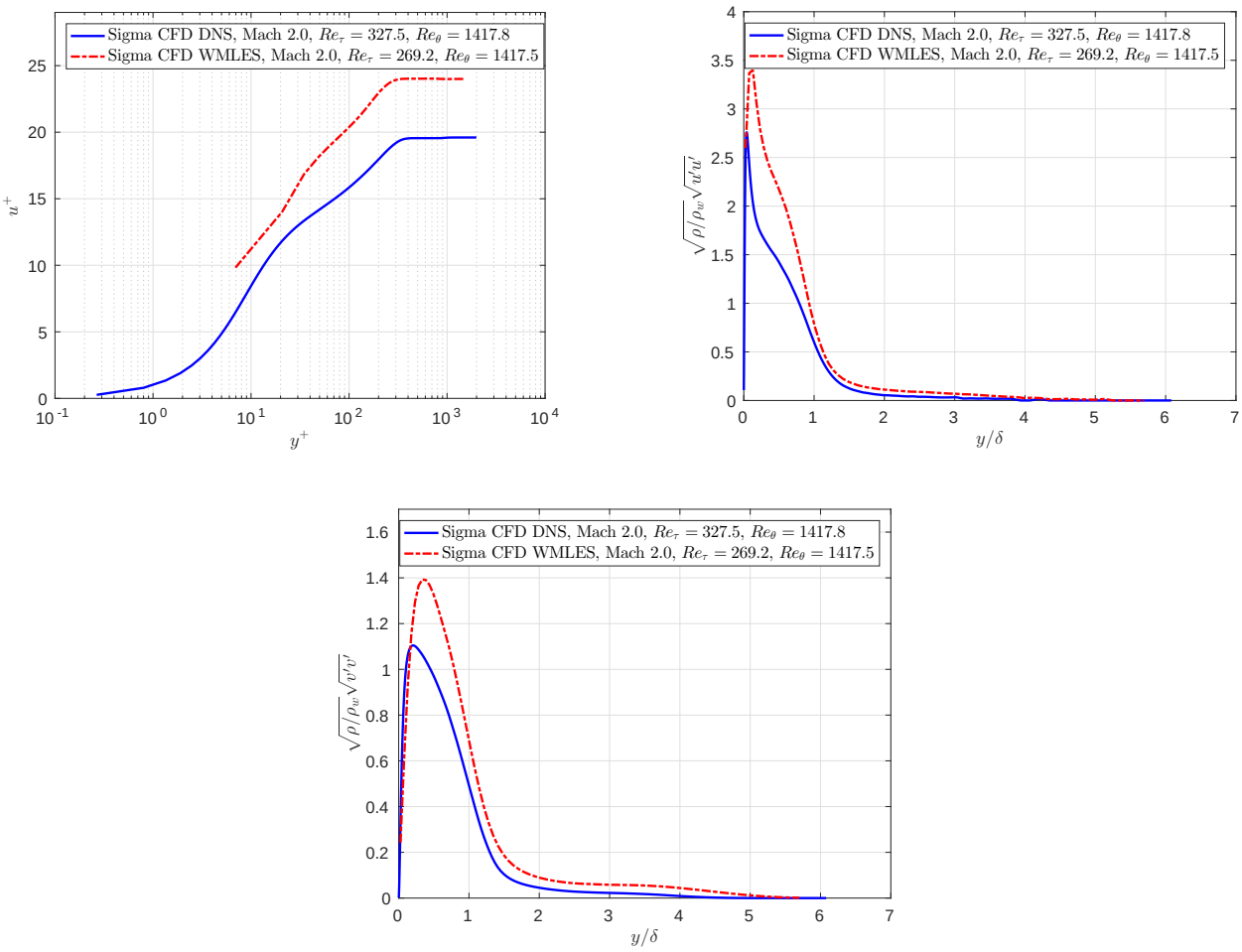

Figure 7. Comparisons of mean streamwise velocity and density-scaled RMS velocity fluctuations with previous DNS results of Bernardini and Pirozzoli.[21]

architecture, splitting method, and heights in the boundary layer.

Figure 12 shows a summary of the RMS of the ML prediction errors, summed over all frequencies and all testing points, for each of the NN architectures and training data splitting methods. For these plots, the error at each height index is computed as the average of the RMS errors at each point in the testing set. The plots show how the error changes as the height of the input PSD is varied. It might be expected that the error would increase as the height above the wall is increased. This behavior can be seen in the results using the bracketed split to the training data, but the trend is not apparent for the sequentially split data. Figure 13 shows how the RMS of the ML prediction error varies at each point in the testing set for Mach 2.0. It was anticipated that the error would be largest at the locations furthest from the training data. This would be at the middle of the bracketed set and the last point in the sequential set. However, this particular data does not show this anticipated trend. Figure 14, for Mach 3.5, on the other hand does show the anticipated trends. It has been observed that some cases exhibit the trends while others do not, making general conclusions difficult. Nonetheless, the prediction errors are relatively small in all cases, O(1dB) or less, demonstrating the robustness of the algorithm for this problem.

In some cases, it has been observed that when training with PSD data in $\mathrm{Pa}^2/\mathrm{Hz}$, the resulting predictions of the wall spectra can be negative at some frequencies. This is an unphysical result, which could be eliminated by imposing a constraint on the machine learning procedure. The negative values cause problems when we subsequently convert from $\mathrm{Pa}^2/\mathrm{Hz}$ to dB/Hz or plot on a log scale. To avoid these numerical issues, we clip the predicted PSDs to have minimum values of $1.0 \cdot 10^{-16}$. This is not an issue when training with PSD data in dB/Hz since these values are expected to be both positive and negative, providing another reason to train with dB/Hz data. For the reasons discussed above, we will limit the rest of the comparisons

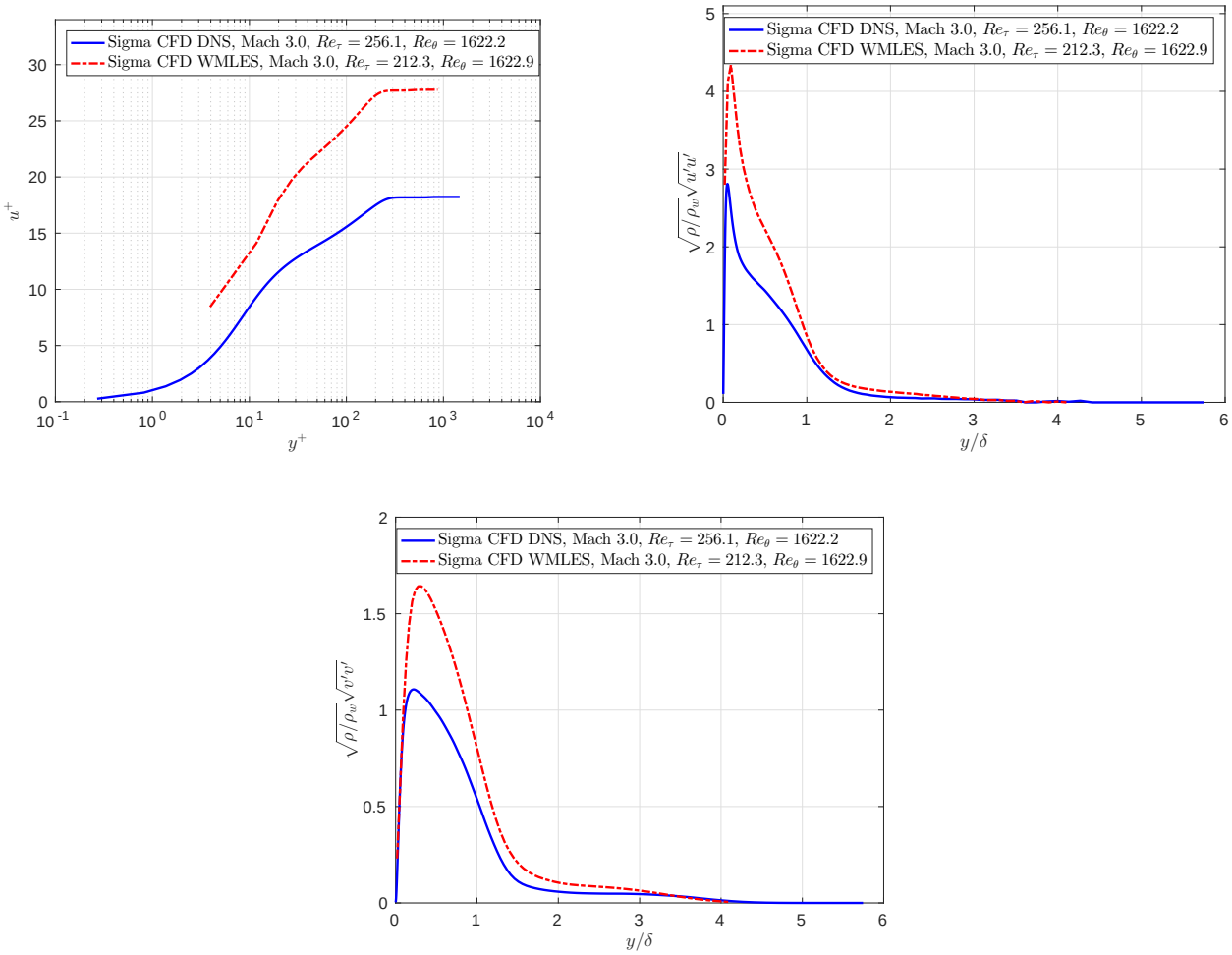

Figure 8. Comparisons of mean streamwise velocity and density-scaled RMS velocity fluctuations with previous DNS results of Bernardini and Pirozzoli.[21]

in this paper to training data in units of dB/Hz.

The above results demonstrate the accuracy of our ML methods by using DNS spectra at various heights to predict the DNS spectra at the wall. However, the main goal of this work is to be able to predict the error in WMLES generated wall pressure spectra. In other words, we are going to attempt to predict the difference in the WMLES and DNS pressure spectra at the wall using the WMLES pressure spectra at various heights in the boundary layer. To accomplish this we match locations with the same values of Re_θ between the WMLES and DNS runs so that we are taking the difference between two flow states that are similar in a well-defined sense. This matching procedure leaves us with a subset of the original data.

Table 4 provides information on the training and testing sets used for the WMLES spectra to wall spectra model error comparisons. This table provides the range of Reynolds numbers and the number of points in the training and testing sets for each Mach number and data splitting approach. The training and testing sets are greatly reduced from the previous DNS to DNS comparisons. This reduction in the amount of training data was expected to have some impact on the accuracy of our ML predictions.

Figure 15 shows the true and predicted wall spectra model error for the Mach 3.0 case for input PSDs at three different heights. Overall, the predictions show very good agreement with the true model error, especially given the reduction in training data. Figure 16 shows these same comparisons, but the plots are zoomed in to better show the differences at lower frequencies. Figure 17 shows the ML prediction error, i.e. the difference between the true and predicted wall spectra model error, at each height. There is little difference in the error between these three heights. For this particular case, i.e. Mach number, NN architecture, etc., the input PSD for lower y^+ value seems to produce greater error at high frequencies than

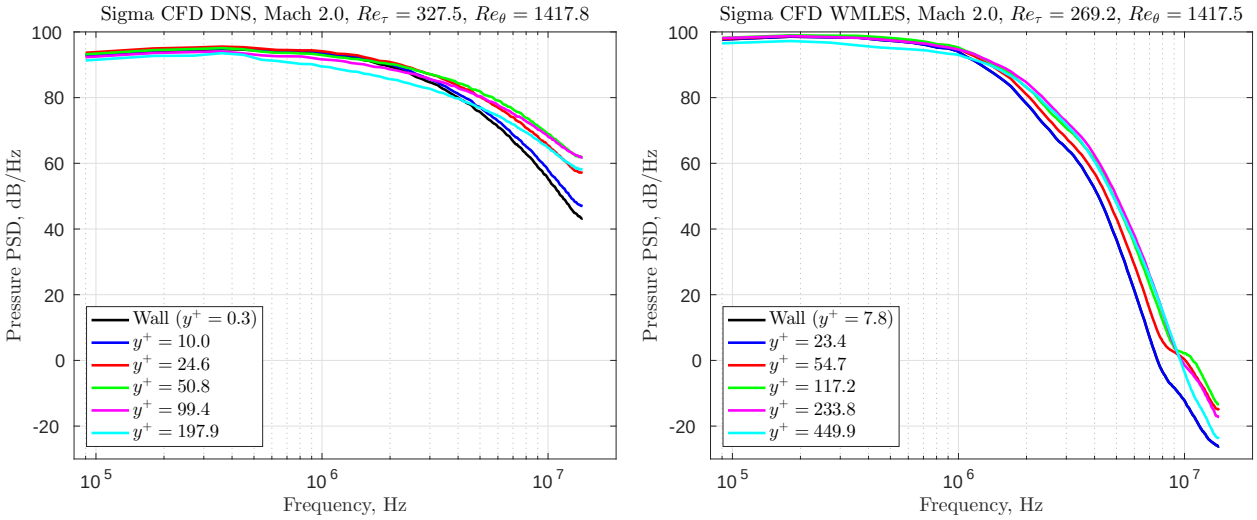

Figure 9. Comparisons of pressure spectra at several location in the boundary layer.

Mach Number	Split Method	# Training Points	# Testing Points	Re_θ, Training	Re_θ, Testing
2.0	Sequential	96	23	1286 to 1919	1938 to 2070
2.0	Bracketed	94	23	1286 to 1602, 1786 to 2070	1622 to 1766
2.5	Sequential	81	19	1331 to 1981	2004 to 2138
2.5	Bracketed	79	19	1331 to 1656, 1848 to 2138	1681 to 1825
3.0	Sequential	70	16	1379 to 2055	2082 to 2217
3.0	Bracketed	68	16	1379 to 1716, 1914 to 2217	1740 to 1886
3.5	Sequential	63	11	1384 to 2067	2098 to 2218
3.5	Bracketed	61	11	1384 to 1723, 1920 to 2218	1756 to 1888

Table 3. Training and testing sets for the DNS spectra to DNS wall spectra neural networks.

the input PSDs further from the wall. However, this is not the general trend for all of the NN predictions.

Figure 18 shows the true and predicted wall spectra model error for the Mach 3.5 case with an input PSD at $y^+ = 312$ for the three different NN architectures. Again, the predictions show very good agreement with the true model error, especially given the reduction in training data. Figure 19 shows these same comparisons, but the plots are zoomed in to better show the differences at lower frequencies. Figure 20 shows the ML prediction error for each NN architecture at this Mach number and input PSD.. All three architectures appear to be equally accurate for this case.

Figure 21 shows how well each NN architecture performs overall. There are three plots in this figure, one for each NN architecture. In these plots, each point shows the predicted model error vs. the true model error for each Mach number, input PSD, training point, and data splitting method. The dashed line shows an ideal prediction, while the dotted lines show ± 3 dB. These plots show a consistent unformity of model error prediction across the three NN arcitectures; there are no significant outlying points with large prediction error. We observe that the CDNN distribution is slightly wider than the MLP or CNN cases.

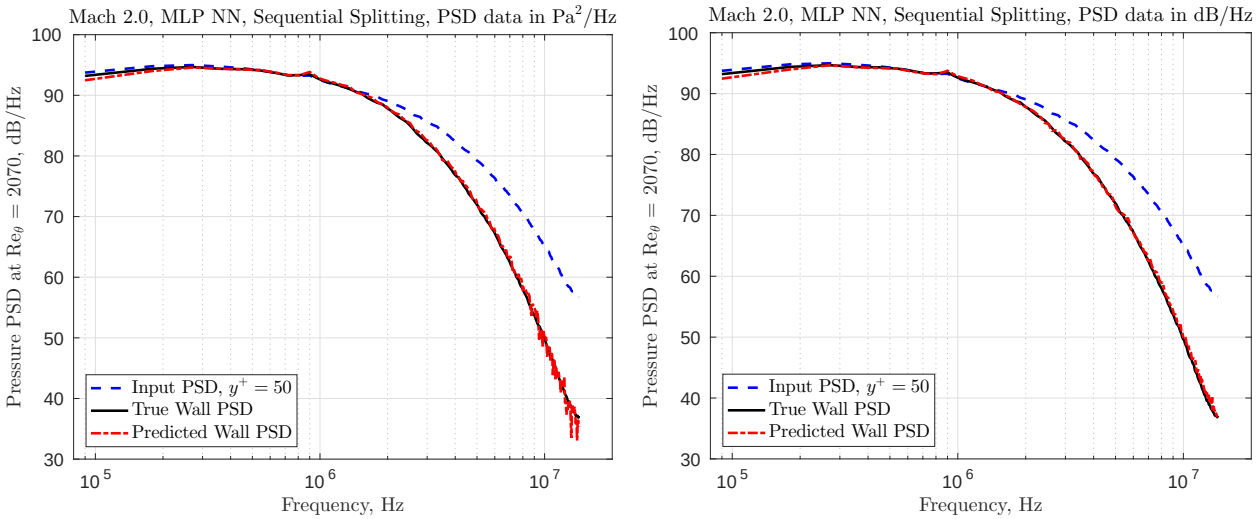

Figure 10. A comparison between predicted spectra using two sets of training data with different units. These plots are for an MLP NN with the sequential split of the Mach 2.0 training data, but similar results are observed for other permutations.

Mach Number	Split Method	# Training Points	# Testing Points	Re$_\theta$, Training	Re$_\theta$, Testing
2.0	Sequential	60	14	1286 to 1688	1708 to 1792
2.0	Bracketed	58	14	1286 to 1478, 1603 to 1792	1500 to 1590
2.5	Sequential	31	6	1331 to 1583	1607 to 1648
2.5	Bracketed	29	6	1331 to 1458, 1540 to 1648	1483 to 1516
3.0	Sequential	24	5	1379 to 1622	1652 to 1681
3.0	Bracketed	22	5	1379 to 1492, 1582 to 1681	1522 to 1552
3.5	Sequential	22	4	1384 to 1622	1657 to 1680
3.5	Bracketed	20	4	1384 to 1486, 1588 to 1680	1521 to 1554

Table 4. Training and testing sets for the WMLES spectra to wall spectra error neural networks.

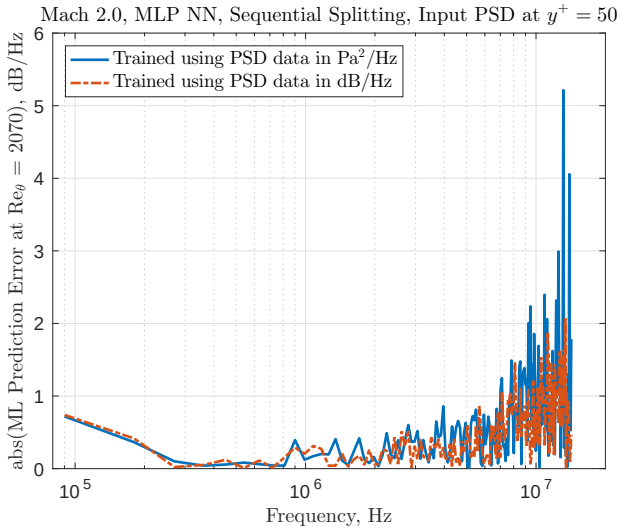

Figure 11. A comparison of the ML prediction error using two sets of training data with different units. These plots are for an MLP NN with the sequential split of the Mach 2.0 training data, but similar results are observed for other permutations.

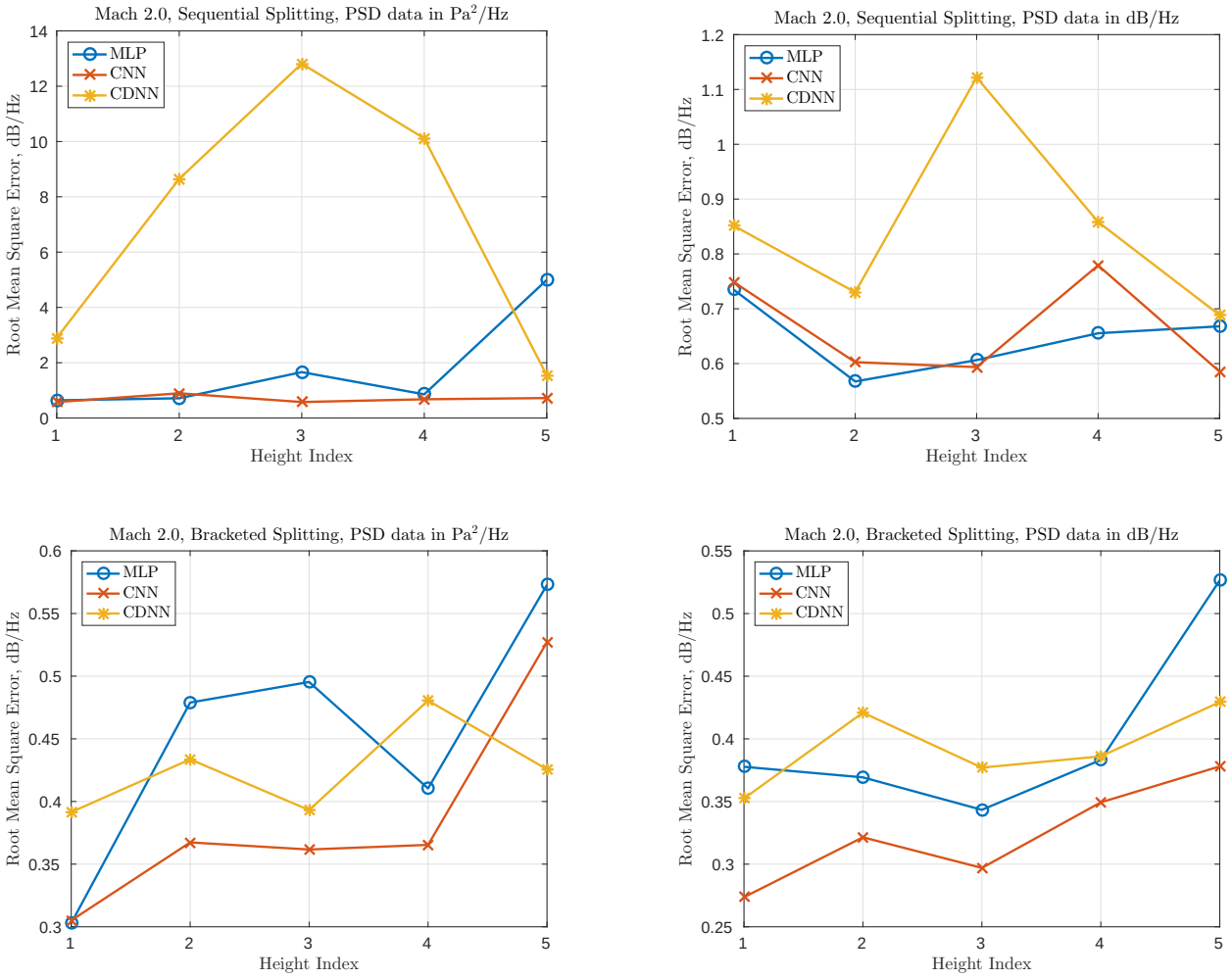

Figure 12. A summary of the RMS of the ML prediction errors for each of the NN architectures and training data splitting methods as the height of the input PSD varies.

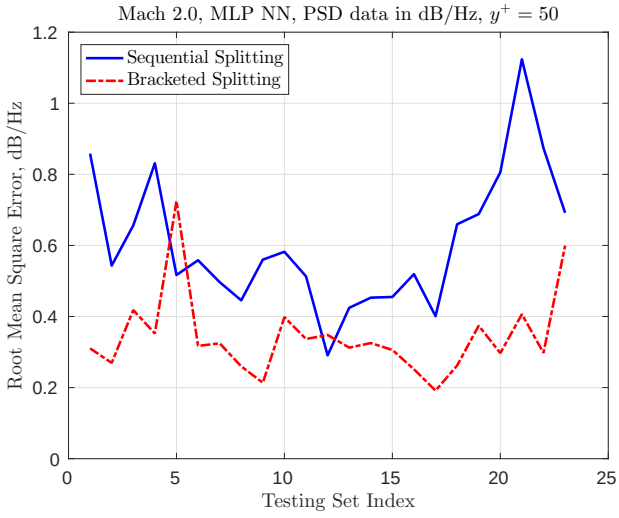

Figure 13. A comparison of the RMS of the ML prediction errors at each point in the testing set.

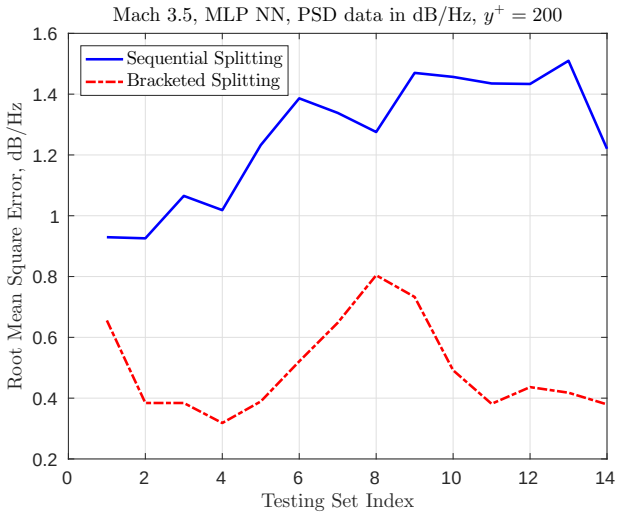

Figure 14. A comparison of the RMS of the ML prediction errors at each point in the testing set.

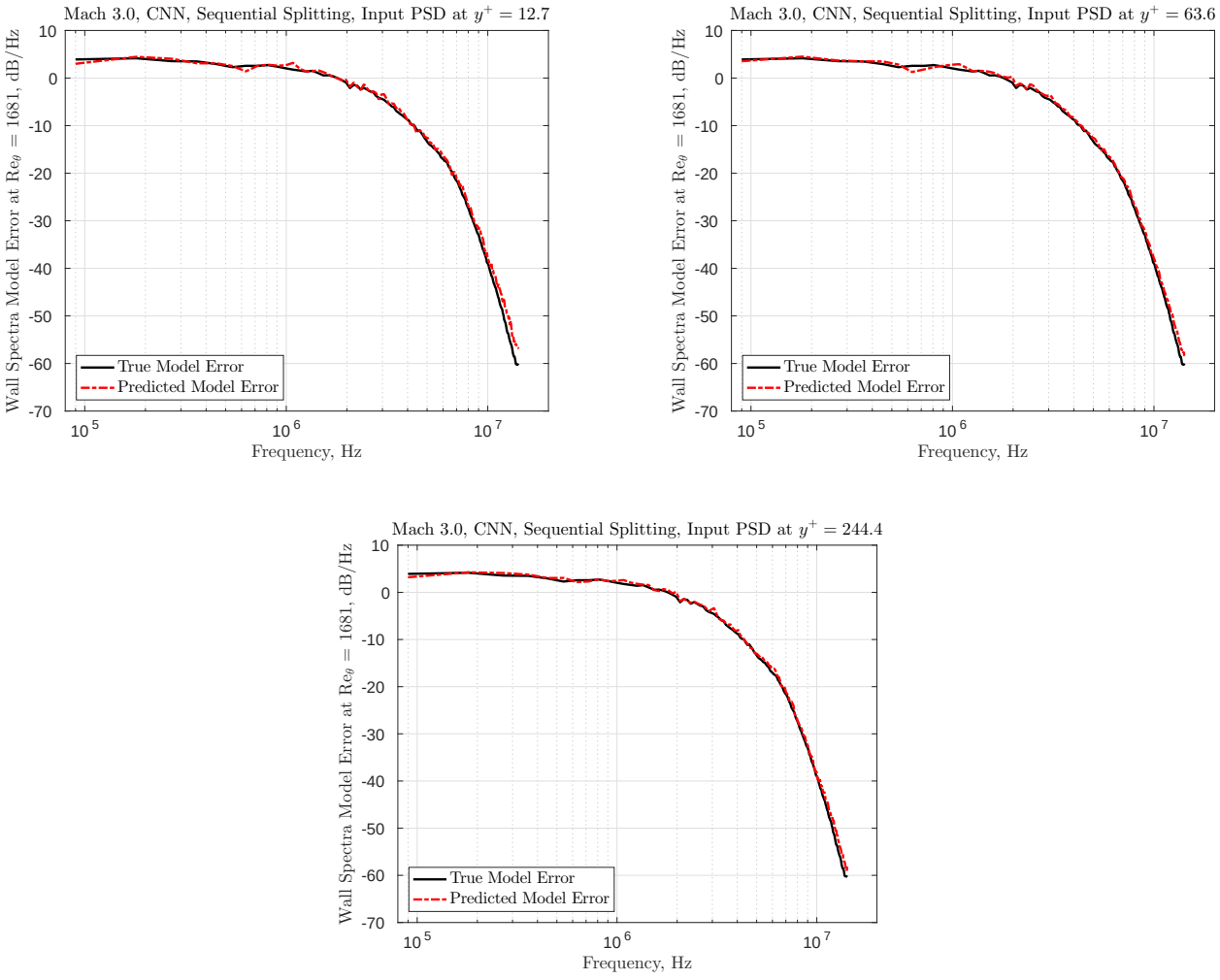

Figure 15. A comparison of the true and predicted WMLES model error for three input PSD locations.

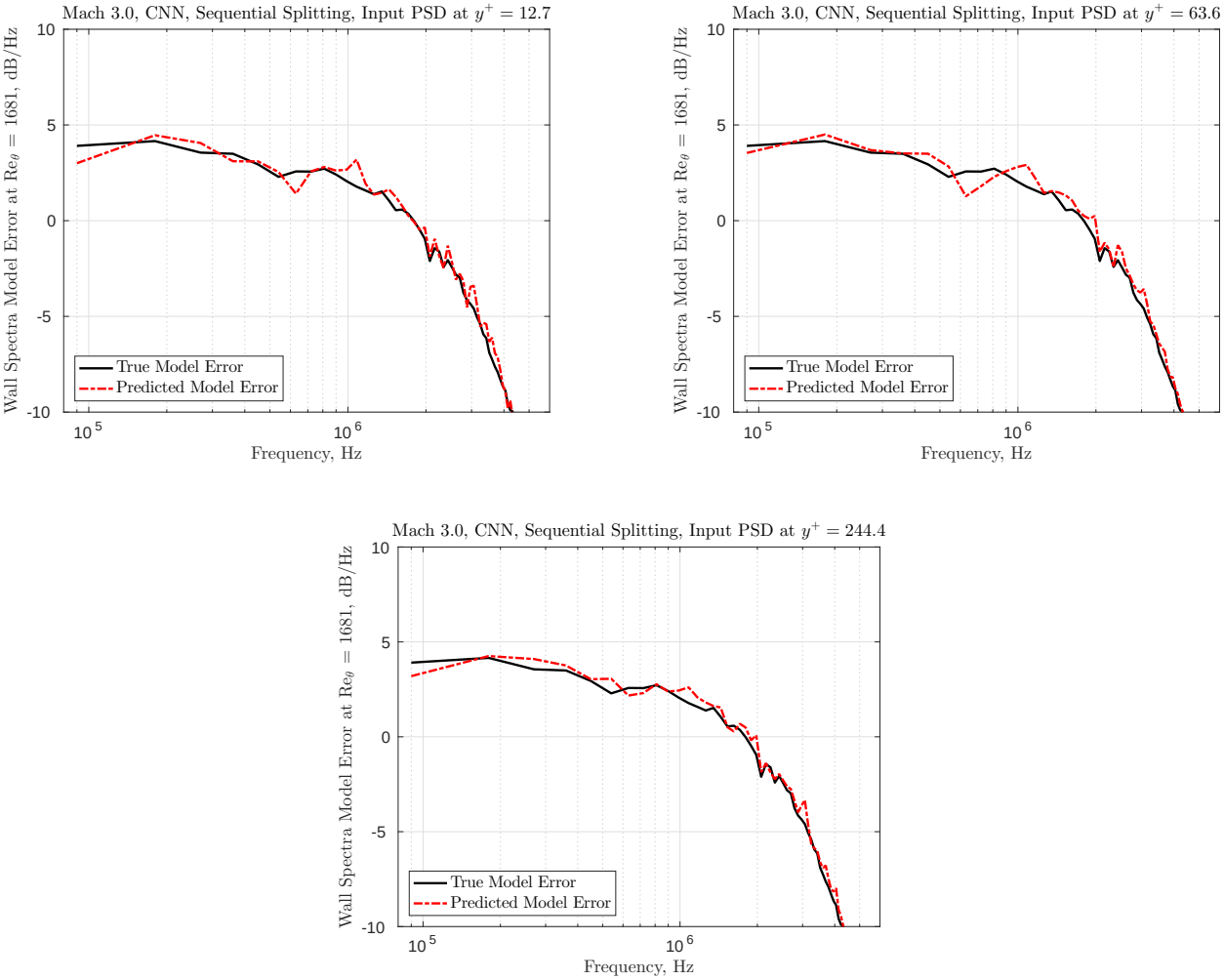

Figure 16. A comparison of the true and predicted WMLES model error for three input PSD locations, zoomed in to show lower frequency behavior.

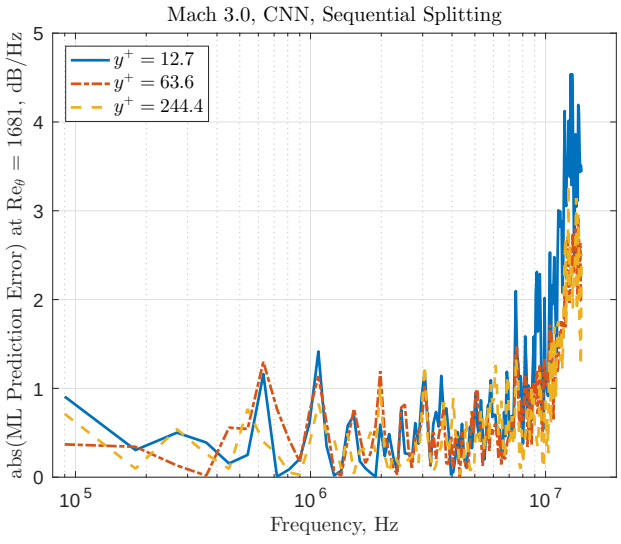

Figure 17. A comparison of the ML prediction errors for the WMLES model error as the height of the input PSD is varied.

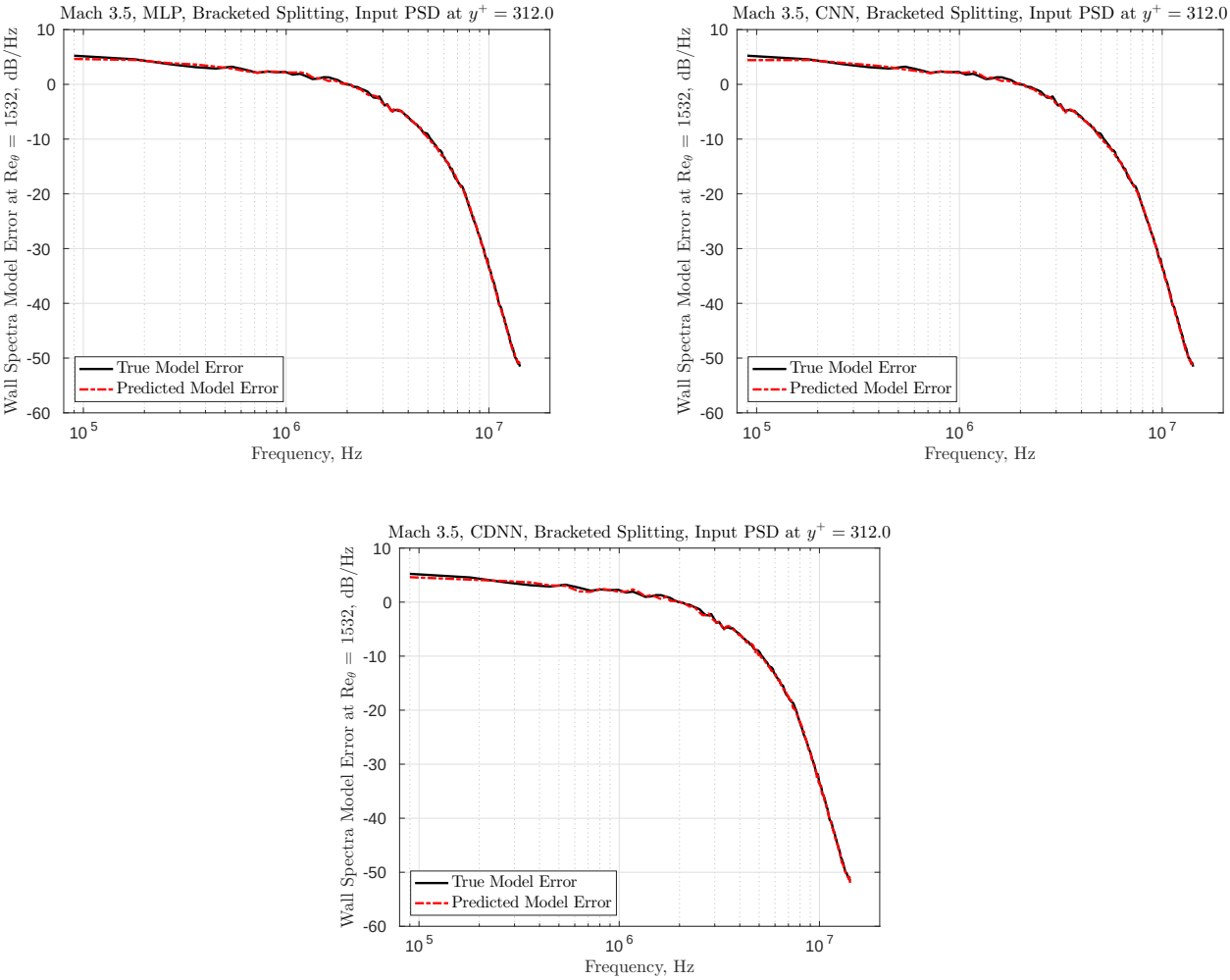

Figure 18. A comparison of the true and predicted WMLES model error for one choice of Mach number, input PSD location, and data splitting method.

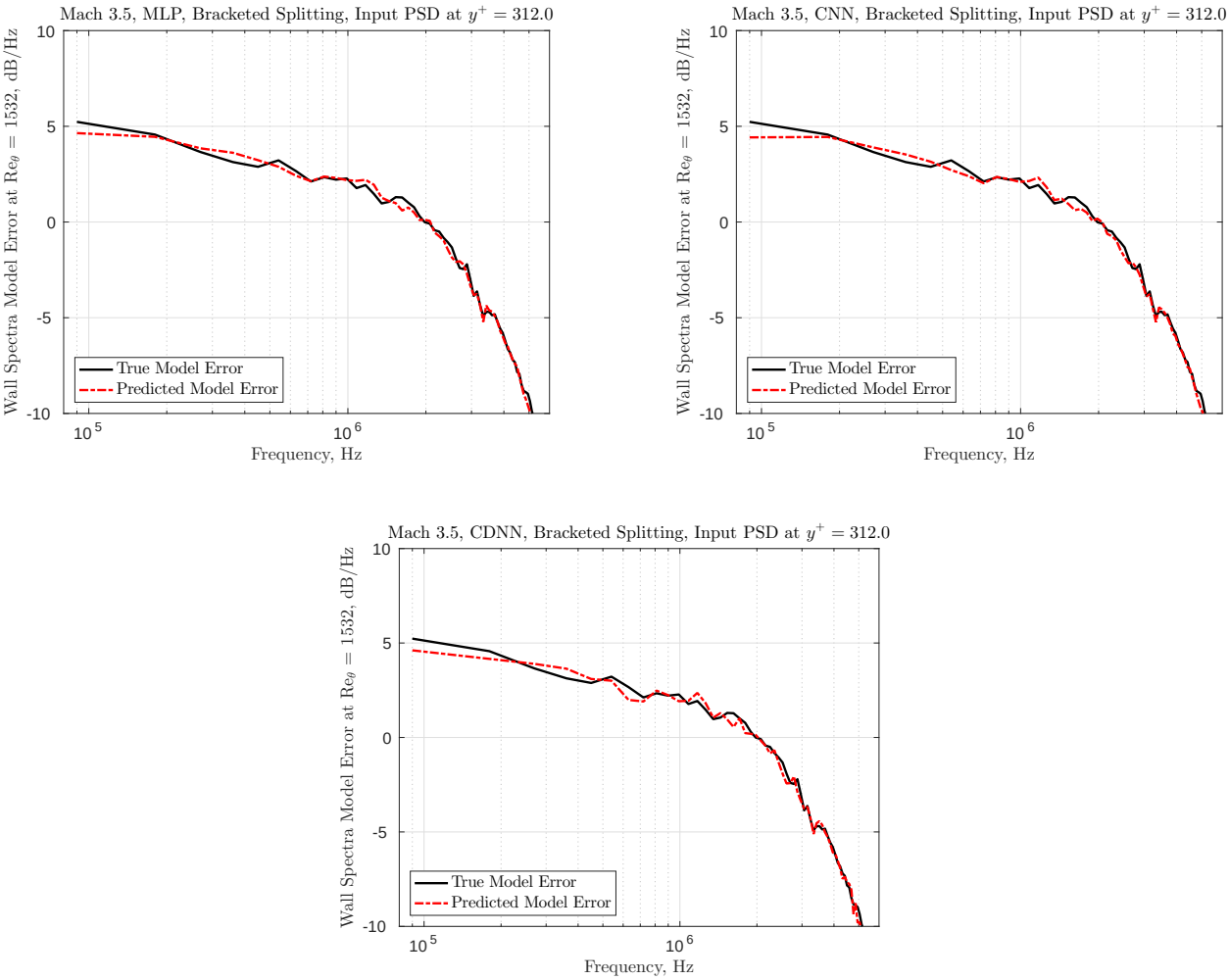

Figure 19. A comparison of the true and predicted WMLES model error for one choice of Mach number, input PSD location, and data splitting method, zoomed in to show lower frequency behavior.

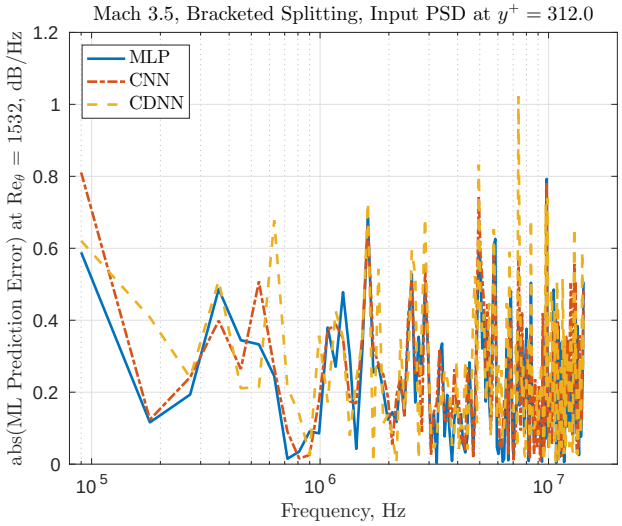

Figure 20. A comparison of the ML prediction errors for the WMLES model error as the NN architecture is varied for one choice of Mach number, input PSD location, and data splitting method.

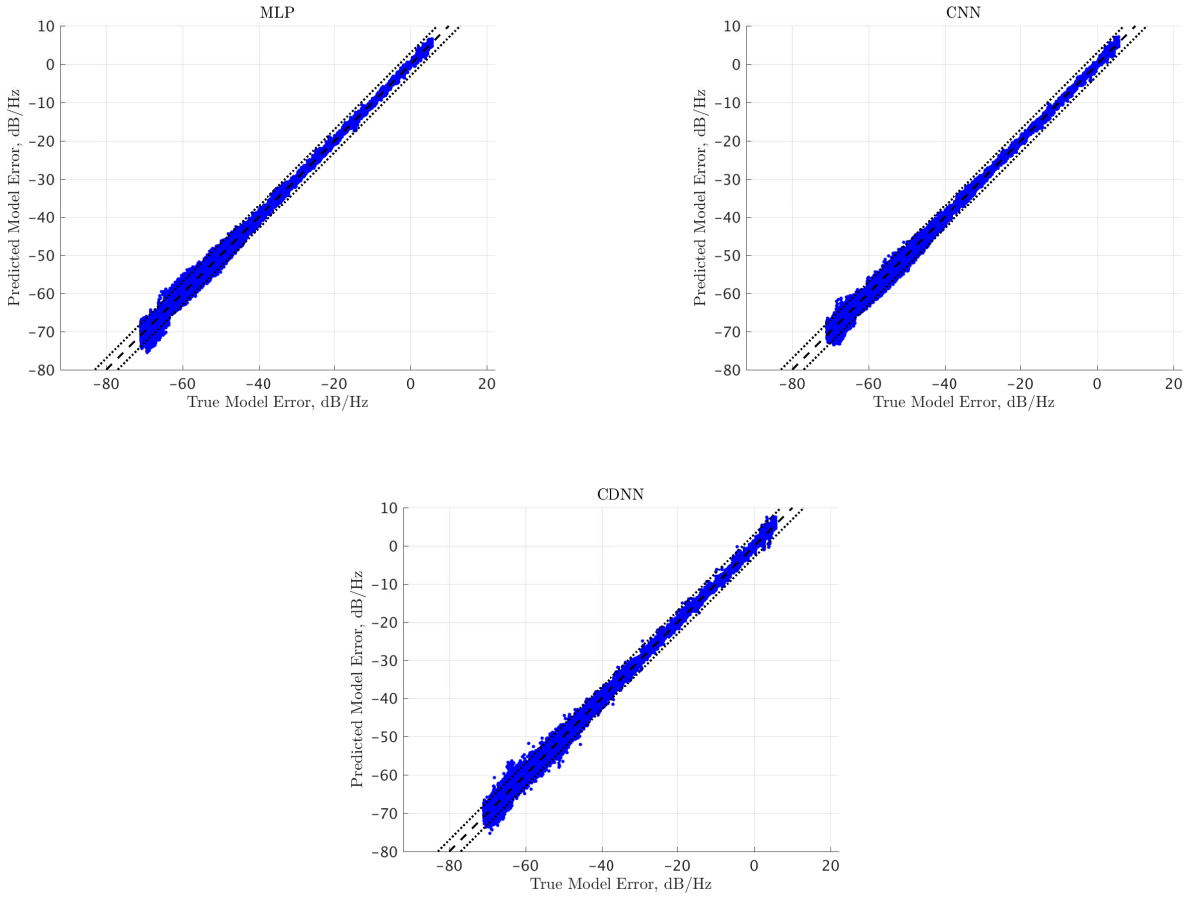

Figure 21. A comparison of the true and predicted WMLES model error for each NN architecture.

VII. Discussion and Future Work

In this study we examined the ability of various neural networks to predict errors in simulated wall pressure spectra, using nearby pressure spectra as inputs to the networks. Since our training and testing domains were distinguished by different streamwise locations in physical space, we were effectively making predictions at a different Reynolds number given training over some other range of Reynolds numbers. Admittedly, the range of Reynolds numbers considered is very modest, since the training and testing data were drawn from the same simulation of a spatially developing boundary layer. However, we were able to verify the ability of neural networks to represent these spectral relationships in a turbulent flow, and compare the performance of several different network architectures.

In future work, we will consider more difficult parameter variations between training and testing. For example, using our current flow simulation database, we can make predictions at one Mach number using a network trained on other Mach numbers. We will also broaden our set of candidate features, with an aim towards more practical application of neural networks in the turbulence modeling setting. For example, in a large-eddy simulation, a typical near-wall model requires specification of an instantaneous surface shear stress as a function of flow quantities that are local in both space and time. Neural networks or other machine learning approaches can be used to discern important local features that are useful for modeling this relationship; ultimately the machine learning map, suitably trained, could itself function as the turbulence model.

Acknowledgments

Supported by the Laboratory Directed Research and Development program at Sandia National Laboratories. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. The authors thank Sergio Pirozzoli for sharing his DNS data with us.

References

[1]K. M. Casper, S. J. Beresh, J. F. Henfling, R. W. Spillers, B. Pruett, and S. P. Schneider. Hypersonic wind-tunnel measurements of boundary-layer pressure fluctuations. AIAA 2009-4054, 39th AIAA Fluid Dynamics Conference, 2009.

[2]S. Arunajatesan and M. Barone. Towards computational study of flow within cavities with complex geometric features. AIAA 2015-0008, Proceedings of the 53rd Aerospace Sciences Meeting, 2015.

[3]G. I. Park and P. Moin. Space-time characteristics of wall-pressure and wall shear-stress fluctuations in wall-modeled large eddy simulation. *Phys. Rev. Fluids*, 1, 2016.

[4]J. Kocheemoolayil and S. K. Lele. Wall modeled Large Eddy Simulation of Trailing Edge Noise. 67th Annual Meeting of the APS Division of Fluid Dynamics, 2014.

[5]B. Tracey, K. Duraisamy, and J.J. Alonso. A machine learning strategy to assist turbulence model development. AIAA 2015-1287, Proceedings of the 53rd Aerospace Sciences Meeting, 2015.

[6]K. Duraisamy, Z.J. Shang, and A.P. Singh. New approaches in turbulence and transition modeling using data-driven techniques. *AIAA SciTech*, pages 2015–1284, 2015.

[7]E. Parish and K. Duraisamy. A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of computational physics*, 305:758–774, 2016.

[8]J. Ling, A. Ruiz, G. Lacaze, and J. Oefelein. Uncertainty analysis and data-driven model advances for a jet-in-crossflow. *Journal of Turbomachinery*, 139, 2017.

[9]J. Ling, A. Kurzawski, and J. Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

[10]F. Sarghini, G. De Felice, and S. Santini. Neural networks based subgrid scale modeling in large eddy simulations. *Computers and Fluids*, 32:97–108, 2003.

[11]J. Ling, M. Barone, W. Davis, K. Chowdhary, and J. Fike. Development of machine learning models for turbulent wall pressure fluctuations. AIAA 2017-0755, Proceedings of the 55th Aerospace Sciences Meeting, 2017.

[12]M. Barone and S. Arunajatesan. Pressure loading within rectangular cavities with and without a captive store. AIAA 2014-1406, Proceedings of the 52nd Aerospace Sciences Meeting, 2014.

[13]D. K. Lilly. The representation of small-scale turbulence in numerical simulation experiments. Proc. IBM Scientific Computing Symposium on Environmental Sciences, 1967.

[14]Q. Li and G. N. Coleman. DNS of an oblique shock wave impinging upon a turbulent boundary layer. Direct and Large-Eddy Simulation V, 2004.

[15]A. Maas, A. Hannun, and A. Ng. Rectifier nonlinearities improve neural network acoustic models. *Proceedings of ICML*, 30:1–6, 2013.

[16]D. Kingma and J.L. Ba. ADAM: A method for Stochastic Optimization. *ICLR*, pages 1–15, 2015.

[17]Sander Dieleman, Jan Schlter, Colin Raffel, Eben Olson, Sren Kaae Snderby, Daniel Nouri, et al. Lasagne: First release., August 2015.

[18]Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[19]David H. Hubel and Torsten N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology (London)*, 195:215–243, 1968.

[20]G E Hinton and R R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.

[21]M. Bernardini and S. Pirozzoli. Wall pressure fluctuations beneath supersonic turbulent boundary layers. *Phys. Fluids*, 23, 2011.