

We would like to thank both reviewers for their constructive feedback, which helped us to clarify the presented methods and improve the manuscript significantly.

Please find our response below, and all the changes in the revised manuscript (tracked changes).

Interactive comment on “Fast domain-aware neural network emulation of a planetary boundary layer parameterization in a numerical weather forecast model” by Jiali Wang et al.

Anonymous Referee #1

Received and published: 5 June 2019

This paper presents a series of neural networks designed to reproduce the output of the YSU PBL parameterization in the WRF model. The goal is to use the NN as a proxy model to reduce the computational cost of running the WRF model. The premise of the work is interesting and worthy of publication. There are several points of clarification and confusion however.

1. The topology of the models is confusing. The FFN is fine, but the size of the hidden layers should be noted. The two hierarchical models appear to be a series of nested single layer neural networks that output at each level. The goal is to enforce layer specificity, though I am not sure why this cannot also happen in the FFN as with what I assume are a larger number of internal weights and fully connected, it should be able to encode this information as well.

Response: In the revised manuscript, we detailed the figure description for Figure 1, which shows the architecture of each neural network. We specified that we have 17 hidden layers for all three neural network, indicated by the purple unit in Figure 1. We also revise the manuscript to further emphasize the difference between the three neural networks. The FFN takes all the output data (17 vertical WRF model layer \times 5 variables/layer) and input data for training. The training process doesn't know which data belongs to which layer, the output layer comprises 85 output variables. While this is a typical way to develop neural network, it doesn't consider any vertical mixing in a PBL profile. Therefore, we develop the other two neural networks, which has 17 output layers with each of them having 5 output variables for the particular PBL layer. Each of the 17 hidden layers uses the output from each of these output layers and also the near-surface input, and calculate the output for the next output layer. From the neural network perspective, the key advantage of HPC and HAC over FNN is effective back-propagation while training. In HPC and HAC, each hidden layer has an output layer; therefore, during the back propagation, the gradients from each of the output layer can be used to update the weights of the hidden layer directly to minimize the error for PBL layer specific outputs.

2. I don't understand why the FFN performance is so much worse. If the intermediate layers are sized large enough, then it should have a much larger number of connections and be able to encode more than the hierarchical models. It appears to cut out in training much earlier however. Is this just overfitting due to larger number of connections vs training data? If the amount of data was vastly increased, would we expect FFN to eventually overtake the performance of the hierarchical models?

Response: We clarified this issue and provided an explanation in the revised manuscript. As we mentioned in the response to your first comment, the FFN takes *all* the output data (17 vertical WRF model layers \times 5 variables per layer) and input data for training. The training process doesn't know which data belongs to which layer, the 85 variables are treated as a *whole thing*. The key advantage of HPC and HAC over FNN is that, during the training of HPC and HAC, because each of their hidden layer has an output layer, so during the back propagation, the gradients from each of the output layer can be used to update the weights of the hidden layer directly to minimize the error for that particular PBL layer. While for FFN, there is no output layers for each hidden layer, so there is no information that the backward propagation of FFN can take to update weights and minimize errors.

3. The writeup of the evaluation is a bit confusing. In particular L231-236. I assume it means that you trained using a single grid location, then applied the model to multiple grid points within 800km. If so this should be made more clear. Also why specify individual sites. One could calculate performance on all grid points within 800km. While doing this, it would be useful to see the drop off in performance as a function of distance. The last two plots start down this path, but with the density of points in a model, it should be straight forward to give performance as a function of distance from training point within the 800km range.

Response: Thanks for your suggestions. We have modified the text in line 231-236 to clarify that we trained our DNNs at a single location (e.g. Logan, Kansas) and then we apply the DNN to multiple grid points nearby. We also update the last two figures and the corresponding discussions. Instead of picking several stations, we test the DNN models over all the grid points of an 1100×1100 km area, with 90×91 grid points. To reduce the computing burden, we pick every other 7 grid points and get 13×13 grid points over the area, which still maintain the terrain height variability. Then we calculate RMSE and Pearson correlations of the neural network predictions compared with observations (here WRF model simulation), and see how RMSE and correlation change with distance from the center (where we develop the DNNs) of the 13×13 area. Results show that the neural network can be used by other locations if they are not far, and they have very similar terrain conditions for temperature, water vapor and wind speed. For wind direction, to use the same neural network, the nearby grid points should also under the same weather regimes, such as the large-scale circulations, etc. This indicates that, it's not always safe to train one model over a region unless the region has homogenous features in every regard we discussed here.

If we trained the network over a larger domain, we may need larger dataset for training. There is likely an optimum domain size over which a network would be useful. Therefore, we think that there may be a small number of region specific networks necessary for representing the PBL process for our entire model coverage of North American continent. The transferability tests described in the paper are one approach that we can use to determine the size of a single DNN model and its application extent. Developing DNNs for the whole region will require significantly larger computing resources. These ideas are beyond the scope of this study and will be explored in the future.

4. Can the authors comment on where they see this being put in an online model? It seems like the round trip to and from a GPU (IO) would cause a much bigger delay than just calculating the YSU parameterization in place. Offline this is not as much of a concern as all the data can be preloaded, but when there is a round trip at every model time step, it seems like the IO would become the predominant factor, and not the computation.

Response: We need GPUs for faster training. In a deployment scenario, as shown in several deep learning case studies, CPUs are enough for fast inference/prediction. In that case, we do not need GPUs and can avoid the data movement cost. Moreover, we are anticipating that the next HPC platforms that will be available will be 'accelerated' CPUs (e.g. the exascale systems at the DOE leadership computing facilities at Argonne National Laboratory and Oak Ridge Laboratory). The goal of the accelerated CPU architecture is to decrease the IO costs and make the GPU an integral part of the CPU design. The types of models we discuss here will be highly suitable for these machines.

5. I assume this is meant to be used as a proxy model for YSU when you are interested in fiddling with a different portion of the model and just want something "good enough" that is computationally cheap (For instance, you're examining microphysical parameterizations, and don't care about PBL explicitly). Is there a concern that the feedback loops with being off by as much as this NN is (up to 60% for some parameters, though much less for most) would cause the output from YSU and this model to diverge quite quickly when being run as a replacement for YSU? If so is this just meant to be a parallel option for a parameterization, or as a drop in for YSU?

Response: The expectation we have is that these types of DNN models could function as a drop-in replacement for existing parameterizations. We have trained the model with a limited amount of grid cells

as a proof of concept. Eventually, this model will be trained for all regions and conditions for the extensive simulation database we have. The goal is to develop DNN emulators for all the expensive parts of the model (radiation, microphysics, cumulus etc.) that would function as a high-spatial resolution ‘emulator’ of the model. Another possible path is to develop an emulator for the entire model disregarding each process (e.g. Scher, JGR 2019). We expect both paths to lead to the development of emulators that will be critical for generating larger ensemble of model simulations for uncertainty quantification in future climate projections.

Reference:

Scher, S.: Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning. *Geophysical Research Letters*, 45, 12,616–12,622, 2018.

Anonymous Referee #2

Received and published: 6 June 2019

This is a very interesting work. To the best of my knowledge, the authors are the first researchers who applied machine learning techniques to PBL parameterization. Their results are thought-provoking. Of course, the final evaluation of the developed NN must be performed in parallel runs of WRF with the original PBL parameterization and with the developed NN. However, it is an issue for a separate paper. I believe that this paper should be accepted after some revision and clarifications.

General comments:

1. It is not clear from the text if the authors developed a NN emulation of the PBL parameterization. NN emulation has the same inputs (sometimes augmented by additional metadata) and the same outputs as the original (in this case YSU) parameterization. Is this the case for the presented study?

Response: The DNN developed here is an emulator in the sense that it is trained using the output from YSU scheme (not MYJ or MYNN PBL scheme). The inputs for our DNNs closely correspond to variables that are used as inputs to the YSU scheme in the WRF model.

2. Domain-aware NN is a confusing term. Which domain are the authors talking about (geographic domain, domain covered by inputs in the input space, etc.)? From the sentence in the paper: “a key drawback of the naïve FFN is that it does not consider the underlying PBL domain structure, such as the patterns that are locality specific and the vertical dependence between different vertical levels of each profile”, it can be concluded that it is about vertical correlations between different vertical levels. Probably “domain-aware” name is misplaced (see also comment 5).

Response: We have used the word “domain-aware” to mean subject expertise and the word “domain” to mean the domain of science rather than a geographical or spatial domain. Thus, one of the goals of this study is to show the importance of collaborations between data scientists and domain science experts. We first develop a neural network without any additional insights from a domain expert, such as local and nonlocal mixing in the vertical direction but purely driven by a knowledge of the key inputs to the YSU scheme. We then develop neural network that incorporate domain expertise, and we consider both the local and non-local mixing by taking into account the connection between one certain layer and the previous one layer (HPC) and the previous all layers (HAC) as well as the near-surface variable as inputs. This leads to a significant improvement of the prediction accuracy. We had an explanation of the use of domain-aware in the abstract; in the revised manuscript; we made an effort to further explain it by pointing the nonlocal mixing, which are vital for the PBL process to capture the turbulence in the lower troposphere.

As we respond to Reviewer #1, the FFN takes all the output data (17 vertical WRF model layer x 5 variables/layer) and input data for training. The training process doesn't know which data (among the 85 variables) belongs to which layer, the output layer of FFN consider all 85 variables as a whole thing. While this is a typical way to develop neural network, it doesn't consider any vertical mixing in a PBL layer. Therefore, we develop the other two neural networks, which have 17 output layers with each of them having 5 variables for the particular PBL layer. Each hidden layer uses the output from each of these output layers and also the near-surface input, and calculate the output for the next output layer. From the neural network perspective, the key advantage of HPC and HAC over FNN is effective back-propagation while training. In HPC and HAC, each hidden layer has an output layer; so during the back propagation, the gradients from each of the output layer can be used to update the weights of the hidden layer directly to minimize the error for PBL layer specific outputs.

In summary, we still keep the domain-aware term, but we clarified that the domain-aware is about considering local and non-local mixing in the PBL.

3. I cannot completely agree with the aforementioned (in comment 2) sentence from the paper. First, FFN does accounts for vertical correlations in profiles because all level components of profile are built from the same neurons of the previous hidden layer. Second, in this particular study, as is explained in the text, all outputs (including all vertical components of the same profile) are normalized independently, which significantly reduces the sensitivity of FFN (and any NN) to vertical correlations between levels. To preserve vertical dependencies, a profile should be normalized as a whole but not each component independently.

Response: In the revised manuscript, we made an efforts to clarify the difference between FFN and HAC/HPC in both text and Figure 1. As we response to your comment 2, FFN takes all the output data (17 vertical WRF model layer x 5 variables/layer) and input data for training. The training process doesn't know which data belongs to which layer, the 85 output variables are treated as a whole. So it doesn't consider any vertical correlations in the profile.

We apologize for any unclear text in the original manuscript. we normalize each output variable independently, not each vertical layer independently. In other words, we normalize the whole profile of each variable separately, because the values of the five output variables are in different scale (range of values). The normalization is done per output variable so that they all have the same scale. This is a common approach in NN training as it allows the back propagation to treat the errors equally. Note that for prediction, we apply inverse transformation and compute the prediction error (R^2 and RMSE) in the original scale.

4. The time length of data set to be used for training is not a valuable and universal characteristic. It depends on representativeness of data set, i.e. on how well the variety of atmospheric states is represented in the training set, or how well the domain of input space is sampled. For example, including in the training set more grid points would enrich it with new/different atmospheric states and made more representative. It may shorten the time length of data required for training.

Response: Thanks for your comment. Reviewer #1 also had a similar insight. We do agree that the neural network developed in this study based on the individual location (Logan, Kansas) likely will not be applicable universally. However, this model will have a region of applicability that can be tested using similar approach to those discussed in the manuscript. We may need several such networks to cover the entire model simulated region or train the DNN with data from the entire model simulated region. The later option as explained will need to performed on HPC systems and will be the target of our future research.

We agree that training more grid points (in space) would enrich the dataset, but it also have the risk of introducing more noise for the neural network, unless the regions is homogenous. This might be done over a very small region by taking several grid points but not a relatively large region. By homogenous we mean that different grid points should also under the same weather regimes, such as the large scale circulations, etc. The reason we say this is that, from our spatial transferability analysis, we found the neural network can be used for temperature, water vapor, and even wind speed over other locations as far as 500km, but for wind direction, the different grid points should also under the same circulation patterns (for example, if the wind is driven by terrain over one location, then the network doesn't apply to locations that are driven by large-scale circulations). On the other hand, we do see it is worthwhile to develop DNNs for the whole region instead of individual locations. As we mentioned in our discussion, however, this will require additional computational consideration on HPC and will be considered in the follow-on effort.

5. Most/all problems with applications to neighboring grid points can be alleviated or completely removed if all grid points (entire grid) are included in training set together with some metadata, i.e. if lat, lon, and the terrain conditions (elevation etc.) are included as additional inputs at each grid point. The NN trained in such a way I'd call "domain-aware NN".

Response: Thanks for the suggestion and we found it's very helpful. In our revised manuscript, for the last two figures, instead of picking several stations we test the DNN models over all the grids of a 1100x1100 km area, with 90x91 grid points over that area. To reduce the computing burden, we pick every other 7 grid points and get 13x13 grid points over the area, which still maintain the terrain height variability. Then we calculate RMSE and correlations of the neural network prediction compared with observations (here WRF model simulation), and see how RMSE and correlation change with distance from the center (where we develop the DNNs) of the 13x13 area.

As we respond to your comment 2, we use the word 'domain' in the sense of domain-science expertise and not spatial domain. In other words, our neural networks were not developed considering spatial domain factors, they were developed only based on individual locations. It is referred to domain knowledges about the PBL structure, specifically, the local and nonlocal mixing of turbulence in the lower troposphere.

Specific comments:

1. Is the sizes of the input and output layers are $16 + 85 = 101$? (= near-surface variables) and 85 (= 17 vertical levels _ 5 output variables).

Response: For FFN, we have an input layer, which has 16 near-surface variables; we have 17 hidden layers; and one output layer, which has 85 variables (5 variables for each of the 17 WRF vertical layer).

For HPC, we have 16 near-surface variables as one part of the input, and we also use the output (5 variables) of each previous hidden layer as input for the next hidden layer. We have 17 hidden layers, and 17 output layers. HAC is similar to HPC, but uses the output of ALL the previous hidden layer as input for the next hidden layer.

We have specified this in text and also added clarifications in the caption of Figure 1.

2. It is not clear from the text how 17-level profiles produced by NN are integrated with WRF profiles in total 38 level profiles?

Response: The middle and upper troposphere (all layers above the PBL) are considered fully resolved by the dynamics simulated by the WRF model. So the upper 21 layers will be still from the WRF model itself. There may be discontinuity between the 17th and the 18th layer (which are from NN and the WRF, respectively), and need to be smoothed. This will be future study when we implement the NNs into WRF.

3. How many hidden layers have different NNs that the authors use?

Response: we used 17 hidden layers for all three NNs developed in this study. We add this information in the revised manuscript. The hidden layers are represented by the purple unit in Figure 1.

4. How many trained parameters (NN weights) has each of these NNs?

Response: for FNN we have 10,693 trained parameters; for HPC we have 16,597 trained parameters, and for HAC we have 26,197 trained parameters.

5. How many records has the training set that is used?

Response: We have described this in data at line 120-125 in the original manuscript. in the revised manuscript we move this description to **2.3 Setup** as following:

“The 22-year data from the WRF simulation was partitioned into three parts: a training set consisting of 20 years (1984–2003) of 3-hourly data to train the NN; a development set (also called validation set) consisting of 1 year (2004) of 3-hourly data used to tune the algorithm's hyperparameters and to control overfitting (the situation where the trained network predicts well on the training data but not on the test data); and a test set consisting of 1 year of records (2005) for prediction and evaluations.”

6. Did not the authors try to train a shallow NN with the same number of weights as the best DNN has? Without all this information it is difficult to understand why NNs with different architectures perform so differently.

Response: as we explained earlier, the FFN takes all the output data (17 vertical WRF model layer x 5 variables/layer) and input data for training. The training process doesn't know which data belongs to which layer, the 85 variables are treated as a whole. While this is a typical way to develop neural network, it doesn't consider any vertical mixing in a PBL layer. Therefore, we develop the other two neural networks, which has 17 output layers with each of them having 5 variables for the particular PBL layer. Each hidden layer uses the output from each of these output layers and also the near-surface input, and calculate the output for the next output layer. From the neural network perspective, the key advantage of HPC and HAC over FNN is effective back- propagation while training. In HPC and HAC, each hidden layer has an output layer; consequently, during the back propagation, the gradients from each of the output layer can be used to update the weights of the hidden layer directly to minimize the error for PBL layer specific outputs.

Fast domain-aware neural network emulation of a planetary boundary layer parameterization in a numerical weather forecast model

¹Jiali Wang, ²Prasanna Balaprakash, and ¹Rao Kotamarthi*

¹Environmental Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Lemont, IL 60439, USA

²Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Lemont, IL 60439, USA

Correspondence to: Rao Kotamarthi (vrkotamarthi@anl.gov)

Abstract. Parameterizations for physical processes in weather and climate models are computationally expensive. We use model output from a set of simulations performed using the Weather Research Forecast (WRF) model to train deep neural networks and evaluate whether trained models can provide an accurate alternative to the physics-based parameterizations. Specifically, we develop an emulator using deep neural networks for a planetary boundary layer (PBL) parameterization in the WRF model. PBL parameterizations are commonly used in atmospheric models to represent the diurnal variation of the formation and collapse of the atmospheric boundary layer—the lowest part of the atmosphere. The dynamics of the atmospheric boundary layer, mixing and turbulence, within the boundary layer, as well as velocity, temperature, and humidity profiles within the boundary layer are all critical for determining many of the physical processes in the atmosphere. PBL parameterizations are used to represent these processes that are usually unresolved in a typical numerical weather model that operates at horizontal spatial scales in the tens of kilometers. We demonstrate that a domain-aware deep neural network, which takes account of underlying domain structure that are locality specific (e.g., terrain, spatial dependence vertically nonlocal mixing between multiple vertical layers), can successfully simulate the vertical profiles within the boundary layer of velocities, temperature, and water vapor over the entire diurnal cycle. We then assess the spatial transferability of the domain-aware neural networks by using a trained model from one location to nearby locations. Results show that a single trained model from a location over the midwestern United States produces predictions of wind speed components, temperature, and water vapor profiles over the entire diurnal cycle and all nearby locations with similar terrain

[conditions](#) with [correlations higher than 0.9](#)~~errors less than a few percent~~ when compared with the WRF simulations used as the training dataset.

1 Introduction

Model developers use approximations to represent the physical processes involved in climate and weather that cannot be resolved at the spatial resolution of the model grids or in cases where the phenomena are not fully understood (Williams, 2005). These approximations are referred to as parameterizations (McFarlane, 2011). While these parameterizations are designed to be computationally efficient, calculation of a model physics package still takes a good portion of the total computational time. For example, in the community atmospheric model (CAM) developed by National Center for Atmospheric Research (NCAR), with spatial resolution of approximately 300 km and 26 vertical levels, the physical parameterizations account for about 70% of the total computational burden (Krasnopolsky and Fox-Rabinovitz, 2006). In the Weather Research Forecast (WRF) model, with spatial resolution of tens of kilometers, time spent by physics is approximately 40% of the computational burden. The input and output overhead is around 20% of the computational time at low node count (100's) and can increase significantly at higher node count as a percentage of the total wall-clock time.

An increasing need in the climate community is performing high spatial resolution simulations ([grid spacing of 104 km or less](#)~~grid spacing~~) and generating large ensembles of these simulations in order to address uncertainty in the model projections and to assess risk and vulnerability [due to climate variability at local scale](#). Developing process emulators (Leeds et al., 2013; Lee et al., 2011) that can reduce the time spent in calculating the physical processes will lead to much faster model simulations, enabling researchers to generate high spatial resolution simulations and a large number of ensemble members.

A neural network (NN) is composed of multiple layers of simple computational modules, where each module transforms its inputs to a nonlinear output. Given sufficient data, an appropriate NN can model the underlying nonlinear functional relationship between inputs and outputs with minimal human effort. During the past two decades, NN techniques have found a variety of applications in atmospheric science. For example, Collins and Tossot (2015) developed an artificial NN model by taking numerical weather prediction model (e.g., WRF) output as input to

predict thunderstorm occurrence within a few hundreds of square kilometers about 12 hours in advance. Krasnopolsky et al. (2016) used NN techniques for filling the gaps in satellite measurements of ocean color data. Scher (2018) used deep learning to emulate the complete physics and dynamics of a simple general circulation model and indicated a potential capability of weather forecasts using this NN-based emulator. Neural networks are particularly appealing for emulations of model physics parameterizations in numerical weather and climate modeling, where the goal is to find nonlinear functional relationship between inputs and outputs (Cybenko, 1989; Hornik, 1991; Chen and Chen, 1995a,b; Attali and Pagès, 1997). NN techniques can be applied to weather and climate modeling in two ways. One approach involves developing new parameterizations by using NNs. For example, Chevallier et al. (1998; 2000) developed a new NN-based longwave radiation parameterization, NeuroFlux, which has been used operationally in the European Centre for Medium-Range Weather Forecasts four-dimensional variational data assimilation system. ~~and~~ NeuroFlux is found eight times faster than the previous parameterization. Krasnopolsky et al. (2013) developed a stochastic convection parameterization based on learning from data simulated by a cloud-resolving model, (CRM), initialized with and forced by the observed meteorological data. The NN convection parameterization was tested in the NCAR CAM and produced reasonable and promising results for the tropical Pacific region. Jiang et al. (2018) developed a deep NN-based algorithm or parameterization to be used in the WRF model to provide flow-dependent typhoon-induced sea surface temperature cooling. Results based on four typhoon case studies showed that the algorithm reduced maximum wind intensity error by 60–70% compared with using the WRF model. The other approach for applying NN to weather and climate modeling is to emulate existing parameterizations in these models. For example, Krasnopolsky et al. (2005) developed an NN-based emulator for imitating an existing atmospheric longwave radiation parameterization for the NCAR CAM. They used output from the CAM simulations with the original parameterization for the NN training. They found the NN-based emulator was 50–80 times faster than the original parameterization and produced almost identical results.

We study NN models to emulate existing physical parameterizations in atmospheric models. Process emulators that can reproduce physics parameterization can ultimately lead to the development of a faster model emulator that can operate at very high spatial resolution as compared with most current model emulators that have tended to focus on simplified physics

(Kheshigi et al., 1999). Specifically, this study involves the design and development of a domain-aware NN to emulate a PBL parameterization using 22-year-long output created by a set of WRF simulations. To the best of our knowledge, we are among the first to apply deep neural networks to the WRF model to explore the emulation of physics parameterizations. As far as we know from the literature available at the time of this writing, the only application of NNs for emulating the parameterizations in the WRF model is by Krasnopolsky et al. (2017). In their study, a three-layer NN was trained to reproduce the behavior of the Thompson microphysics (Thompson 2008) scheme in the WRF-ARW model. While we focus on learning the PBL parameterization and developing domain-aware NN for emulation of PBL, the ultimate goal of our on-going project is to build an NN-based algorithm to empirically understand the process in the numerical weather/climate models that could be used to replace the physics parameterizations that were derived from observational studies. This emulated model would be computationally efficient, making the generation of large ensemble simulations feasible at very high spatial/temporal resolutions with limited computational resources. The key objectives of this study are to answer the following questions specifically for PBL parameterization emulation: (1) What and how much data do we need to train the model? (2) What type of NN should we apply for the PBL parameterization studied here? (3) Is the NN emulator accurate compared with the original physical parameterization? This paper is organized as follows. Section 2 describes the data and the neural network developed in this study. The efficacy of the neural network is investigated in Section 3. Discussion and summary follow in Section 4.

2 Data and Method

2.1 Data

The data we use in this study is [22-year](#) output from the regional climate model WRF version 3.3.1, [driven by NCEP-R2 for the period 1984-2005](#). WRF is a fully compressible, nonhydrostatic, regional numerical prediction system with proven suitability for a broad range of applications. The WRF model configuration and evaluations are given by Wang and Kotamarthi (2014). Covering all the troposphere are 38 vertical layers, between the surface to approximately 16 km (100 hPa). The lowest 17 layers cover from the surface to about 2 km above the ground. The PBL parameterization we used for this WRF simulation is known as the YSU scheme ([Yonsei University](#); Hong et al., 2006). The YSU scheme uses a nonlocal-mixing scheme with an

explicit treatment of entrainment at the top of the boundary layer and a first-order closure for the Reynolds-averaged turbulence equations of momentum of air within the PBL.

~~We use the output of the WRF model driven by NCEP-R2 for the period 1984–2005. The 22-year data was partitioned into three parts: a training set consisting of 20 years (1984–2003) of 3-hourly data to train the NN; a development set (also called validation set) consisting of 1 year (2004) of 3-hourly data used to tune the algorithm’s hyperparameters and to control overfitting (the situation where the trained network predicts well on the training data but not on the test data); and a test set consisting of 1 year of records (2005) for prediction and evaluations. The~~
goal of the work described here is to develop an NN-based parameterization [emulator](#) that can be used to replace the PBL parameterization in the WRF model. Thus, we expect the NN submodel to receive a set of inputs that are equivalent to the inputs provided to the YSU scheme at each timestep. ~~However, a key difference in our approach is that the vertical profiles of various state variables are reconstructed by the NN using only the inputs (near surface variables and 700 hPa geostrophic winds).~~

Table 1 shows the architecture in terms of inputs and outputs used in our experiments. The inputs are near-surface characteristics including 2-meter water vapor, ~~2-meter~~ [and](#) air temperature, 10-meter zonal and meridional wind, ground heat flux, incoming shortwave radiation, incoming longwave radiation, PBL height, sensible heat flux, latent heat flux, surface friction velocity, ground [temperature](#), soil temperature at 2 m below the ground, soil moisture at 0–0.3cm below the ground, and geostrophic wind component at 700 hPa. The outputs for the NN architecture are the vertical profiles of the following [five](#) model prognostic and diagnostic fields: temperature, water vapor mixing ratio, ~~and~~ zonal and meridional wind (including speed and direction), [as well as vertical motions](#). In this study we develop an NN emulation of the PBL parameterization; hence we focus only on predicting the profiles within the PBL, which is on average around 200 m and 400 m during the night and afternoon of winter, respectively, and around 400 m and 1300 m during the night and afternoon of summer, respectively, for the locations studied here. The middle and upper troposphere (all layers above the PBL) are considered fully resolved by the dynamics simulated by the model and hence not parameterized. Therefore, we do not consider the levels above PBL height because (1) they carry no information about input/output functional dependence that affects the PBL and (2) if not removed, they introduce additional noise in [the](#)

training. Specifically, we use the WRF output from the first 17 layers, which are within 1,900 meters and well cover the PBL.

2.2 Deep neural networks for PBL parameterization emulation

A class of machine learning approaches that is particularly suitable for emulation of PBL parameterization is supervised learning. This approach models the relationship between the outputs and independent input variables by using training data (x_i, y_i) , for $x_i \in T \subset D$, where T is a set of training points, D is the full data set, and x_i and $y_i = f(x_i)$ are inputs and its corresponding output y_i , respectively. The function f that maps the inputs to the outputs is typically unknown and hard to derive analytically. The goal of the supervised learning approach is to find a surrogate function h for f such that the difference between $f(x_i)$ and $h(x_i)$ is minimal for all $x_i \in T$. Many supervised learning algorithms exist in the machine learning literature. ~~In this paper,~~ This study focuses on deep neural networks (DNNs).

DNNs are composed of an input layer, a series of hidden layers, and an output layer. ~~The input layer receives the input x_i , which is connected to the hidden layers.~~ neural layers: a stack of nodes organized in a hierarchical way to model a nonlinear function. ~~Within each neural hidden layer, nodes receives~~ inputs from the previous hidden neural layer (except the first hidden layer that is connected to the input layer) ~~s~~ and perform certain nonlinear transformations through a system of weighted connections and a nonlinear activation function on the received input values. The last hidden layer is connected to the output layer from which the predicted values are obtained. The training data is given to the DNN neural network through the input neural layer. ~~The last neural layer of the stack in the network is the output neural layer from which the predicted values are obtained.~~ The training procedure consists of modifying the weights of the connections in the network to minimize a user-defined objective function that measures the prediction error of the network. Each iteration of the training procedure comprises two phases: forward pass and backward pass. In the forward pass, ~~consists of passing~~ the training data are passed to the network and ~~computing~~ the prediction error is computed; in the backward pass, the gradients of the error function with respect to all the weights in the network is computed and used to update the weights in order to minimize the error. Once the entire dataset ~~is passed~~ both forward and backward through the DNN neural network (with many iterations), one epoch is completed.

We consider three variants of DNN (see below). We construct all of them using a neural block that comprises a dense neural layer with N nodes and a rectified linear activation function, where N is user-defined parameters.

Naïve DNN:

Deep feed-forward neural network (FFN): This is a fully connected feed-forward ~~DNN~~ deep neural network constructed as a sequence of K - hidden layers~~neural blocks~~, where the input of the i th hidden layer ~~neural block~~ is from $\{i-1\}$ th hidden layer~~block~~ and the output of the i th hidden layer ~~neural block~~ is given as the input of the $\{i+1\}$ th- hidden layer~~neural block~~. The sizes of the input and output neural layers are 16 (= near-surface variables) and 85 (= 17 vertical levels \times 5 output variables). See Figure 1a for an illustration.

Domain-aware DNN:

While the FFN is a typical way of applying NN for finding the nonlinear relationship between input and output, a key drawback ~~of the naïve FFN~~ is that it does not consider the underlying PBL ~~domain~~ structure, such as ~~the patterns that are locality specific and~~ the vertical connection dependence between different vertical levels within the PBL of each profile. In fact, the FFN does not know which data (among the 85 variables) belongs to which vertical levels in a certain profile. This is not typically needed for NNs in general and in fact is usually avoided because, for classification and regression, one can find visual features regardless of their locations. For example, a picture can be classified as a certain object even that object has never appeared in the given location in the training set. In our case, however, the location is fixed and the profiles over that location is distinguishable from other locations if they have different terrain conditions. Consequently, we want it is desired to learn the particular influence of location vertical connection between multiple layers within the PBL in the forecast. For example, the feature at a lower level of a profile plays a role in the feature at a higher level and can help refine the output at the higher level and accordingly the entire profile. This dependence may inform the NN and provide better accuracy and data efficiency. To that end, we develop two variants of ~~domain-aware DNNs~~ for PBL emulation.

Hierarchically connected network with previous layer only connection (HPC): We assume that the outputs at each altitude level depend not only on the 16 near-surface variables but also

on the adjacent altitude level below it. To model this explicitly, we develop ~~a a-domain-aware~~ DNN variant ~~in which as follows:~~ the input layer is connected to the first hidden layer followed by the output layer of size 5 (five variable at each layer: temperature, water vapor, zonal and meridional wind, and vertical motions) that corresponds to the first PBL-layer. This output layer along with the input layer is connected to second hidden layer, which is connected to the second output layer of size 5 that corresponds to the second PBL-layer. Thus, the input to an i th hidden layer comprises the input layer of the 16 near-surface variables and the $i-1$ th output layer below it. ~~17 neural blocks are connected as follows: the input to an i th ($i > 1$) neural block comprises the input neural layer of the 16 near surface variables and the 5 outputs of the $\{i-1\}$ th neural block. The first neural block, which is next to the input layer, receives inputs only from the input neural layer of the 16 near surface variables.~~ See Figure 1b for an example.

Hierarchically connected network with all previous layers connection (HAC): We assume that the outputs at each PBL depend not only on the 16 near-surface variables but also on *all* altitude levels below it. ~~To model this explicitly, we modify HPC DNN as follows:~~ The input to an i th ~~hidden layer neural block~~ comprises the input ~~neural~~ layer of the 16 near-surface variables and *all* output ~~layers s of the $\{1, 2, \dots, i-1\}$ neural blocks~~ below it. See Figure 1c for an example.

~~From the physical process perspective, FNN, HPC, and HAC considers both local and non-local mixing processes within the PBL by taking into account not only the connection between a given point and its adjacent point (local mixing), but also the connections from multiple vertical altitude levels (e.g., surface and all the points that below the given points). Compared with solely local mixing process, non-local mixing process is showed to perform more accurately in simulating deeper mixing within an unstable PBL (Cohen et al. 2015). are inspired by K-Diffusion, hybrid and non-local mixing model hypothesis. [Rao, Jiali: Fill this]~~

~~From the neural network perspective, the key advantage of HPC and HAC over FNN is effective back-propagation while training. In HPC and HAC, each hidden layer has an output layer; consequently, during the back propagation, the gradients from each of the output layer can be used to update the weights of the hidden layer directly to minimize the error for PBL-layer specific outputs.~~

2.3 Setup

For preprocessing, we applied StandardScaler (removes the mean and scales each variable to unit variance) and MinMaxScaler (scales each variable between 0 and 1) transformations before training, and we applied the inverse transformation after prediction so that the evaluation metrics are computed on the original scale.

We note that there is no default value for N ~~units nodes~~ in a dense ~~hidden neural~~ layer. We conducted an experimental study on FFN and found that setting N to 16 results in good predictions. Therefore, we used the same value of $N = 16$ in HPC and HAC.

For the implementation of DNN, we used Keras (version 2.0.8), a high-level neural network Python library that runs on the top of the TensorFlow library (version 1.3.0). We used the scikit-learn library (version 0.19.0) for the preprocessing module. The experiments were run on a Python (Intel distribution, version 3.6.3) environment.

All three DNNs used the following setup for training: optimizer = adam, learning rate = 0.001, epochs = 1000, batch size = 64. Note that batch size and number of epochs define the number of randomly sampled training points required before updating the model parameters and the number of times that training will work through the entire training dataset. To avoid overfitting issues in DNNs, we use an early stopping criterion in which the training stops when the validation error does not reduce for 10 subsequent epochs.

The 22-year data from the WRF simulation was partitioned into three parts: a training set consisting of 20 years (1984–2003) of 3-hourly data to train the NN; a development set (also called validation set) consisting of 1 year (2004) of 3-hourly data used to tune the algorithm’s hyperparameters and to control overfitting (the situation where the trained network predicts well on the training data but not on the test data); and a test set consisting of 1 year of records (2005) for prediction and evaluations.

We ran training and inference on a NVIDIA DGX-1 platform: Dual 20-Core Intel Xeon E5-2698 v4 2.2 GHz processor with 8 NVIDIA P100 GPUs with 512 GB of memory. The DNN’s training and inference leveraged only a single GPU.

3 Results

In the following discussion we evaluate the efficacy of the three DNNs by comparing their prediction results with WRF model simulations. We refer to the results of WRF model simulations as observations because the DNN learns all the knowledge from the WRF model output, not from in situ measurements. We refer to the values from the DNN models as predictions. We initiate our DNN development at one grid cell from WRF output that is close to a site in the midwestern United States (Logan, Kansas, latitude= 38.8701°N; longitude= 100.9627°W) and another grid cell at a site in Alaska (Kenai Peninsula Borough, AK, latitude= 60.7237 °N; longitude=150.4484 °W) to evaluate the robustness of the developed DNNs. We then apply our DNNs to an area with size of stations within ~11800 km x 1100 km, centered at from the Logan site to assess the spatial transferability of the DNNs. In other words, we train our DNNs using a single location, and then apply the DNNs to multiple grid points nearby. While the Alaska site has different vertical profiles, especially for wind directions, and lower PBL heights in both January and July, the conclusion in terms of the model performance is similar to the site over Logan, Kansas.

3.1 DNN performance in temperature and water vapor

Figure 2 shows the diurnal variation (explicitly 3 PM and 12 AM local time at Logan, Kansas) of temperature and water vapor mixing ratio vertical profiles in the first 17 layers from the observation and three DNN model predictions. The 17 layers are within 1,900 meters and well cover the PBL. The figures present results for both January and July of 2005. The dashed lines show the lowest and highest (5th and 95th percentile, respectively) PBL heights for that particular time. In general, the DNNs are able to produce similar shapes of the observed profiles, especially within the PBL. Both the temperature and water vapor mixing ratio are lower in January and higher in July. Within the PBL, the temperature and water vapor do not change much with height; above the PBL to the entrainment zone, the temperature and water vapor start decreasing. Among the three DNNs, HAC and HPC show very low bias and high accuracy in the PBL; but the FFN shows a relatively large discrepancy from the observation. Figure 3 shows the root-mean-square error (RMSE) and Pearson correlation coefficient (COR) between observation and three DNN predictions in the afternoon and midnight of January and July. The RMSE and COR consider not only the time series of observation and prediction but also their vertical profiles below the PBL heights for each particular time. Among the three DNNs, HPC and HAC always

show better skill with smaller RMSEs and higher CORs than does FFN. The reason is that the FFN uses only the 16 near-surface variables as inputs and all the 85 variables (17 layers \times 5 variables/layer) as output, and does not have the knowledge about the vertical connections~~consider dependence~~ between each of the vertical levels. In contrast, HPC and HAC use ~~not only both~~ the near-surface variables ~~but also~~ and the five output variables of one previous vertical level (HPC) or all previous vertical levels (HAC) as inputs for predicting a certain vertical level~~the profiles~~ of each field. This architecture is helpful for reducing errors of each hidden layer during the backward propagation. This approachIt is also important because PBL parameterizations are used to represent the vertical ~~dependence of these variables~~ mixing of heat, moist, and momentum within the PBL and this mixing can be across a larger scale than just the adjacent altitude levels. and are This process is usually unresolved in a typical climate and weather models that operate at horizontal spatial scales in the tens of kilometers. ~~Compared with HAC, HPC sometimes shows slightly better accuracy with smaller RMSEs and higher CORs, but in other cases HPC performs similar to HAC. These results indicate that the information from all previous levels is not as important as information from the previous layer right below the predicted layer. We find in general HAC and HPC perform similarly, although in winter especially midnight when the PBL is shallow, the RMSE of temperature predicted by HAC is larger than that predicted by HPC. In contrast, in summer especially in afternoon when the PBL is deep, the RMSE of temperature predicted by HAC is smaller than that predicted by HPC. This emphasizes the importance of consideration of multi-level vertical connection for deep PBL case in the DNNs.~~

3.2 DNN performance in wind component

Figure 4 shows the diurnal variation of zonal and meridional wind (including wind speed and direction) profiles in January and July 2005 from observation and three DNN predictions. Compared with the temperature and water vapor profiles, the wind profiles are more difficult to predict, especially for days (e.g., summer) that have a higher PBL. The wind direction does not change much below the majority of the PBL, and it turns to westerly winds when going up and beyond the PBL. The DNN prediction has difficulty predicting the profile above the PBL height, as is expected because these layers are considered fully resolved by the dynamics simulated by the WRF model and hence not parameterized. Therefore, we do not consider DNN performance

at the levels above PBL height, because the DNNs carry no information about input/output functional dependence that affects the PBL. The wind speed increases with height in both January and July within the PBL. Above the PBL heights, the wind speed still increases in January but decreases in July. The reason is that in January the zonal wind, especially westerly wind, is dominant in the atmosphere and the wind speed increases with height; in July, however, the zonal wind is relatively weak, and the meridional wind is dominant with southerly wind below ~2 km and northerly wind above 2 km. The decrease in wind speed above the PBL is just about the transition of wind direction from southerly to northerly wind. Figure 5 shows the RMSEs and CORs between the observed and predicted wind component within the PBL. The wind component is fairly well predicted ~~especially~~ by the HAC and HPC networks with correlation above 0.8 for wind speed and 0.7 for wind direction except in July at midnight, which is ~~below near 0.5~~zero. Similar to the predictions for temperature and water vapor, the FFN shows the poorest prediction accuracy with large RMSEs and low CORs, especially for wind direction in July midnight, the COR is below zero. For accurately predicting the wind direction, we found that using the geostrophic wind at 700 hPa as one of the inputs for the DNNs is important.

3.3 DNN dependence on length of training period

Next, we evaluate how sensitive the DNN is to the amount of available training data and how much data one would need in order to train a DNN. While we present Figures 2–5 using 20-year (1984–2003) training data, here we gradually decrease the length of the training set to 12 (1992–2003), 6 (1998–2003), 2 (2002–2003) years, and 1 (2003) year. The validation data (for tuning hyper-parameters and controlling overfit) and the test data (for prediction) are kept the same as in our standard training dataset, which is year 2004 and 2005, respectively. Figures 6 and 7 show the RMSE and CORs between observed and predicted profiles of temperature, water vapor, and wind component for January ~~and July at their local~~ midnight. Overall, the FFN network depends heavily on the length of training dataset. For example, the RMSE of FFN predicted temperature decreases from 7.2 K using one year of training data to 3.0 K using 20-year training data. HAC and HPC also depend on the length of training data especially when less than 6-year training data is available, but even their worst prediction accuracy (using one year of training data) is still better than FFN using 20-year training data. The RMSEs of HPC and HAC predicted temperature decrease from ~2.4 K using 1 year of training data to ~1.5 K using 20 years of

training data. The CORs of FFN predicted temperature increase from 0.73 using one year of training data to 0.92 using 20 years of training data. The CORs for HPC and HAC increase slightly with more training data, but overall they are above 0.85 using one year to 20 years of training data.

Regarding the question about how much data one would need to train a DDN, for FFN, at least from this study, the performance is not stable until one has 12 or more years of training data, which is significantly better than having only 6 years or less of training data. For HAC and HPC, however, having 6 years of training data seems sufficient to show a stable prediction. Increasing the amount of training data shows only marginal improvement in predictive accuracy. In fact, in contrast to HAC and HPC, the performance of FFN has not reached a plateau even with the 20 years of training data. This suggests that with longer training sets the predicting skill of ~~an even naïve approach like~~ FFN could be further improved even though it does not explicitly consider the physical process within a PBL and eventually reach the accuracy of HAC and HPC using 6 or more years of training data.

3.4 DNN performance for nearby stlocations

This section assesses the spatial transferability of the domain-aware neural networks (specifically HAC and HPC) by using a trained model from one location (at Logan, Kansas, as presented above) to other locations within an area with size of 11800 km x 1100 km, covering latitude from 33.431 to 44.086°N, and longitude from 107.418 to 93.6975°W, centered at kilometers from the Logan site with different terrain ~~conditions~~ and vegetation conditions (Figure 8, top) types. To reduce the computational burden, we pick every other 7 grid points in this area and use the 13 × 13 grid points (which can still capture the terrain variability) to test the spatial transferability of the DNNs developed based on the single location at Logan, Kansas. We choose ten locations, as shown in Figure 8, among which two (Sites 1 and 2) are 300 km away from the Logan site; three (Sites 3, 4, and 5) are 430 km away from the Logan site; and five (Sites 6 to 10) are 450–800 km away from the Logan site, with Sites 9 and 10 the furthest and having the most different elevations from the Logan site. For each of the 13× 13 grid points, we calculate the differences and correlations between observations and predictions. Different from the preceding section, here we calculate normalized RMSEs relative to each grid pointsite's observations averaged over a particular time period, in order to make the comparison feasible between different grid

points over the areasites. As shown in Figures 8 and 9 and 10 by the normalized RMSEs and Pearson correlations, in general, for temperature, water vapor, and wind speed, the neural network still work fairly well for surrounding locations and even far locations with similar terrain height, except over the grid points where the terrain height is much higher than the Logan site, and the prediction skill gets worse with larger RMSEs. This suggests the DNNs developed based on Logan site are not applicable for these locations. However, for wind direction, the prediction skill is good over the western part of the tested area, but is not so good over the far eastern part of the area. One of the reasons is perhaps because that the driver of the wind direction over the western and the eastern part of the area are different (complex terrain versus large-scale system). when going farther from Logan site, where our domain-aware neural networks (HPC and HAC) were developed, the prediction skill either does not change or gets slightly worse depending on the locations and the difference in terrain conditions between the reference site (Logan, Kansas) and the remote sites (S1 to S10 in Figure 8). For example, the RMSEs for wind direction over Sites 2, 4, and 8 are similar to that over the Logan site. However, the RMSEs over the other sites, which have different elevations (either higher or lower) than that for Logan site, are much larger, suggesting the DNNs developed based on Logan site are not applicable for these locations. Overall, These results indicate that, at least for this study, as long as the terrain conditions (slope, elevation, and orientation) are similar, the DNNs developed based on one single location can be applied with similar prediction skill for locations that are as far as 520 km (equal to more than 40 grid cells in the WRF output used in this study) to predict the wind and also other variables assessed in this study. The results also suggest that when implementing the NN-based algorithm into the WRF model, if a number of grid cells are over a homogenous region, one may not need to train the NN over every grid cell. This will significantly save computing time because the training process takes the majority of the computing resource (see below). Similar to Figure 6, we see that the HPC network works better than HAC especially for temperature and water vapor over all the sites and for wind component over most of the sites examined here, indicating that the input from all previous layers is not as important as that from the input from only the layer next to the predicted layer. While we show results predicted by HAC in January here, we find similar conclusion from HPC prediction and both HAC and HPC predictions in July, expect that the prediction skills are even better in July for the adjacent locations.

3.5 DNN training and prediction time

Table 2 shows the number of epochs and time required for training FNN, HPC, and HAC for various numbers of training years. Because of the early stopping criterion, the number of training epochs performed by different methods is not same. Despite setting the maximum epochs to 1,000, all these methods terminate within 178 epochs. We observed that HPC performs more training epochs than do FNN and HAC: given the same optimizer and learning rate for all the methods, HPC has a better learning capability because it can improve the validation error more than HAC and FNN can. For a given set of training data, the difference in the training time per epoch can be attributed to the number of trainable parameters in FNN, HPC, and HAC (10,693, 16,597, and 26,197, respectively). As we increase the size of training data, the training time per epoch increases significantly for all three DNN models. The increase also depends on the number of parameters in the model. For example, increasing the training data from 1 year to 20 years increases the training time per epoch from 1.4 seconds to 11.4 seconds for FNN, from 1.1 seconds to 17.4 seconds, and from 1.4 seconds to 19.6 seconds for HPC and HAC, respectively.

The prediction times of FNN, HPC, and HAC are within 0.5 seconds for one-year data, making these models promising for PBL emulation deployment. The difference in the prediction time between models can be attributed to the number of parameters in the DNNs: the larger the number of parameters, the [higher-longer](#) the prediction time. For example, the prediction times for FNN are below 0.2 seconds when using different numbers of years for training, while those for HAC are around 0.4 seconds. Despite the difference in the number of training years, the number of parameters for a given model is fixed. Therefore, once the model is trained, the DNN prediction time depends only on the model and the number of points in the test data (1 year in this study). Theoretically, for the given model and the test data, the prediction time should be constant even with different amounts of training dataset. However, we observed slight variations in the prediction times that range from 0.17 to 0.29 seconds for FNN, 0.30 to 0.34 seconds for HPC, and 0.36 to 0.42 seconds for HAC, which can be attributed to the system software.

4 Summary and Discussion

This study developed DNNs for emulating [the YSU](#) PBL parameterization that is used by the WRF model. Two of the DDNs take into account the domain-specific features [\(e.g., nonlocal](#)

446 ~~mixing in terms of such as spatial~~vertical dependence ~~between multiple PBL layers in the vertical~~
447 ~~direction over the location where we develop the DNNs~~. The input and output data for the DNNs
448 are taken from a set of 22-year-long WRF simulations. We developed the DNNs based on a
449 midwestern location in the United States. We found that the domain-aware DNNs can reproduce
450 the vertical profiles of wind, temperature, and water vapor mixing ratio with high accuracy yet
451 require fewer data than the traditional DNN, which does not care about the domain-specific
452 features. The training process takes the majority of the computing time. Once trained, the model
453 can quickly predict the variables with decent accuracy. This ability makes the deep neural
454 network appealing for parameterization emulator.

455 Following the same architecture that we ~~applied~~developed for Logan, Kansas, we also built DNNs
456 for one location at Alaska. The results share the same conclusion as we have seen for the Logan
457 site. For example, among the three DNNs, HPC and HAC show much better skill with smaller
458 RMSEs and higher correlations than does FFN. The wind profiles are more difficult to predict
459 than the profiles of temperature and water vapor. For FFN, the prediction accuracy increases
460 with more training data; for HPC and HAC, the prediction skill stays similar when having six or
461 more years of training data.

462 While we trained our DNNs over individual locations in this study using only one computing
463 node (with multiple processors), there are 300,000 grid cells over our WRF model domain,
464 which simulated the North American continent as a horizontal resolution of 12 km. To train a
465 model for all the grid cells or all the homogeneous regions over this large domain, we will need
466 to scale up the algorithm to hundreds if not thousands of computing nodes to accelerate the
467 training time and the make the entire NN-based simulation faster than the original
468 parameterization.

469 The ultimate goal of this project is to build an NN-based algorithm to empirically understand the
470 process in the numerical weather and climate models and to replace the PBL parameterization
471 and other time-consuming parameterizations that were derived from observational studies. ~~This~~
472 ~~emulated model thus would be computationally efficient and enable researchers to generate large~~
473 ~~ensemble simulations at very high spatial/temporal resolutions with limited computational~~
474 ~~resources~~. The DNNs developed in this study can provide numerically efficient solutions to a

wide range of problems in environmental numerical models where lengthy, complicated calculations describing physical processes must be repeated frequently or need a large ensemble of simulations to represent uncertainty. A possible future direction for this research is implementing these NN-based schemes in WRF for a new generation of hybrid regional-scale weather/climate models that fully represent the physics at a very high spatial resolution at a fast computational time so as to provide the means for generating large ensemble model runs.

Data and code availability. The data used and the code developed in this study are available at <https://github.com/pbalapra/dl-pbl>.

Author contributions. JW participated in the entire project by providing domain expertise and analyzing the results from the NN-based emulator. PB developed the deep neural networks and ~~did all~~[conducted](#) the experiments ~~presented in this study~~. RK proposed the idea of this project and provided high-level guidance and insight for the entire study.

Competing interests. The authors declare that they have no conflict of interest.

Acknowledgments. The WRF model output was developed through computational support by the Argonne National Laboratory Computing Resource Center and Argonne Leadership Computing Facility. This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357.

References

Attali, J. G., and Pagès, G.: Approximations of functions by a multilayer perception: A new approach, *Neural Networks*, 6, 1069–1081, 1997.

Chevallier, F., Chérut, F., Scott, N. A., and Chédin, A.: A neural network approach for a fast and accurate computation of longwave radiative budget, *J. Appl. Meteorol.*, 37, 1385–1397, 1998.

Chen, T., and Chen, H.: Approximation capability to functions of several variables, nonlinear functionals and operators by radial basis function neural networks, *Neural Networks*, 6, 904–910, 1995a.

Chen, T., and Chen, H.: Universal approximation to nonlinear operators by neural networks with arbitrary activation function and its application to dynamical systems, Neural Networks, 6, 911–917, 1995b.

Chevallier, F., Morcrette, J.-J., Chérut, F., and Scott, N. A.: Use of a neural-network-based longwave radiative transfer scheme in the EMCWF atmospheric model, Q. J. R. Meteorol. Soc., 126, 761–776, 2000.

Collins, W., and Tissot, P.: An artificial neural network model to predict thunderstorms within 400 km² South Texas domains, Meteorol. Appl., 22 (3), 650-665, 2015.

[Cohen, A.E., Cavallo, S.M., Coniglio, M.C., Brooks, H.E.: A review of planetary boundary layer parameterization schemes and their sensitivity in simulating a southeast U.S. cold season severe weather environment, Weather Forecast., 30, 591-612, 2015.](#)

Cybenko, G.: Approximation by superposition of sigmoidal functions, Math. Control Signals Syst., 2, 303–314, 1989.

Hauser, A., and Bühlmann, P.: Characterization and greedy learning of interventional Markov equivalence classes of directed acyclic graphs. J. Mach. Learn. Res., 13, 2409–2464, 2002.

Hong, S.-Y., Noh, S.Y., and Dudhia, J.: A new vertical diffusion package with an explicit treatment of entrainment processes. Mon. Wea. Rev., 134, 2318–2341, 2006.

Hornik, K.: Approximation capabilities of multilayer feedforward network, Neural Networks, 4, 251–257, 1991.

Jiang, G.-Q., Xu, J., and Wei, J.: A deep learning algorithm of neural network for the parameterization of typhoon-ocean feedback in typhoon forecast models, Geophysical Research Letters, 45, 3706–3716, 2018.

Kheshgi, H. S., Jain, A. K., Kotamarthi, V. R., and Wuebbles, D. J.: Future atmospheric methane concentrations in the context of the stabilization of greenhouse gas concentrations. *Journal of Geophysical Research: Atmospheres*, 104, D16: 19183–19190, 1999.

Krasnopolsky, V. M., Fox-Rabinovitz, M. S., and Chalikov, D. V.: New approach to calculation of atmospheric model physics: Accurate and fast neural network emulation of long wave radiation in a climate model, *Mon. Weather Rev.*, 133, 1370–1383, 2005.

Krasnopolsky, V. M., and Fox-Rabinovitz, M. S.: Complex hybrid models combining deterministic and machine learning components for numerical climate modeling and weather prediction, *Neural Networks*, 19, 122–134, 2006.

Krasnopolsky, V. M., Fox-Rabinovitz, M.S., and Belochitski, A. A.: Using ensemble of neural networks to learn stochastic convection parameterizations for climate and numerical weather prediction models from data simulated by a cloud resolving model. *Adv. Artif. Neural. Syst.*, 1–13, 2013.

Krasnopolsky, V. M., S. Nadiga, A. Mehra, E. Bayler, and D. Behringer: Neural networks technique for filling gaps in satellite measurements: Application to ocean color observations, *Computational Intelligence and Neuroscience*, 2016, Article ID 6156513, 9 pages, 2016. doi:10.1155/2016/6156513

Krasnopolsky, V. M., J. Middlecoff, J. Beck, I. Geresdi, and Z. Toth. A neural network emulator for microphysics schemes. 97th AMS annual meeting, Seattle, WA. January 24, 2017.

Lee, L. A., Carslaw, K. S., Pringle, K. J., Mann, G. M., and Spracklen, D. V.: Emulation of a complex global aerosol model to quantify sensitivity to uncertain parameters, *Atmos. Chem. Phys.*, 11, 12,253–12,273, 2011.

563 Leeds, W. B., Wike, C. K., Fiechter, J., Brown, J., and Milliff, R. F.: Modeling 3D spatio-
564 temporal biogeochemical processes with a forest of 1D statistical emulators. *Environmetrics*,
565 24(1): 1–12, 2013.

566

567 McFarlane, N.: Parameterizations: representing key processes in climate models without
568 resolving them. *Wiley Interdisciplinary Reviews: Climate Change*, 2 (4): 482–497, 2011.

569

570 Scher, S.: Toward data-driven weather and climate forecasting: Approximating a simple general
571 circulation model with deep learning. *Geophysical Research Letters*, 45, 12,616–12,622, 2018.

572

573 Thompson, G., Field, P.R., Rasmussen, R.M., Hall, W.D.: Explicit forecasts of winter
574 precipitation using an improved bulk microphysics scheme. Part II: Implementation of a new
575 snow parameterization, *Mon. Weather Rev.* 136, 5095–5115, 2008.

576

577 Wang, J., and Kotamarthi, V. R.: Downscaling with a nested regional climate model in near-
578 surface fields over the contiguous United States, *Journal of Geophysical Research, Atmosphere*,
579 119, 8778–8797, 2014.

580

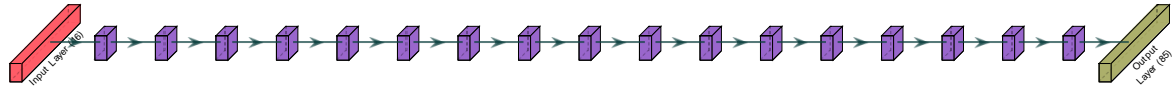
581 Williams, P. D.: Modelling climate change: the role of unresolved processes. *Philosophical*
582 *Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 363
583 (1837): 2931–2946, 2005.

584 **Table 1: Inputs and outputs for the NN developed in this study. The variable names of**
585 **these inputs and outputs in the WRF are shown in the parentheses.**

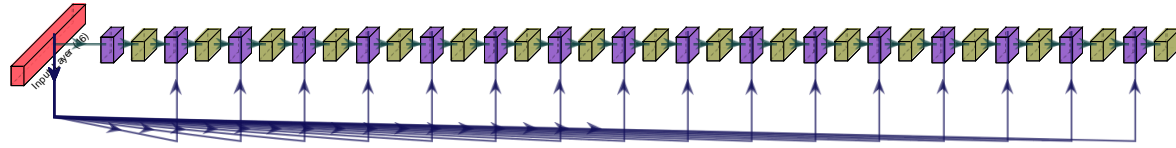
Input Variable	Output Variable
2-meter water vapor mixing ratio (Q2),	zonal wind (U)
2-meter air temperature (T2),	meridional wind (V)
10-meter zonal and meridional wind (U10, V10)	temperature (tk)
Ground heat flux (GRDFLX)	water vapor mixing ratio (QVAPOR)
Downward short wave flux (SWDOWN)	
Downward long wave flux (GLW)	
Latent heat flux (LH)	
Upward heat flux (HFX)	
Planetary boundary layer height (PBLH)	
Surface friction velocity (UST)	
Ground temp (TSK)	
Soil temperature at 2 m below ground (TSLB)	
Soil moisture for 0-0.3cm below ground (SMOIS)	
Geostrophic wind component at 700 hPa (Ug, Vg)	

587 **Table 2: Training and prediction time (unit: seconds) for the three DNNs using different**
588 **lengths of training data. The predicted period is for one year (2005).**

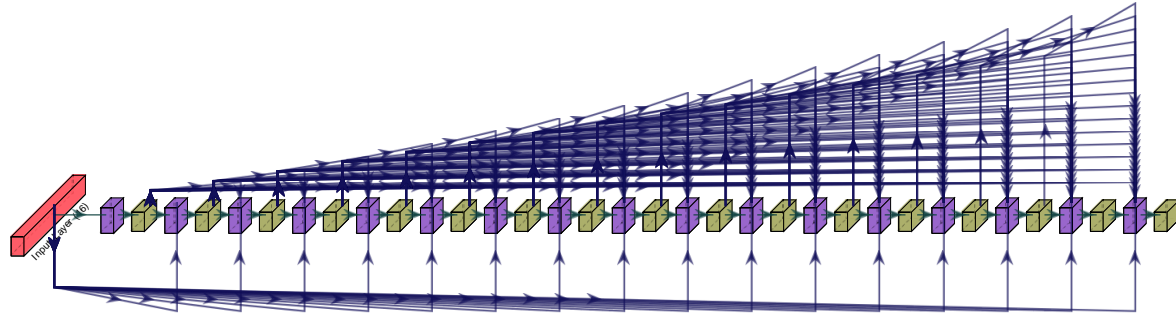
DNN Type	Training Data (years)	Training Time (s)	Number of Epochs	Training Time (s) per Epoch	Prediction Time (s) for 1 Year (2005)
FNN	1	85.969	61	1.409	0.197
FNN	2	137.359	47	2.923	0.196
FNN	6	376.209	70	5.374	0.171
FNN	12	199.468	23	8.673	0.193
FNN	20	306.665	27	11.358	0.199
HPC	1	199.152	178	1.119	0.336
HPC	2	454.225	91	4.991	0.343
HPC	6	1233.908	133	9.278	0.317
HPC	12	1225.880	88	13.930	0.302
HPC	20	1181.716	68	17.378	0.331
HAC	1	131.104	95	1.380	0.366
HAC	2	468.884	85	5.516	0.411
HAC	6	870.753	80	10.884	0.406
HAC	12	737.921	47	15.700	0.420
HAC	20	1351.898	69	19.593	0.381



(a)



(b)



(c)

Figure 1: Three variants of DNN developed in this study. Red, yellow, and purple indicate the input layer (16 near-surface variables), output layers, and hidden layers, respectively. (a) fully connected feed forward neural network (FFN), which has only one output layer with 85 variables (5 variables for each of the 17 WRF model vertical levels), and 17 hidden layers which only consider the near-surface variables as inputs. (b) hierarchically connected network with previous layer only connection (HPC), which has 17 output layers (corresponding to the PBL levels) with each of them having 5 variables, and 17 hidden layers with each them considering both near-surface variables and output variables from previous output layer as inputs. and (c) hierarchically connected network with all previous layers connection (HAC), same as HPC, but each hidden layer also considers output variables from all previous output layers as inputs.

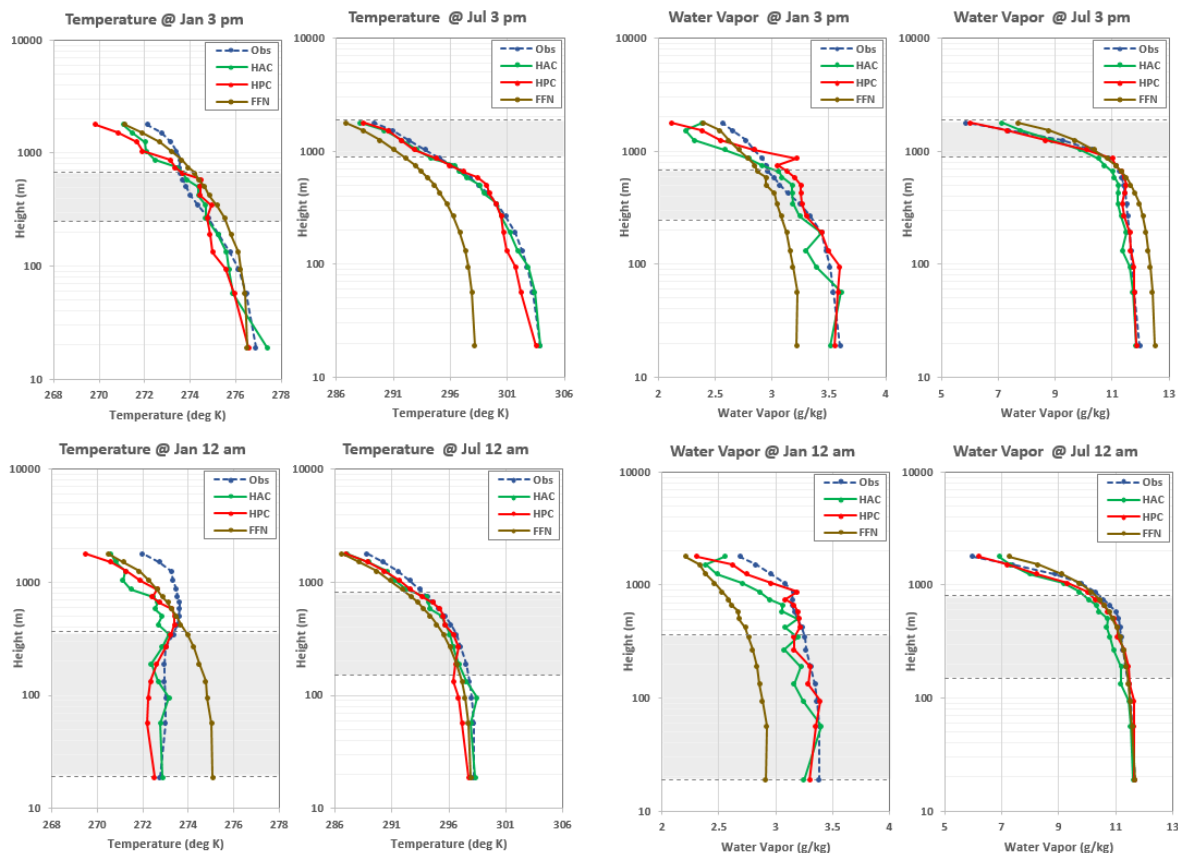


Figure 2: Temperature and water vapor mixing ratio from the observation and three DNN predictions: FFN, HPC, and HAC in January and July of 2005 at 3 PM and 12 AM local time. The y-axis uses log scale. The training data are from 20 years (1984 to 2003) of 3-hourly WRF output. The lower and upper dash lines show the lowest and highest (5th and 95th percentile) PBL heights at that particular time. For example, in January 12 AM, the lowest PBL height is about 19 m, while the highest PBL height is about 365 m.

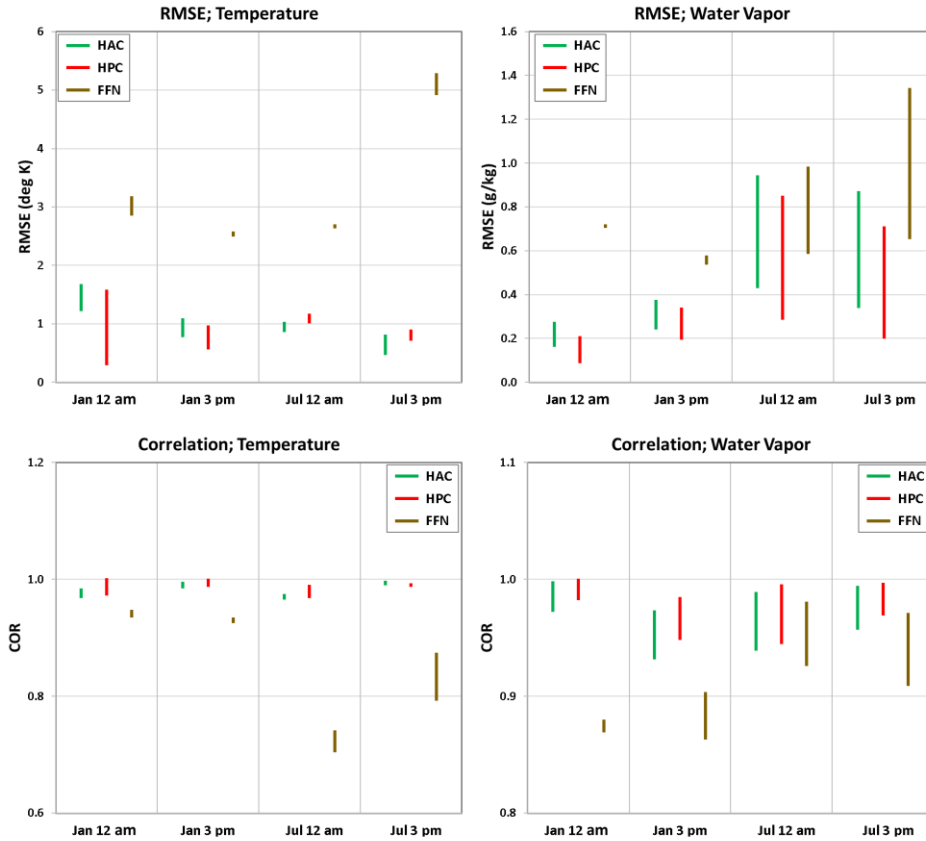
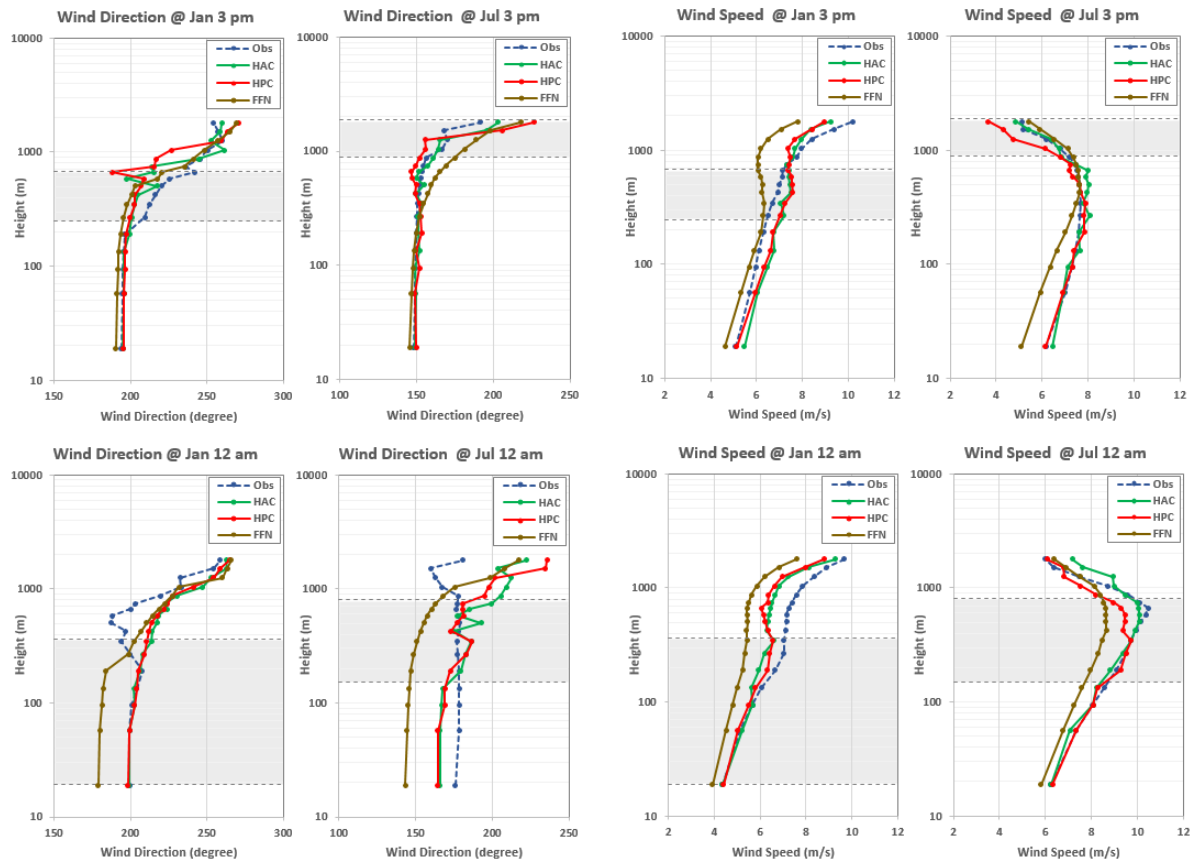


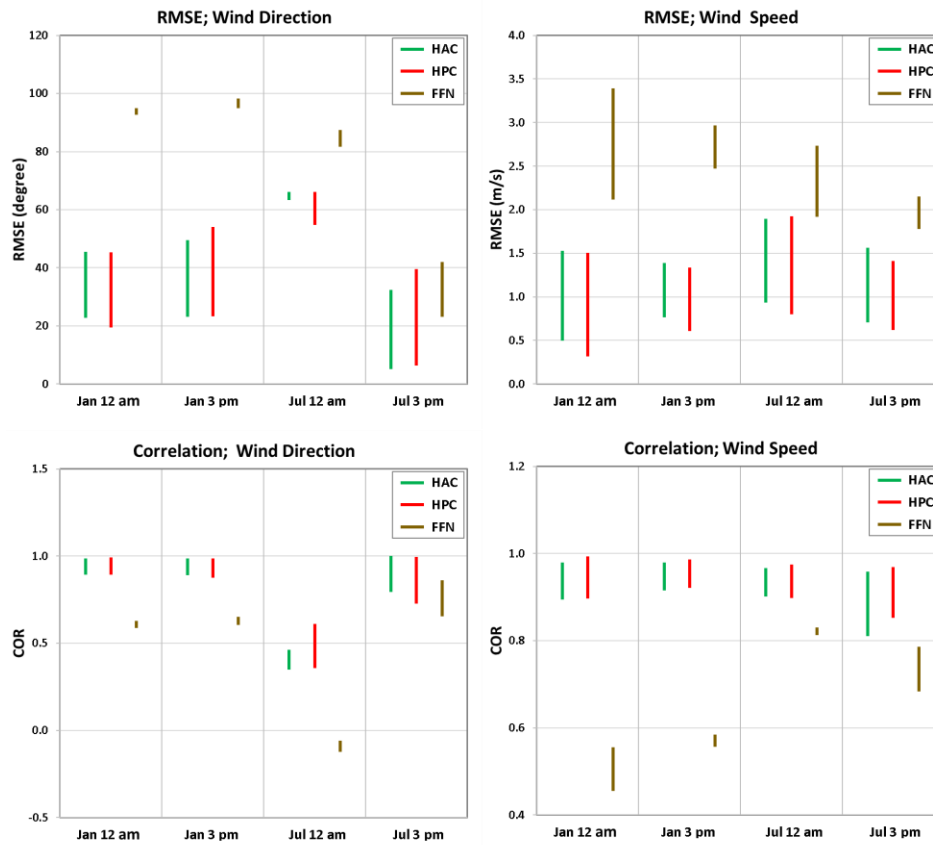
Figure 3: RMSE and correlations for time series of temperature and water vapor vertical profiles within the PBL predicted by the three DNNs compared with the observations. The vertical lines show the range of RMSEs and correlations when considering the lowest and highest PBL heights at each particular time (shown by the dashed horizontal lines in Figure 2). The training data are 3-hourly WRF output from 1984 to 2003.



617

618

Figure 4: Same as Figure 2 but for wind direction and wind speed.



619

620 **Figure 5: Same as Figure 3 but for wind components.**

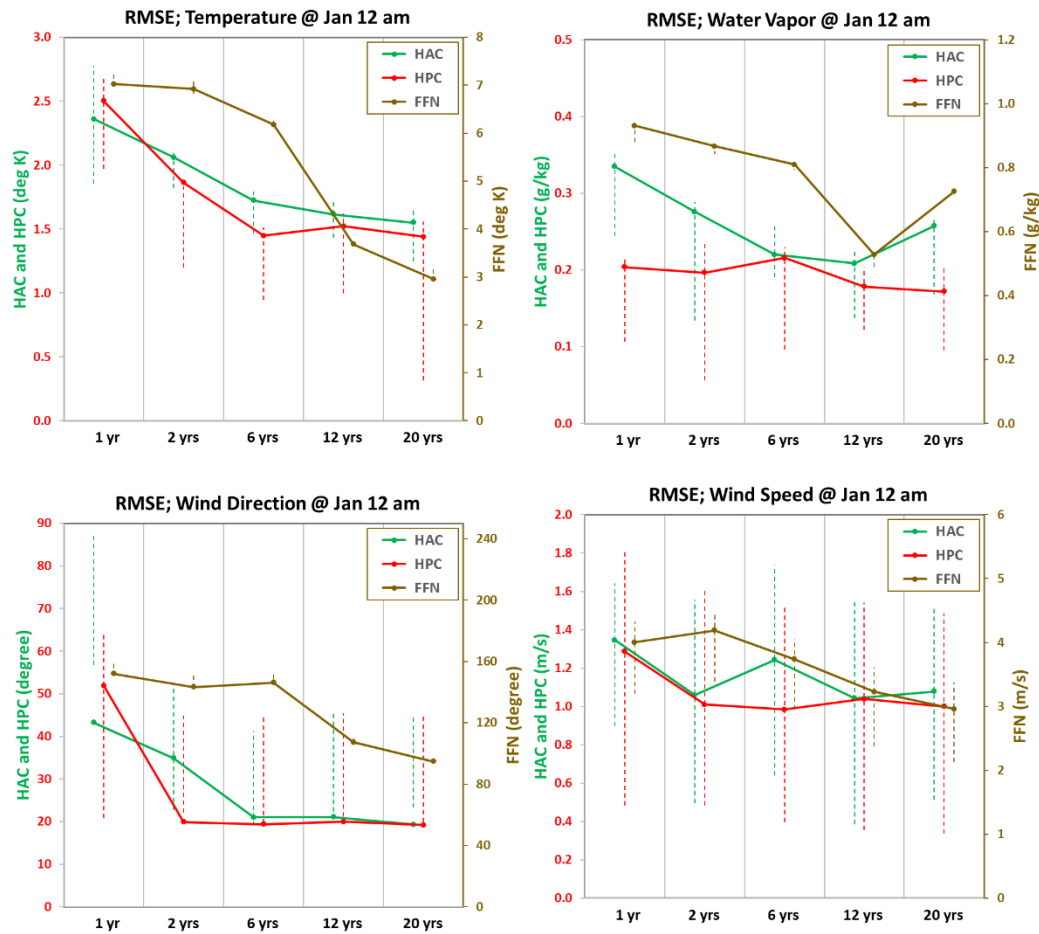


Figure 6: RMSEs for temperature, water vapor, and wind components at midnight of January using three DNNs. Left y-axis is for RMSEs of HAC and HPC; right y-axis is for RMSE of FFN. The RMSEs are calculated along the time series below the PBL height for January midnight at local time. The lower and upper end of the dash lines are RMSEs that consider the lowest and highest PBL heights as shown in Figure 2.

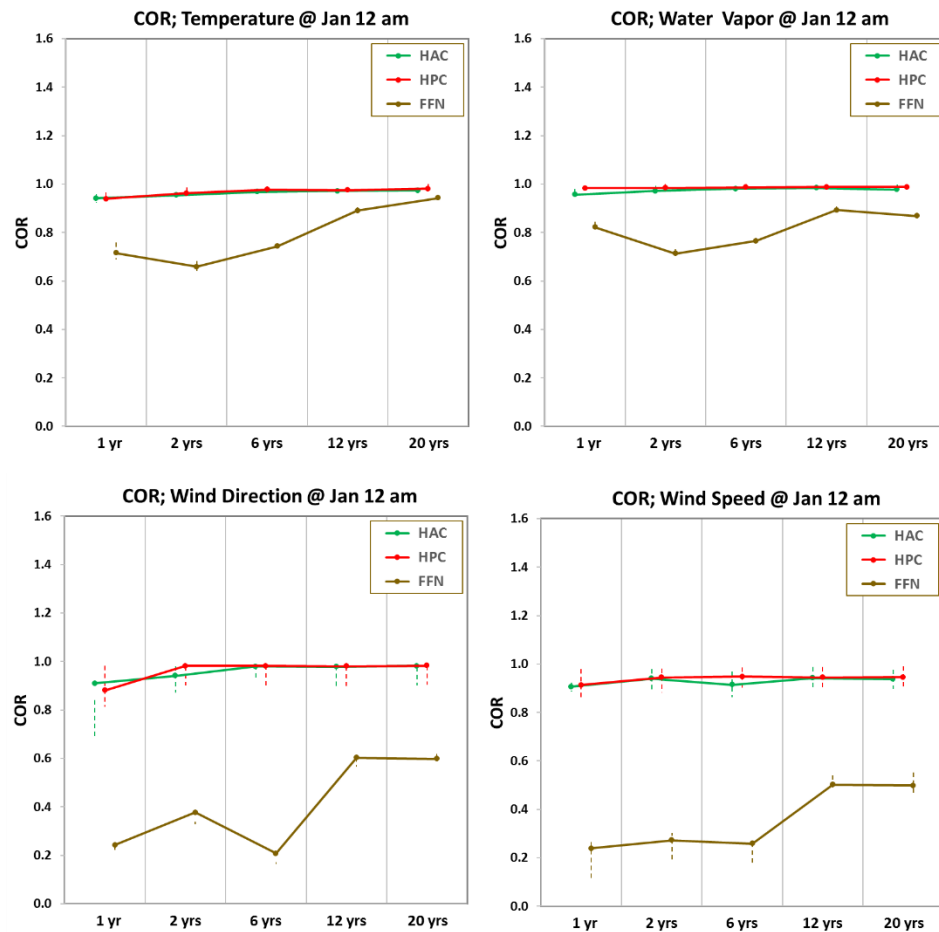


Figure 7: Same as Figure 6 but for Pearson correlations.

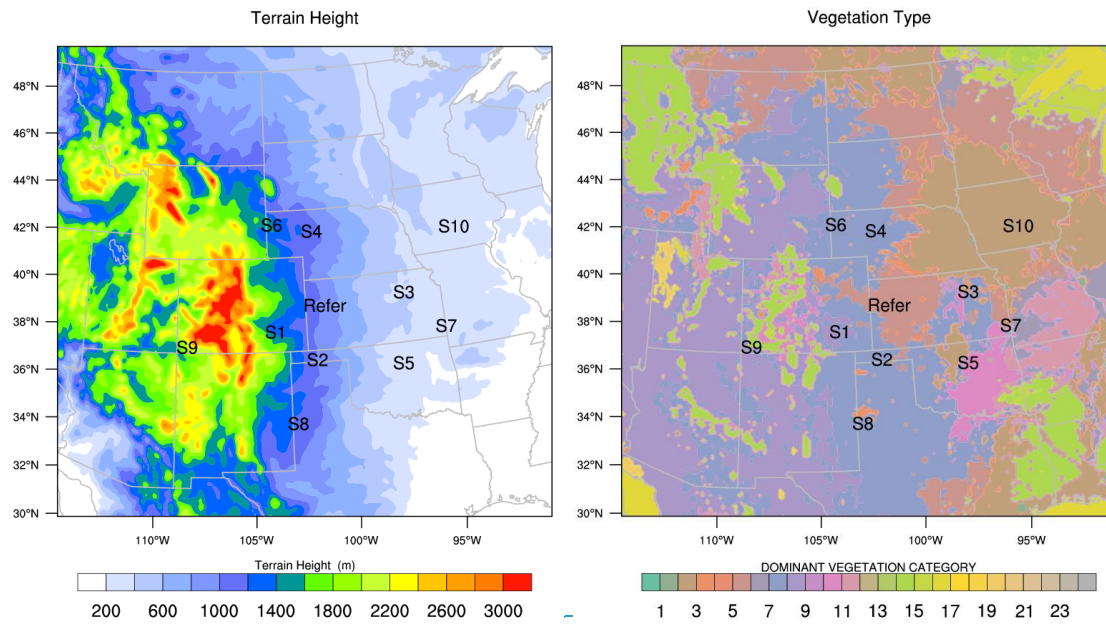
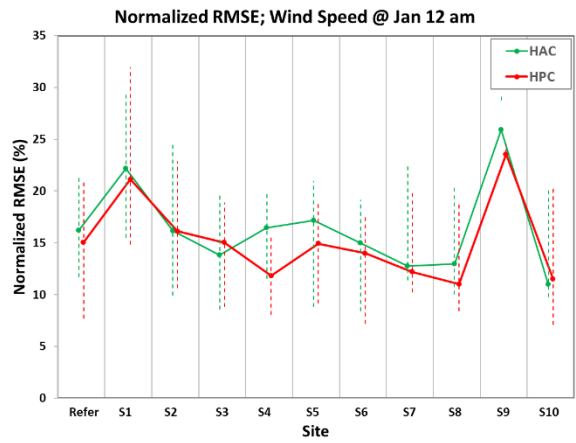
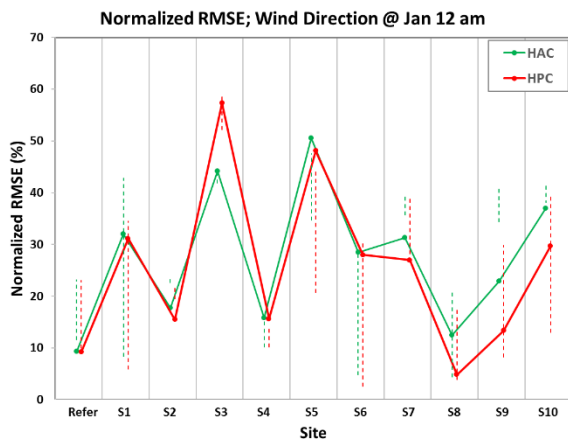
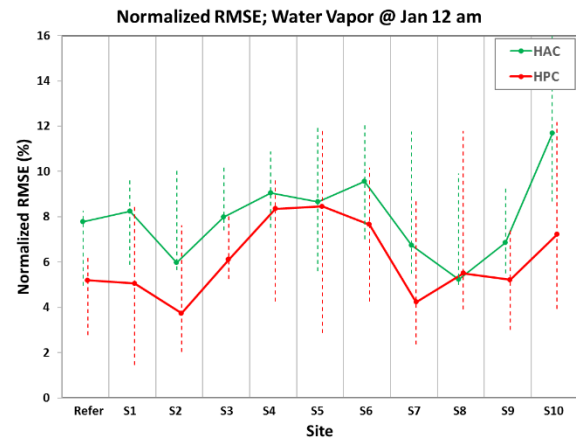
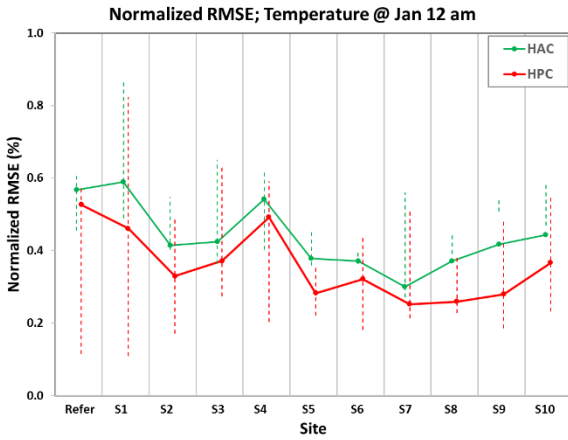
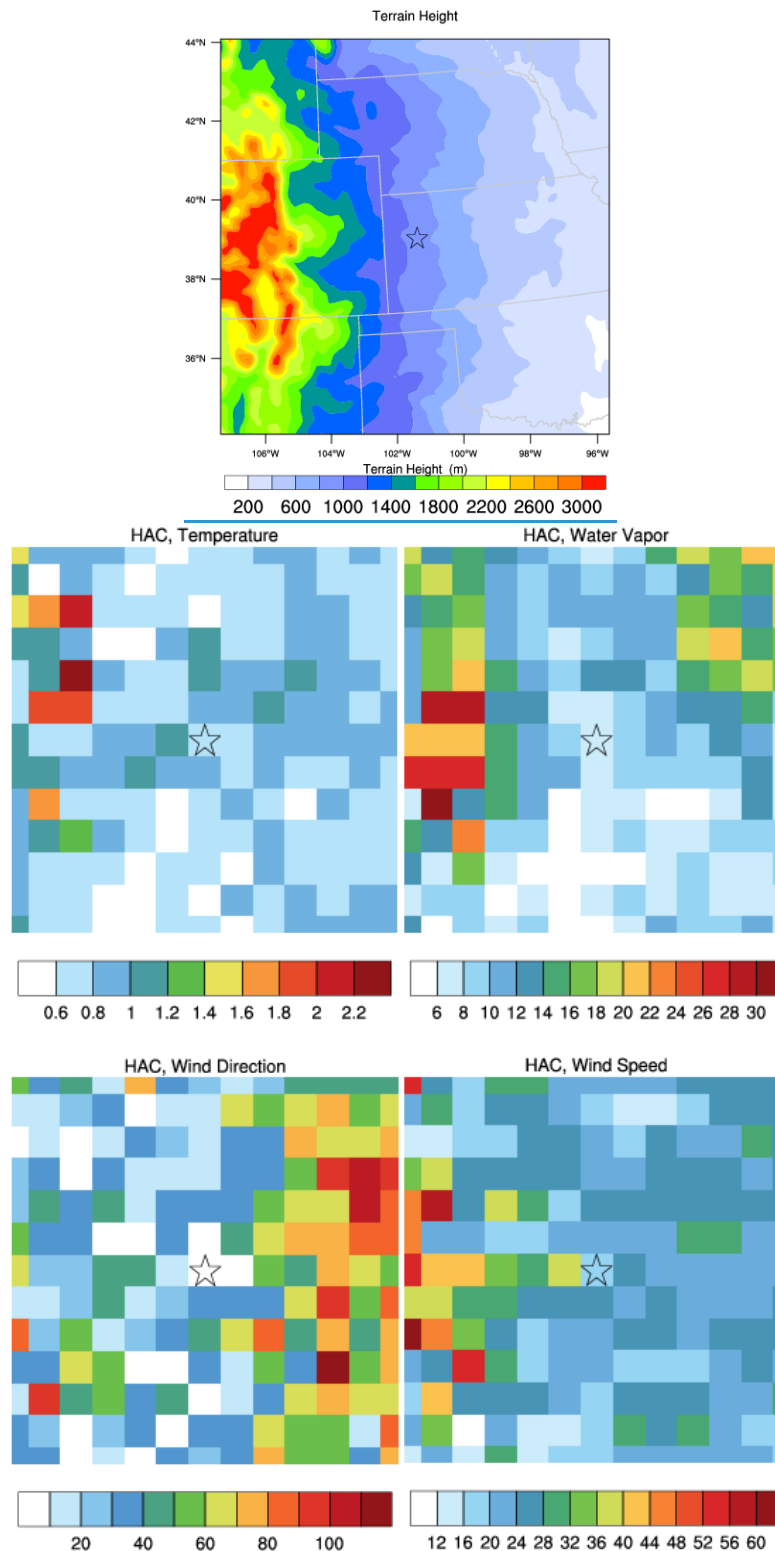


Figure 8: Terrain height (left) and vegetation types (right) for Logan, Kansas, and other locations that we used to assess the spatial transferability of our domain-aware DNNs.



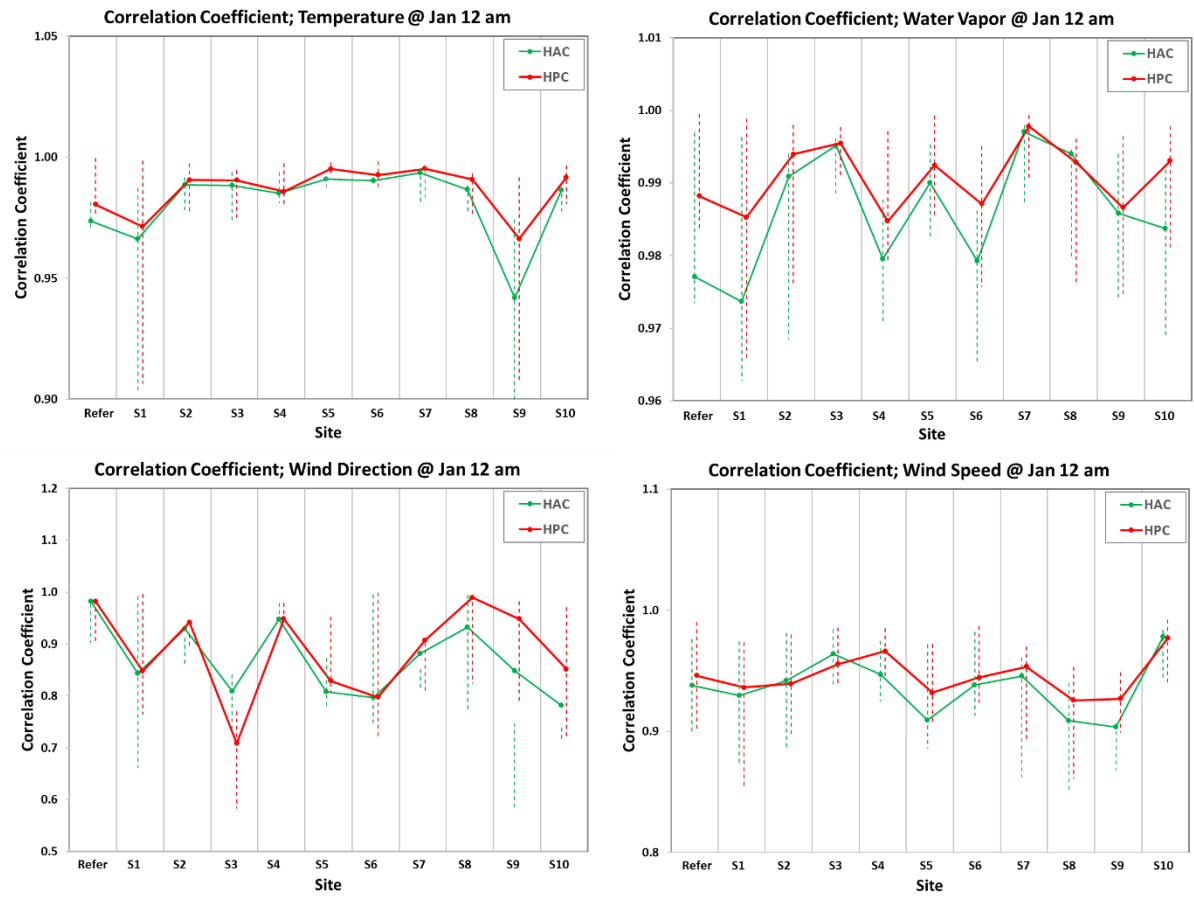
633



634

635 **Figure 98: Terrain height (in meters) over the tested area; and $N_{\text{normalized}}$ RMSEs in %**
 636 **(relative to their corresponding observations) of HAC predicted relative to their**

corresponding observations at midnight of January for temperature, water vapor mixing ratio, and wind direction and speed component at midnight of January. The star shows where the DNNs are developed sites are in the order of short to long distance from the reference site at (Logan, Kansas).



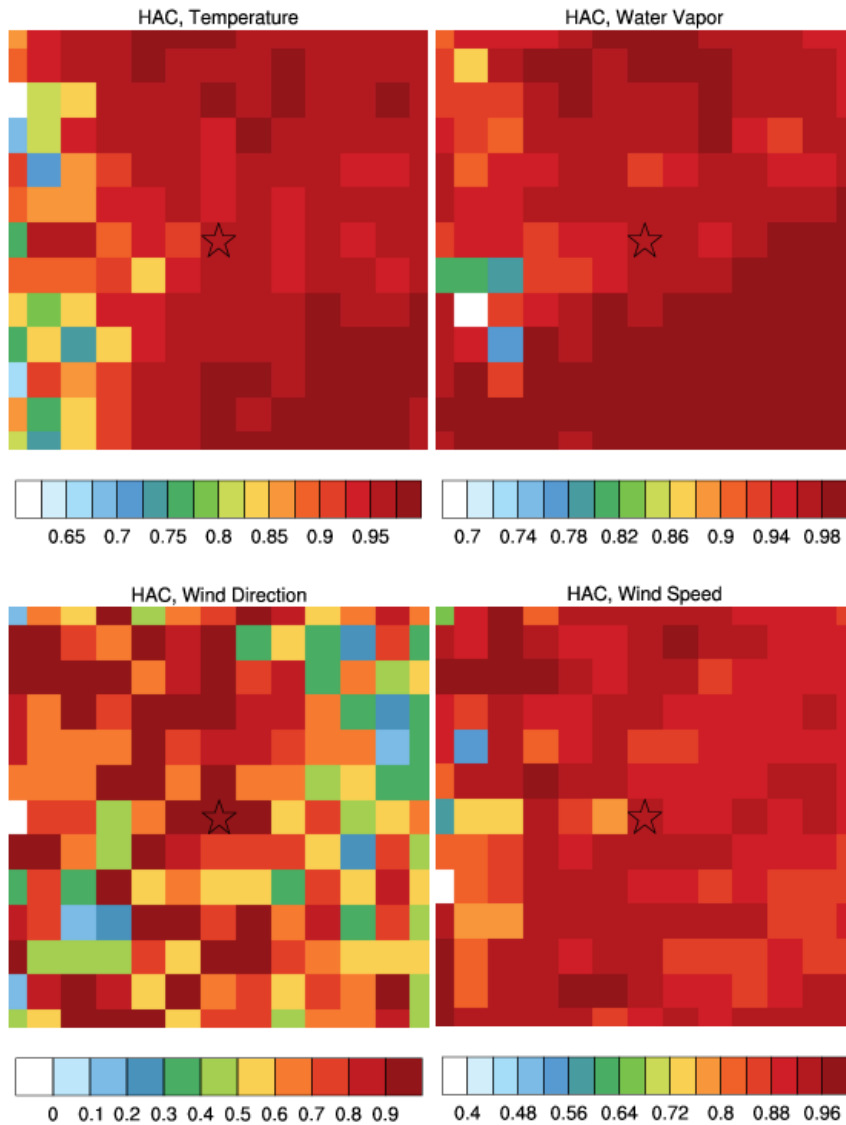


Figure 109: Same as Figure 9 but for Person correlations between observed and HAC DNN predictions temperature, water mixing ratio, wind direction and speed for midnight of January in 2005 and observations.