M.Sc. REPORT (OPTION I)

September 18, 2017

Daniel Flam-Shepherd

Supervised by Professor Dylan Jones

Predicting the Planetary Boundary Layer Depth with Bayesian Neural Networks

Contents

1 Introduction

This report is divided into six sections: the first introduces the Planetary Boundary Layer (PBL) and how to determine its depth. The next two sections introduce the machine learning algorithms used to learn and predict this depth. The fifth and sixth sections describe the implementation and results of these algorithms on two different data sources. The first data source utilized is the the Goddard Earth Observing System Model (Version 5) Forward Processing data product (GEOS-FP). The second source of PBL height data comes from a LiDAR based device, which indirectly infers them. The last section consists of a brief discussion of the overall results and a conclusion.

1.1 The Planetary Boundary Layer

The PBL is lowest region of the atmosphere, nearest to the surface of the Earth. As a result of this proximity to the surface, the PBL experiences rapid fluctuations in physical properties such as flow velocity and temperature. The resultant turbulent exchanges of momentum, heat and moisture in the PBL must be accounted for in climate and weather models if they are to accurately reflect the local evolution of the atmosphere. Above the PBL is the free atmosphere, a regime of weak vertical mixing that typically does not display turbulent phenomena. This means that the wind is close to geostrophic, which indicates that the Coriolis force is roughly in balance with the pressure gradient force.

The PBL is a regime of strong vertical mixing. Inside it, drag from the surface of the Earth causes a vertical gradient in the wind flow *i.e.* $\frac{\partial \mathbf{u}}{\partial z}$. Specifically, wind speed increases with height above the earth as a result of the no-slip condition. Surface discrepancies cause wind flow turbulence, which in turn causes vertical mixing. This process is integral to dispersion of pollutants and aerosols. The altitude of the transition out of this turbulent, mixing regime varies significantly throughout a 24 hour period. The PBL can be 100 meters during the night and upwards of 2 km during the day due to radiative forcing. The response time of the PBL to radiative forcing is quick and the the PBL will adapt to it on a hourly time scale, thus a clear diurnal variation exists in the evolution of the PBL.

Accurate models of the PBL and its processes are important for understanding weather and climate. Specifically, determination of the PBL height is important in atmospheric science for weather forecasting and understanding air pollution. However, the turbulent motions in the PBL are too small in scale and must be parametrized. Furthermore, large scale weather and climate models are very sensitive to the PBL parameterization used (*e.g.* Beljaars and Betts, 1992). In general, the PBL depth varies significantly based on the methodology used to estimate them.

Most of the material in this section follows chapter 9 of (Wallace and Hobbs 2006) and (Sugiyama and Nasstrom 1999)

1.2 PBL Constituent Layers and Types

There are two main PBL types:

① The **Stable Boundary Layer (SBL)** is typically a night time (with the exception of high latitude regions) PBL where turbulence is less prominent due to negative buoyancy flux at the surface. In this regime the free atmosphere wind helps create wind shear turbulence.

② The **Convective Boundary Layer (CBL)** is the daytime PBL which is severely impacted by heating of the Earth from incident solar radiation. During the day, turbulent mixing is strong as a result of the surface's upward heat flux. Both PBL types can exist over a diurnal period (see Figure 1 below).

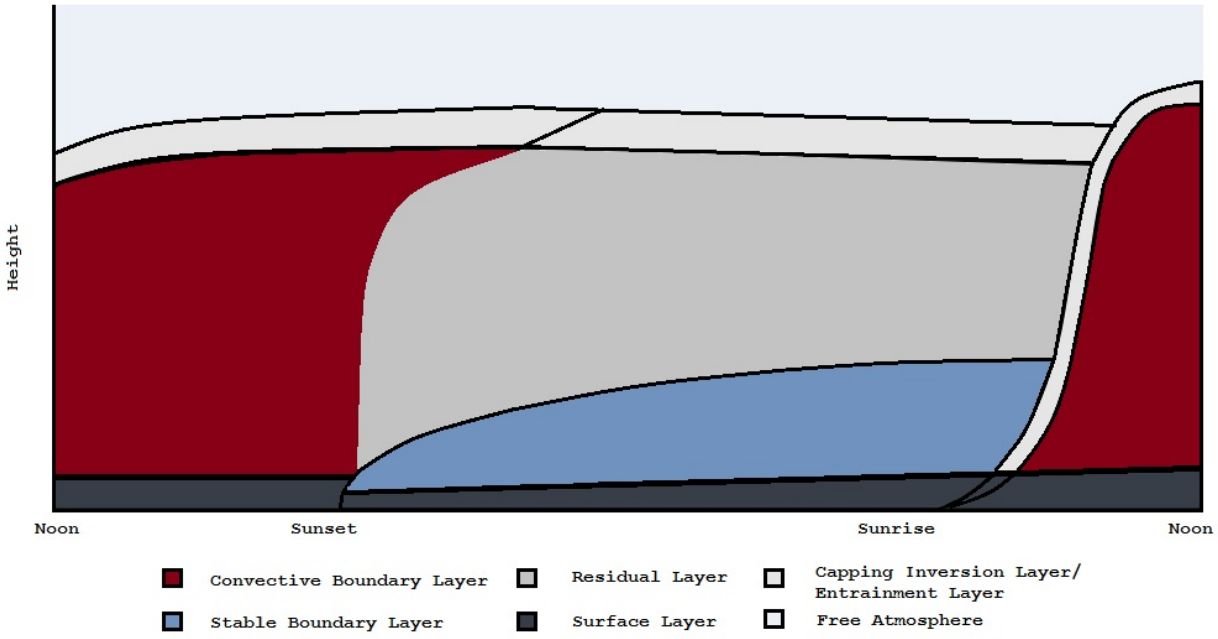

Figure 1: The diurnal evolution of the PBL.

Up to the top 30% of the PBL is called the **entrainment layer** or **capping inversion layer**, where there is significant change in atmospheric properties with increasing height as free atmosphere air entrains down into the PBL and thermal columns of air move up out of it. Here, potential temperature (the temperature which the air would have if it were brought to surface pressure without gaining or losing heat; Θ) and wind speeds sharply increase across this regime with moisture sharply decreasing.

The next component of the PBL is the **Mixing Layer**, either convective or stable, which takes up the next 40% to 80% of the PBL. Here, Θ is conserved and the capping inversion inhibits thermals from rising aloft. Below this, the **Surface Layer**, making up the bottom 10%. Here, the Θ gradient is largest, and is accompanied by strong wind shear with moisture decreasing as a function of height. The surface layer is responsible for much of the turbulence production in the PBL. However, above it turbulence dissipates, and eventually ceases. Above the SBL the remnants of the CBL form the **Residual Layer**.

1.3 Methods for Empirical Determination of the PBL height

The following are the main factors in the formation of the PBL depth and general vertical structure : ① the surface heat balance, ② the free atmosphere density stratification (when air masses of different densities act as barriers to vertical mixing), ③ the wind speed and vertical shear in the free atmosphere.

There are a number of different definitions for what actually constitutes the PBL, leading to different approaches for calculating its height h_{PBL}. For example, Hanna (1969) considers h_{PBL} the height of the lowest discontinuity in the temperature profile, while Wetzel (1982) defines h_{PBL} as the top of a layer, in which the virtual potential temperature varies linearly with height. In the following subsections we will briefly review a few methods and formulae for the empirical determination of the PBL height.

1.3.1 Temperature Profile Method: Critical Temperature Inversion

Heffter (1980) asserts that one can analyze Θ profiles for a critical elevated inversion as an indicator of the top of the PBL and thus determine its height. The inversion is the height z at which the Θ lapse rate and the difference between the inversion base and top satisfy

$$\frac{\Delta\Theta}{\Delta z} \geq \frac{0.4^{\circ}K}{10^2\ m} \quad \text{and} \quad \Delta\Theta > 2^{\circ}K$$

Thus, h_{PBL} is where, in the critical inversion layer, the temperature is 2° K above the inversion base temperature.

1.3.2 Based on Surface Fluxes : Stable Conditions

From Zilitinkevich (1972), we have the following expression for the PBL height in stable conditions:

$$h_{\mathrm{PBL}} = c_s \frac{u_* L}{|f|} \text{ when } \left|\frac{u_*}{fL}\right| > 4$$

where u_* is friction velocity, f is the Coriolis parameter, L is the Monin-Obukhov length and c_s is a proportionality constant set to 0.4. This formula predicts small values (on order of ~ 10 meters) for the PBL height in very stable conditions, thus it may also be necessary to give it a lower bound.

1.3.3 Based on Surface Fluxes : Unstable, Convective Conditions

For the daytime unstable PBL we can use slab models (Tennekes 1973) to describe the PBL and obtain h_{PBL}. These models assume that relevant physical quantities (such as temperature) have constant mean values with height (in the PBL). They also assume that the entrainment layer is infinitesimally thin and that these physical quantities have a discontinuous jump across the entrainment layer.

When buoyancy-generated turbulence is dominant we can use this rate equation to determine h_{PBL}:

$$h_{\mathrm{PBL}} \frac{dh_{\mathrm{PBL}}}{dt} \frac{\partial \Theta}{\partial z} = \beta \overline{w'\Theta'_o}$$

where $\beta = 1.4$ and $\overline{w'\Theta'_o}$ is the surface heat flux, which is assumed to be proportional to the entrainment heat flux at the PBL height. It is also assumed that heat flux varies linearly with height and the effects of latent heating, horizontal advection and large-scale vertical velocities are negligible. This approach has been used in air pollution and dispersion modeling (Sykes 1996; EPA 1995).

1.3.4 Richardson Number

One common definition of the PBL is the layer in which turbulent mixing occurs due to the presence of the ground. So h_{PBL} can be specified to be the height at which the Richardson number R exceeds a critical value R_c. The Richardson number is defined as the ratio of buoyancy to shear production of turbulence and provides a measure of the dynamic stability of the flow. The gradient Richardson number is:

$$R = (g/\Theta_v) \frac{\frac{\partial \Theta_v}{\partial z}}{\left(\frac{\partial u}{\partial z}\right)^2 + \left(\frac{\partial v}{\partial z}\right)^2}$$

where Θ_v is the virtual potential temperature, z is the height, and u, v are the horizontal wind components. Stability criteria are ① laminar flow becomes turbulent when $R < R_c \sim 0.2 - 0.5$ and ② turbulent flow becomes laminar when $R > R_T \sim 1$. The PBL height can be found by searching from the surface upwards, calculating gradients between successive levels, until R_c is exceeded. For stable conditions, h_{PBL} can be determined from the critical bulk Richardson number R_c^B, which is constant across the PBL (Hanna 1969):

$$h_{\mathrm{PBL}} = (R_c^B/g) \frac{u_h^2 + v_h^2}{(\Theta_v^h/\Theta_v^0 - 1)}$$

where $h, 0$ designates that Θ is calculated at the PBL height or ground, respectively .

1.3.5 Miscellaneous

Numerous other methods for estimating h_{PBL} exist in the literature (Holtslag et al., 1995; Vogelezang and Holtslag, 1996). Another such strategy is to use vertical wind profile criteria: such as, the height of the maximum in the low-level wind speed or the lowest level of negligible vertical wind shear. Another method come from Stull and Driedonks (1987) who used the height at which a rising parcel of surface layer air first becomes neutrally buoyant. Beljaars (1992) used a similar method but looked at excess eddy temperature at the surface from temperature profiles. However, in this report, we use machine learning algorithms to learn and predict the PBL heights. The next two sections introduce these algorithms.

2 Artificial Neural Networks

Artificial Neural Networks (ANNs) are used in this report to 'learn' to predict the PBL heights. A statistical learning algorithm based off of the biological brain, ANNs are organized into 'hidden layers' that have sets of 'hidden' neurons. As a result of this layered structure they have a strong learning capacity. An ANN composed of even just a single hidden layer can approximate any function. See Figure 2 for the general idea of their computational graph.

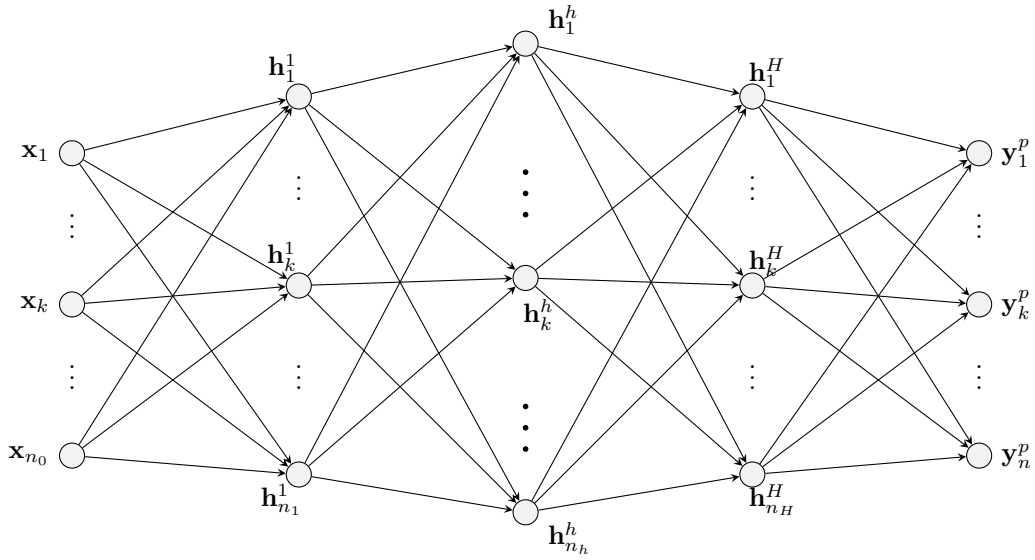

Figure 2: Example computational graph of a neural network. The arrows represents the weights.

Mathematically, ANNs map inputs $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_k, \ldots, \mathbf{x}_{n_0})$ to outputs $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_k, \ldots, \mathbf{y}_n)$ by successive layers of linear transformations through the affine function, followed by nonlinear transformations. Consider a ANN with H hidden layers and $n_h = \{n_0, n_1, \ldots, n_H\}$ neuronal units in each layer. A general layer $i \in \{1, \ldots, H\}$ has output $\mathbf{h}^i = f(\mathbf{a}^i)$ where $\mathbf{a}^i = \mathbf{W}^i\mathbf{h}^{i-1} + \mathbf{b}^i$ is the affine function of the layer weight matrix $\mathbf{W}^i \in \mathbb{R}^{n_i \times n_{i-1}}$ with bias vectors $\mathbf{b}^i \in \mathbb{R}^{n_i \times 1}$. The previous hidden layer output is $\mathbf{h}^{i-1} \in \mathbb{R}^{n_{i-1} \times 1}$. The first hidden layer is actually just the input data to the neural network $\mathbf{h}^0 = \mathbf{x} \in \mathbb{R}^{n_0 \times 1}$ and last layer of the ANN is the prediction $\mathbf{y}^p$ that the network outputs.

Each hidden layer will have the same nonlinear activation function given by some continuous function such as $f(x) = 1/(1 + e^{-x})$ or $f(x) = \tanh(x)$ or $f(x) = \max(0, x)$. The network parameters $\mathbf{W}^i, \mathbf{b}^i$ are determined by minimizing a loss function such as mean square error $\ell(\mathbf{w}) = \frac{1}{N}||\mathbf{y}^p(\mathbf{w}) - \mathbf{y}||^2$ where N is number of training samples. However, there will be no closed form expression for the parameters that minimize the loss function. Rather, we must consider an optimization algorithm, such as gradient descent, to stochastically update the network parameters until they converge : $\mathbf{w} \longleftarrow \mathbf{w} - \eta\nabla_{\mathbf{w}}\ell(\mathbf{w})$ where η is the learning rate which controls the speed and quality of the convergence of the parameters $\mathbf{w} = \{\mathbf{W}^i, \mathbf{b}^i\}_i$.

2.1 Back-Propagation and Optimization of the Network Parameters

Back-propagation is the method of calculating gradients of loss with respect to network parameters using the change rule (Rumelhart *et al.*, 1988). Back-propagation allows us to train the network so that it can construct the necessary internal representation to learn a mapping that maps the inputs to outputs. We can recursively express the weight update for each layer $\mathbf{w}^i = \{\mathbf{W}^i, \mathbf{b}^i\}$, first we calculate

$$\nabla_{\mathbf{w}^i}\ell(\mathbf{w}) = \nabla_{\mathbf{a}^i}\ell(\mathbf{w})\nabla_{\mathbf{w}^i}\mathbf{a}^i = \delta\mathbf{a}^i(\mathbf{h}^{i-1})^T$$

and the gradient of loss with respect to the ith affine transformation can be expressed as:

$$\nabla_{\mathbf{a}^i}\ell(\mathbf{w}) = \delta\mathbf{a}^i = [(\mathbf{W}^i)^T\mathbf{a}^{i+1}] \odot f'(\mathbf{a}^i) \text{ if } i < H \quad \text{and} \quad \nabla_{\mathbf{a}^H}\ell(\mathbf{w}) = \mathbf{y}^p - \mathbf{y}$$

With these equations we are back-propagating error into the network; layer-by-layer from the higher layers to the lower ones, telling each parameter how it must change at each iteration to minimize loss and maximize accuracy on the training data.

Standard stochastic gradient descent $\mathbf{w} \longleftarrow \mathbf{w} - \epsilon\nabla_{\mathbf{w}}\ell(\mathbf{w})$ is a reasonable optimization algorithm but there has been significant development of more efficient algorithms. One such algorithm is called adam (Kingma and Ba 2015), which is also a first-order gradient-based optimization algorithm for stochastic objective functions. It makes use of adaptive estimates of lower-order moments. Consider its mathematical description: first we define a step size α, then some exponential decay rates for the moment estimates $\beta_1, \beta_2 \in [0, 1)$. Then for the objective (loss) function $\ell(\mathbf{w})$ with parameters $\mathbf{w}$ initialized as $\mathbf{w}_0$.

Initialize 1st and 2nd moments $\mathbf{m}_0, \mathbf{v}_0$ then while $\mathbf{w}$ has not converged, do ① to ④

① $\mathbf{g} \leftarrow \nabla_{\mathbf{w}}\ell(\mathbf{w})$ get gradients of objective function

② $\mathbf{m} \leftarrow \dfrac{\beta_1}{1-\beta_1}\mathbf{m} + \mathbf{g}$ update bias corrected first moment estimate

③ $\mathbf{v} \leftarrow \dfrac{\beta_2}{1-\beta_2}\mathbf{v} + \mathbf{g} \odot \mathbf{g}$ update bias corrected second moment estimate

④ $\mathbf{w} \leftarrow \mathbf{w} - \alpha\dfrac{\mathbf{m}}{\sqrt{\mathbf{v}} + \epsilon}$ update parameters

The default settings for the hyperparameters are $\alpha = 0.001,\ \beta_1 = 0.9,\ \beta_2 = 0.999,\ \epsilon = 10^8$ Optimization algorithms are not guaranteed to find the global minimum of the loss function, but rather only a local minimum. Another issue is that non-convexity of the loss function can cause the optimization algorithm difficulty finding the optimal minima. The calculation of gradients in the coding of these algorithms is done using the python package autograd (Mclaurin and Duvenaud 2014). The next section introduces and describes the Bayesian counterpart to the ANN.

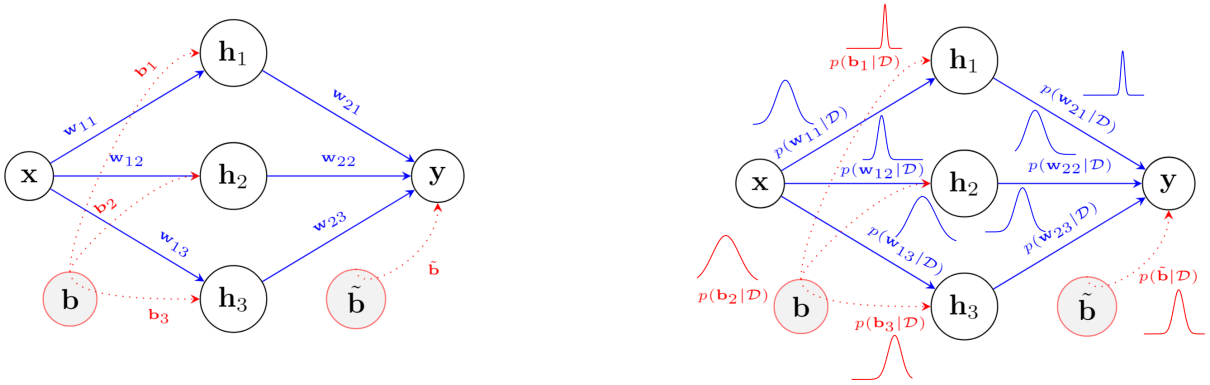

Figure 3: A neural network with discrete weights and biases (left) and a Bayesian neural net (right) with posterior distributions on the weights and biases

3 Bayesian Artificial Neural Networks

When training artificial neural networks, it is difficult to determine the best model configuration. This involves balancing the number of hidden layers and hidden neuron units with the right amount of *regularization* to control over-fitting to the training data. If the neural network over-fits and models some underlying noise in the training data, $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_i$, that is irrelevant to the actual relationship between the inputs $\mathbf{x}$ and outputs $\mathbf{y}$, then the network will have poor *generalization*. This means the network will make poor predictions on external testing data that it is not trained on.

In machine learning this is often done by setting aside training data to form a *validation set*, which can be used to determine the most optimum model complexity that performs best on the validation set. However, in the task we are considering, predicting the PBL heights, data is scarce and we cannot afford to create a meaningful validation set. This problem is solved by using Bayesian ANNs. Figure 3 depicts the difference between a neural network and a Bayesian one.

Consider the following important features of a Bayesian formulation of ANNs :

① ANNs can be seen as approximations of Bayesian ANNs

② Regularization arises naturally without any need for validation.

③ We can immediately categorize uncertainty in a Bayesian ANN.

④ Over-confident predictions from the network due to limited training data can be better controlled.

⑤ Through Bayesian model comparison we can quickly investigate network configurations.

The next section will introduce Bayesian inference and model comparison to illustrate these benefits.

3.1 Bayesian Inference and Model Comparison

Before we formulate BNNs we must introduce Bayesian inference. Consider some data $\mathbf{x}$ and associated parameters $\boldsymbol{\Theta}$ with corresponding distribution: $\mathbf{x} \sim p(\mathbf{x}|\boldsymbol{\Theta})$. $p(\boldsymbol{\Theta})$ is the *prior* probability of the parameters $\boldsymbol{\Theta}$ without knowledge of the data $\mathcal{D} = \{\mathbf{x}^{(i)}\}_i$ and $p(\mathcal{D}|\boldsymbol{\Theta})$ is the *likelihood* the data $\mathcal{D}$ given the parameters $\boldsymbol{\Theta}$. Then using Bayes rule we can find the *posterior* probability of $\boldsymbol{\Theta}$ given the data $\mathcal{D}$:

$$p(\boldsymbol{\Theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\Theta})p(\boldsymbol{\Theta})}{p(\mathcal{D})}, \quad \text{where} \quad p(\mathcal{D}) = \int p(\mathcal{D}|\boldsymbol{\Theta})p(\boldsymbol{\Theta})d\boldsymbol{\Theta}$$

where $p(\mathcal{D})$ is the distribution of $\mathcal{D}$ marginalized over $\boldsymbol{\Theta}$, also known as the marginal likelihood or the model evidence. This can be viewed as normalizing factor so that necessary information is encoded in $p(\boldsymbol{\Theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\Theta})p(\boldsymbol{\Theta})$, which gives a distribution over the parameters conditioned on the data $\mathcal{D}$. If a new data point $\hat{\mathbf{x}}$ is introduced then a predictive distribution can be formed using this posterior. By marginalizing the parameters $\boldsymbol{\Theta}$ out we have the following:

$$p(\hat{\mathbf{x}}|\mathcal{D}) = \int p(\hat{\mathbf{x}}|\boldsymbol{\Theta})p(\boldsymbol{\Theta}|\mathcal{D})d\boldsymbol{\Theta} = \mathbb{E}_{p(\boldsymbol{\Theta}|\mathcal{D})}[p(\hat{\mathbf{x}}|\boldsymbol{\Theta})]$$

This is a distribution for the new data $\hat{\mathbf{x}}$ where we can determine the uncertainty in the model from using the parameters $\boldsymbol{\Theta}$. Bayesian methods also provide a structured way of investigating model complexity. Consider models $\mathcal{M}_1, \mathcal{M}_2, \ldots$ with increasing complexity, different architectures and parameters. The relative probabilities of different models, given data set $\mathcal{D}$ is given by $p(\mathcal{M}_i|\mathcal{D}) = p(\mathcal{D}|\mathcal{M}_i)p(\mathcal{M}_i)/p(\mathcal{D}) \propto p(\mathcal{D}|\mathcal{M}_i)$ where $p(\mathcal{M}_i)$ is uniform (no initial preference to a particular model) and $p(\mathcal{D})$ is independent of $\mathcal{M}_i$. Hence only $p(\mathcal{D}|\mathcal{M}_i)$ needs to be considered to determine how likely it is that the data is modeled by each model. This provides evidence that we are using the correct model for the data.

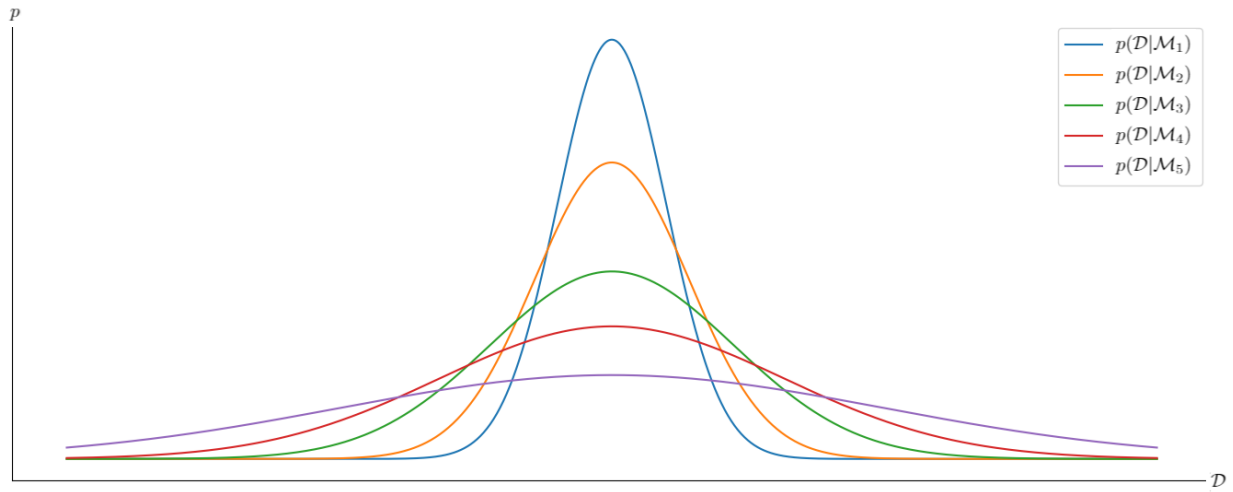

Figure 4: Comparison of models $\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, \mathcal{M}_5$. 2 and 3 are optimal.

3.2 Artificial Neural Networks as a Probabilistic Model

It is possible to view neural networks as a probabilistic model which, given an input $\mathbf{x} \in \mathbb{R}^d$, assigns a probability $p(\mathbf{y}|\mathbf{x}, \mathbf{w})$ to each possible output $\mathbf{y} \in \mathbb{R}$ through use of parameters $\mathbf{w}$. Given that we are doing regression on the PBL heights using a mean square error loss function, $p(\mathbf{y}|\mathbf{x}, \mathbf{w})$ will take the form of a normal distribution. As in the conventional framework, we will use the network parameters $\mathbf{w} = \{\mathbf{W}^i, \mathbf{b}^i\}_{i=1}^H$ to map inputs $\mathbf{h}^0 = \mathbf{x}$ through successive layers of affine transformations $\mathbf{a}^i = \mathbf{W}^i\mathbf{h}^{i-1} + \mathbf{b}^i$ and non-linear transformations $f(\mathbf{a}^i)$ to outputs $\mathbf{y}$. We will train out model on a data set of N inputs and targets $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N$, forming posterior distributions over the parameters $p(\mathbf{w}|\mathcal{D})$ in order to create a predictive distribution $p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathcal{D})$. It is possible to learn point-estimates of the network parameters using maximum likelihood estimation $\mathbf{w}^{\text{MLE}}$ or using maximum a *posteriori* estimation (MAP) $\mathbf{w}^{\text{MAP}}$ where we assume a prior distribution $p(\mathbf{w})$ for the parameters $\mathbf{w}$ for regularization.

$$\mathbf{w}^{\text{MLE}} = \underset{\mathbf{w}}{\text{argmax}} \sum_{i=1}^{N} \log p(\mathcal{D}_i|\mathbf{w}) \quad \text{and} \quad \mathbf{w}^{\text{MAP}} = \underset{\mathbf{w}}{\text{argmax}} \sum_{i=1}^{N} \log p(\mathbf{w}|\mathcal{D}_i)$$
$$= \underset{\mathbf{w}}{\text{argmax}} \sum_{i=1}^{N} \log p(\mathcal{D}_i|\mathbf{w}) + \log p(\mathbf{w})$$

If a Gaussian prior is assumed for the parameters $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbb{I})$ this corresponds to weight decay or L2 regularisation $\log p(\mathbf{w}) \propto \alpha||\mathbf{w}||_2^2$. If a laplacian prior is assumed for the parameters $p(\mathbf{w}) = L(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbb{I})$ this corresponds to L1 regularisation : $\log p(\mathbf{w}) \propto \alpha||\mathbf{w}||_1 = \sum_i |w_i|$ where $||\cdot||_p$ denotes a L^p norm.

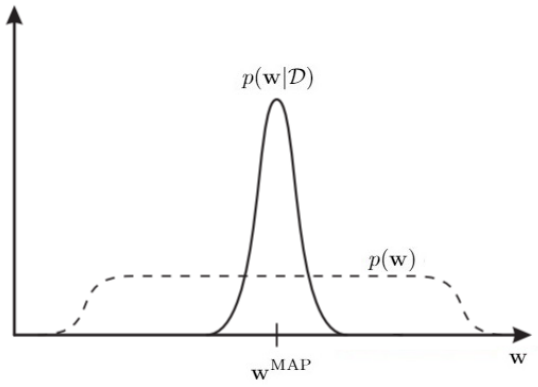

Figure 5: The idea behind point estimate of parameters is that, if the posterior distribution $p(\mathbf{w}|\mathcal{D})$ is sharply peaked about $\mathbf{w}^{\text{MLE}}$ or $\mathbf{w}^{\text{MAP}}$, like this distribution, then we can approximate the the predictive distribution via $p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathcal{D}) \approx p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w}^{\text{MAP}})$. This corresponds to a conventional approach in which predictions are made with the parameters set to a specific value.

If it is possible to approximate $p(\mathbf{w}|\mathcal{D})$ as a sufficiently narrow Gaussian we can also form a Gaussian to approximate the predictive distribution of the BNN : $p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathcal{D}) \approx \mathcal{N}(\hat{\mathbf{y}}|\mathbf{y}_{\mathbf{w}^{\text{MAP}}}, \sigma^2_{\mathbf{w}^{\text{MAP}}}\mathbb{I})$ where $\mathbf{y}_{\mathbf{w}^{\text{MAP}}}$ is the output of the NN using the MAP parameters $\mathbf{w}^{\text{MAP}}$ and $\sigma^2_{\mathbf{w}^{\text{MAP}}}$ is the corresponding uncertainty. For the PBL height observations we cannot assume that $p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathcal{D})$ is sharply peaked and we do not want to learn singular values of the parameters. Instead, for a fully Bayesian treatment of a neural network, we seek to learn a distribution over the parameters rather than a MAP estimate.

3.3 Posterior distributions of Bayesian Neural Nets

The main distribution we are interested in learning for inference with a Bayesian neural network model is $p(\mathbf{w}|\mathcal{D})$, the posterior distribution of the parameters $\mathbf{w}$ conditioned on training data $\mathcal{D}$. Using Bayes rule we can write this distribution in terms of the prior and likelihood:

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}, \quad \text{where} \quad p(\mathcal{D}) = \int p(\mathcal{D}|\mathbf{w})p(\mathbf{w})d\mathbf{w}$$

From a Bayesian point of view, learning the posterior on the parameters will mean updating our prior beliefs about them from $\mathbf{w} \sim p(\mathbf{w})$ to $\mathbf{w} \sim p(\mathbf{w}|\mathcal{D})$ as a consequence of the evidence from the training data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^N \sim p(\mathcal{D})$. The prior $p(\mathbf{w})$ is placed on the parameters on the basis that we want to keep the learned parameters small. This is done since very large parameter values mean the model has fit the training data too closely. The prior allows the network to have better generalization.

The posterior $p(\mathbf{w}|\mathcal{D})$ of the parameters given the training data is learned in Bayesian inference, allowing us to form a predictive distribution by taking expected values. This predictive distribution allows us to use BNNs to make predictions on new input data $\hat{\mathbf{x}}$. Specifically, it is the posterior over $\hat{\mathbf{y}}$ the predictions of the network using the new input after being trained on $\mathcal{D}$. To form this predictive distribution, we marginalize out the network parameters $\mathbf{w}$ or take an expectation of the distribution of the output conditioned on the input being mapped to it through the BNN parameters:

$$p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathcal{D}) = \int p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w} = \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w})]$$

Typically computing this integral requires integrating over a huge number of parameters which is computationally unfeasible. However, it is possible to use random sampling of data in the parameter space to determine an estimate through a class of inference algorithms called Markov chain Monte Carlo methods. Effectively, in this class of algorithms, the parameter space is probed for regions where $p(\mathbf{w}|\mathcal{D})$ is large enough for it to be estimated. However, these algorithms are computationally expensive. Instead we will use a significantly less expensive and more flexible class of variational inference algorithms. Regardless, the Monte Carlo estimate of the predictive posterior is:

$$\int p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w})p(\mathbf{w}|\mathcal{D})d\mathbf{w} \approx \frac{1}{n}\sum_{i=1}^{n} p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w}^{(i)}) \text{ where } \mathbf{w}^{(i)} \sim p(\mathbf{w}|\mathcal{D})$$

n is the number samples we use to estimate. We can also easily extract the posteriors associated variance which is given by $\mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[p^2(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w})] - \mathbb{E}^2_{p(\mathbf{w}|\mathcal{D})}[p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w})]$. The next subsection introduces variational inference before describing the specific algorithm we used to train BNNs to predict PBL heights.

3.4 Variational Inference

In order to evaluate $p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathcal{D})$ above we first find an approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{w})$ for $p(\mathbf{w}|\mathcal{D})$, where $\boldsymbol{\phi}$ are the variational parameters of $q_{\boldsymbol{\phi}}(\mathbf{w})$. For example, if it's Gaussian, $\boldsymbol{\phi} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$. Using samples from this posterior distribution we can obtain a Monte Carlo estimate of $p(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathcal{D})$. In variational approximations the optimal parameters $\boldsymbol{\phi}^*$ are found by minimizing the Kullback-Leibler (KL) divergence

$$\begin{aligned}\boldsymbol{\phi}^* &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\ \mathbb{KL}[q_{\boldsymbol{\phi}}(\mathbf{w})|p(\mathbf{w}|\mathcal{D})] = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \int q_{\boldsymbol{\phi}}(\mathbf{w}) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{w})}{p(\mathbf{w}|\mathcal{D})} d\mathbf{w} \\ &= \underset{\boldsymbol{\phi}}{\operatorname{argmin}}\ \mathbb{KL}[q_{\boldsymbol{\phi}}(\mathbf{w})|p(\mathbf{w})] - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}[\log p(\mathcal{D}|\mathbf{w}] = \ell(\boldsymbol{\phi}, \mathcal{D})\end{aligned}$$

This 'cost' function, denoted $\ell(\boldsymbol{\phi}, \mathcal{D})$, is known as the variational free energy of the expected lower bound. Notice that we can still get an approximation to the true posterior with a robust estimate of the uncertainty, by instead minimizing the forward KL-divergence: $\mathbb{KL}[p(\mathbf{w}|\mathcal{D})|q_{\boldsymbol{\phi}}(\mathbf{w})] = \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[\log p(\mathbf{w}|\mathcal{D}) - \log q_{\boldsymbol{\phi}}(\mathbf{w})]$. Consider a Gaussian approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{w}) = (2\pi)^{-d/2}|\boldsymbol{\Sigma}|^{-1/2} \exp\left(-\frac{1}{2}(\mathbf{w}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})\right)$. For demonstration, we can obtain expressions for the optimal variational parameters $\boldsymbol{\phi}$ since:

$$\nabla_{\boldsymbol{\phi}} \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[\log p(\mathbf{w}|\mathcal{D}) - \log q_{\boldsymbol{\phi}}(\mathbf{w})] = \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[-\nabla_{\boldsymbol{\phi}} \log q_{\boldsymbol{\phi}}(\mathbf{w})]$$

and thus because $\nabla_{\boldsymbol{\mu}} \log q_{\boldsymbol{\phi}}(\mathbf{w}) = -\frac{1}{2}\nabla_{\boldsymbol{\mu}}(\mathbf{w}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu}) = \boldsymbol{\Sigma}^{-1}(\mathbf{w}-\boldsymbol{\mu})$:

$$\mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[-\nabla_{\boldsymbol{\mu}} \log q_{\boldsymbol{\phi}}(\mathbf{w})] = \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\mathbf{w})] = 0 \iff \boldsymbol{\mu} = \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[\mathbf{w}]$$

and likewise since $\nabla_{\boldsymbol{\Sigma}} \log q_{\boldsymbol{\phi}}(\mathbf{w}) = (\mathbf{w}-\boldsymbol{\mu})^T(\boldsymbol{\Sigma}^{-1})^2(\mathbf{w}-\boldsymbol{\mu}) - \boldsymbol{\Sigma}^{-1}$:

$$\mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[-\nabla_{\boldsymbol{\Sigma}} \log q_{\boldsymbol{\phi}}(\mathbf{w})] = \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[\boldsymbol{\Sigma}^{-1} - (\mathbf{w}-\boldsymbol{\mu})^T(\boldsymbol{\Sigma}^{-1})^2(\mathbf{w}-\boldsymbol{\mu})]$$

after setting $\nabla_{\boldsymbol{\mu}} \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[-\log q_{\boldsymbol{\phi}}(\mathbf{w})] = 0$ we can use some matrix manipulation via traces to solve for:

$$\boldsymbol{\Sigma} = \mathbb{E}_{p(\mathbf{w}|\mathcal{D})}[(\mathbf{w}-\boldsymbol{\mu})(\mathbf{w}-\boldsymbol{\mu})^T]$$

Typically, a diagonal Gaussian $\boldsymbol{\Sigma} = \sigma^2\mathbb{I}$ is used for $q_{\boldsymbol{\phi}}(\mathbf{w})$. The concept of finding a variational approximation to the Bayesian posterior distribution on the parameters follows Hinton *et al.* (1993). There have been new developments in variational inference, specifically Kingma (2014) use a stochastic gradient variational Bayes estimator to perform very efficient approximate posterior inference using simple ancestral sampling to efficiently learn parameters. In the next section, this methodology is adapted for BNNs.

3.5 Variational Inference by Back-propagating through a Bayesian Neural Net

We can write the cost as $\ell(\mathbf{w}, \boldsymbol{\phi}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}[\tilde{\ell}(\mathbf{w}, \boldsymbol{\phi})]$ where $\tilde{\ell}(\mathbf{w}, \boldsymbol{\phi}) = \log q_{\boldsymbol{\phi}}(\mathbf{w}) - \log p(\mathbf{w}|\mathcal{D})$. Then we assume the parameters are a deterministic function $\mathbf{w} = g(\phi, \boldsymbol{\epsilon})$ of the variational parameters $\boldsymbol{\phi}$ and some prior noise $\boldsymbol{\epsilon} \sim q(\boldsymbol{\epsilon})$. Thus we can make a change of variable $q(\boldsymbol{\epsilon})d\boldsymbol{\epsilon} = q_{\boldsymbol{\phi}}(\mathbf{w})d\mathbf{w}$ and calculate:

$$\begin{aligned}\nabla_{\boldsymbol{\phi}}\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}[\tilde{\ell}(\mathbf{w}, \boldsymbol{\phi})] &= \nabla_{\boldsymbol{\phi}} \int \tilde{\ell}(\mathbf{w}, \boldsymbol{\phi}) q_{\boldsymbol{\phi}}(\mathbf{w}) d\mathbf{w} = \nabla_{\boldsymbol{\phi}} \int \tilde{\ell}(\mathbf{w}, \boldsymbol{\phi}) q(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon} \\ &= \mathbb{E}_{q(\boldsymbol{\epsilon})}[\nabla_{\mathbf{w}} \tilde{\ell}(\mathbf{w}, \boldsymbol{\phi}) \nabla_{\boldsymbol{\phi}} \mathbf{w} + \nabla_{\boldsymbol{\phi}} \tilde{\ell}(\mathbf{w}, \boldsymbol{\phi})]\end{aligned}$$

The function $g(\boldsymbol{\phi}, \boldsymbol{\epsilon})$ transforms a sample of prior noise and the approximate posterior parameters $\boldsymbol{\phi}$ into a sample from the variational posterior. This is simply the re-parameterization trick from Kingma and Welling (2014) applied to the parameters of a BNN rather than the approximate posterior distribution of latent variables in their paper $\mathbf{z} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$. We will employ Monte Carlo sampling to evaluate the expectations in $\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}[\tilde{\ell}(\mathbf{w}, \boldsymbol{\phi})]$. In this we will use back-propagation on the variational objective $\ell(\mathbf{w}, \boldsymbol{\phi})$ to extract unbiased estimates of its gradients to learn a distribution over the BNN parameters. We form a Monte Carlo estimate of the variational objective with N samples from the variational approximation to the true posterior $\mathbf{w}^{(i)} \sim q_{\boldsymbol{\phi}}(\mathbf{w}^{(i)})$:

$$\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{w})}[\tilde{\ell}(\mathbf{w}, \boldsymbol{\phi})] \approx \frac{1}{N} \sum_{i=1}^{N} \tilde{\ell}(\mathbf{w}^{(i)}, \boldsymbol{\phi}) = \frac{1}{N} \sum_{i=1}^{N} \log q_{\boldsymbol{\phi}}(\mathbf{w}^{(i)}) - \log p(\mathbf{w}^{(i)}) - \log p(\mathcal{D}|\mathbf{w}^{(i)})$$

Denote the variation parameters $\boldsymbol{\phi} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$ of the Gaussian approximation to true posterior. We can construct the following optimization algorithm for learning the approximate posterior. Break up the training data into mini batches *i.e.* $\mathcal{D} = \{\mathcal{D}_i\}$ and initialize the variational parameters $\boldsymbol{\phi}$. Then for all batches $\mathcal{D}_i$ and until the parameters $\boldsymbol{\phi}$ converge do ① to ⑤ :

① $\boldsymbol{\epsilon} \sim q(\boldsymbol{\epsilon}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ sample from the prior noise distribution

② $\mathbf{w} \leftarrow \boldsymbol{\mu} + \boldsymbol{\Sigma} \odot \boldsymbol{\epsilon} = g(\boldsymbol{\phi}, \boldsymbol{\epsilon})$ using the reparameterization trick sample from $q_{\boldsymbol{\phi}}(\mathbf{w})$

③ $\tilde{\ell}(\mathbf{w}, \boldsymbol{\phi}) \leftarrow \log q_{\boldsymbol{\phi}}(\mathbf{w}) - \log p(\mathbf{w}) p(\mathcal{D}|\mathbf{w})$ compute the pseudo variational objective

④ $\nabla_{\boldsymbol{\phi}} \tilde{\ell}(\mathbf{w}, \boldsymbol{\phi}) \leftarrow \nabla_{\mathbf{w}} \tilde{\ell}(\mathbf{w}, \boldsymbol{\phi}) \nabla_{\boldsymbol{\phi}} \mathbf{w} + \nabla_{\boldsymbol{\phi}} \tilde{\ell}(\mathbf{w}, \boldsymbol{\phi})$ get gradients wrt to the variational parameters

⑤ $\boldsymbol{\phi} \leftarrow \text{adam}(\boldsymbol{\phi}, \nabla_{\boldsymbol{\phi}} \tilde{\ell}(\mathbf{w}, \boldsymbol{\phi}))$ using the adam optimization algorithm update $\boldsymbol{\phi}$

Notice that for both $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ the term $\nabla_{\mathbf{w}} \tilde{\ell}(\mathbf{w}, \boldsymbol{\phi})$ is calculated by backpropagation just as in a non-Bayesian ANN. This methodology is termed Bayes by Backprop (Blundell *et al.*, 2015). The next two sections discuss the results of implementing ANNs and BNNs on two different data sources.

4 The Goddard Earth Observing System (GEOS) Data

We first experiment with neural network models on the the Goddard Earth Observing System Model (Version 5) Forward Processing data product (GEOS-5 FP) provided by the Global Modeling and Assimilation Office (GMAO) at the NASA Goddard Space Flight Center. We are motivated to see if ANNs can emulate this model output.

The assimilation is performed at a horizontal resolution of 0.3125-degree longitude by 0.25-degree latitude through 72 vertical levels, extending to 0.01 hPa. All archived, publically available products we utilize are generated at the native resolution of the horizontal grid and most are time-averaged, with the exception of surface pressure, specific humidity and temperature, which are instantaneous. Hourly data intervals are used for two-dimensional products, while 3-hour intervals are used for three-dimensional products. These are on the models native 72-layer vertical grid.

Specifically, from the surface fields we use the hourly averages of the PBL heights, the surface incident short wave flux, and the surface skin temperature. From the dynamical fields we use the horizontal wind speeds u, v averaged every three hours. From the instantaneous fields, which are stored at the midpoint of each vertical layer, we make use of surface pressure, temperature and specific humidity. We use only the 15 lowest vertical levels of the model which are the levels relevant to the PBL. Only data from the location of latitude 44.75 N and longitude 80.3125 W (north of Toronto, Ontario) is used.

Using these physical quantities we derive the potential temperature $\theta = T\left(\frac{P_0}{P}\right)^{R/c_p}$ where $R/c_p = 0.286$ and for all relevant model layers using the hypsometric equation $P_2 = P_1 e^{g(z_1-z_2)/R\bar{T}}$. We also use central finite differences to get vertical gradients of wind and potential temperature: $\frac{\partial u}{\partial z}, \frac{\partial v}{\partial z}, \frac{\partial \theta}{\partial z}$ ie $\frac{\partial u(z_i)}{\partial z} \approx \frac{u(z_{i+1})-u(z_{i-1})}{z_{i+1}-z_{i-1}}$. However, these are at best approximations to the average value of the vertical gradients since throughout the vertical layers $z_{i+1} - z_{i-1} \sim 100$ meters .

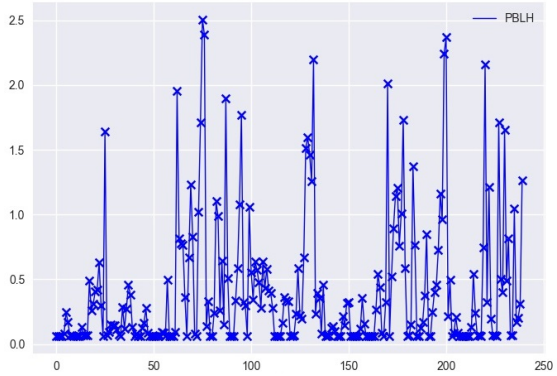

Figure 6: A time-series plot of the planetary boundary layer height during June 2014. The vertical axis is the height in kilometers and the horizontal axis is the time data. There are 248 model outputs for height in July displayed consecutively here. Each day has 8 outputs of 3 hour averaging periods. The grid from 0-50 corresponds to about the first week of July, *etc.*. The model is biased low due to some problem with the PBL mixing scheme in the GEOS-5.

Before training we scale the inputs. This is done by either unit conversion, removing the mean from the data or scaling the input range to the PBL heights. To combat the small quantity of observations per day we tested interpolation by fitting a Gaussian process to the time-series of the input and output data, however, this did not improve performance.

Results:

We train ANNs and BNNs with various combinations of physical quantities as inputs. These included skin temperature, short wave incident flux, horizontal wind speeds , temperature, pressures, specific humidity, potential temperature, vertical gradients of potential temperature, winds, and resultant wind shear magnitude (for the first 15 vertical layers of the model). Typically for training we use a learning rate of 0.001 and the default hyperparameter select for adam. Consider two example networks: one non-Bayesian (Figure 7; green curve) and one Bayesian (Figure 8; red curve). In both figures the network does not predict with any accuracy. Not a single ANN/BNN learned a mapping from the inputs to the PBL heights.

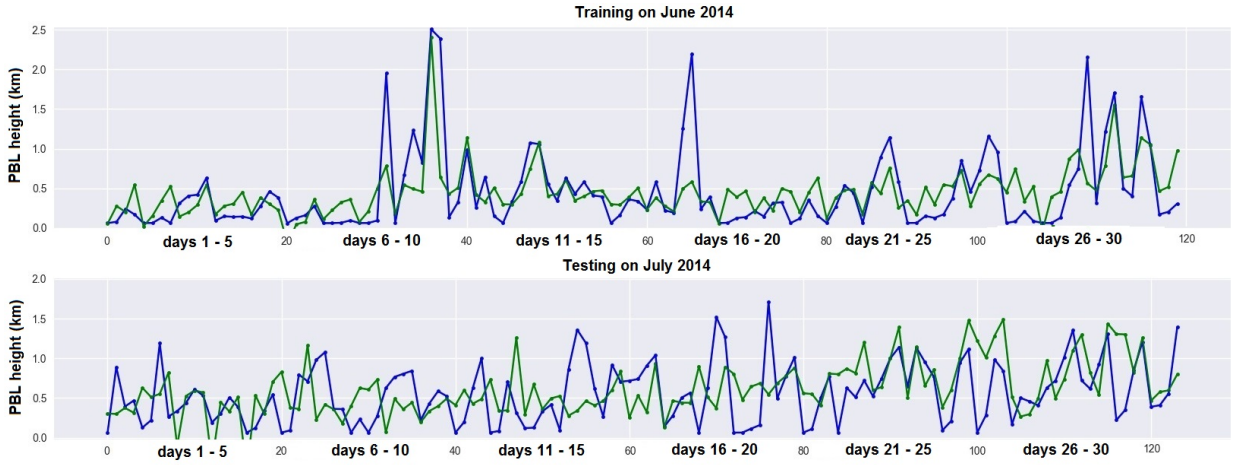

Figure 7: The blue curve is the GEOS-5 PBL height. The green curve is a 2 layer neural net with 35 neurons in each layer. The network is trained on the wind speed gradients, potential temperature gradient and wind shear magnitude; all calculated at the 5th vertical layer of the model.

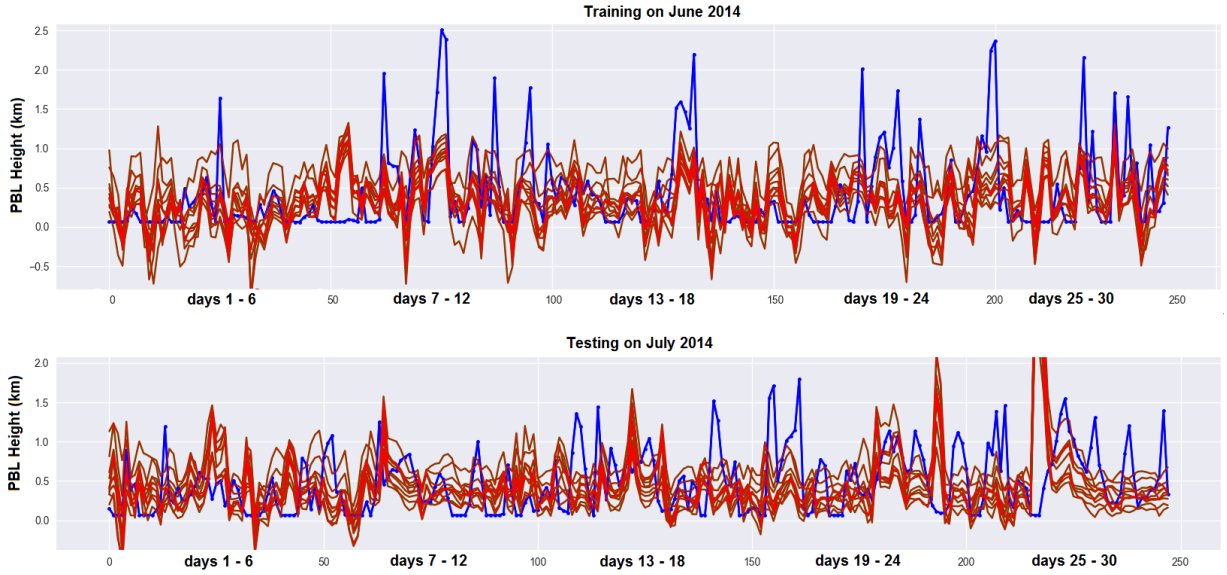

Figure 8: The blue curve is the GEOS-5 PBL height. The red curves are 8 functions sampled from the predictive distribution of a Bayesian neural network with a single hidden layer with 50 units. The network is trained on the wind speed gradients, potential temperature gradient and wind shear magnitude; all calculated at the 5th vertical layer of the model.

5 Mini-Micropulse LiDAR (MiniMPL) Data

Description of Device:

Due to the failure to emulate the GEOS 5 PBL height, we test our machine learning algorithms on real PBL height observations using data from Ware *et al.* (2016). They make use of the Sigma Space Mini-Micropulse LiDAR (MiniMPL) device to estimate the PBL heights by investigating aerosol backscatter information. The specific device used is located at the California Institute of Technology in Pasadena, California at Latitude 34.136 N, Longitude 118.127 W and 237 m ASL.

LIDAR systems can observe the distribution of aerosol in the atmosphere by measuring the backscatter of a laser from the particulate matter. This provides a vertical profile of the concentration of scattering particles. The success of the LIDAR method has been demonstrated since Coulter (1979) who found that PBL heights determined using LIDAR based measurements were similar to and sufficiently correlated with PBL heights determined from the temperature profile method.

Method for PBL height Determination:

Through the use of LIDAR, measuring the PBL height can be done in a variety of ways. Ware *et al.* make use of the wavelet method from Ehret *et al.* (1996), which itself is based on the gradient method but also takes into account the spatial scale of the boundary region at the top of the PBL. The gradient method involves searching for the minimum vertical gradient of the backscatter signal. This indicates a sudden decrease in density of scatterers. Specifically, they find the location of maximal Haar wavelet covariance where the backscatter decreases rapidly with height. This allows them to extract the boundaries between layers with high and low aerosol density. Since aerosols are concentrated within the PBL, a rapid decrease in backscatter occurs at the top of the PBL. Hence the spatial location corresponding to high wavelet covariance values identifies the PBL boundary. Utilizing this method, estimates are made every 30s estimates and are are binned into half-hour intervals. When the instrument view is obstructed by fog, clouds, or rain no estimates are made.

As a measure of confidence that their algorithm has selected the correct PBL, Ware *et al.* use the 'degree of concurrence' score. This is found while processing the days data in several dierent ways via a voting scheme modified from Lewis *et al.* (2013). They describe the 'degree of concurrence' on a scale from one to five where one and two constitute poor quality agreement and data, three is mediocre for agreement and data, and four or five is fairly good agreement and data. Approximately 75 percent of the data is classified as 3/5 or better, and during training we make use of only this data. The vast majority of the data is taken from August to January, making up 74% of the total data. January and September have the best concurrence score overall.

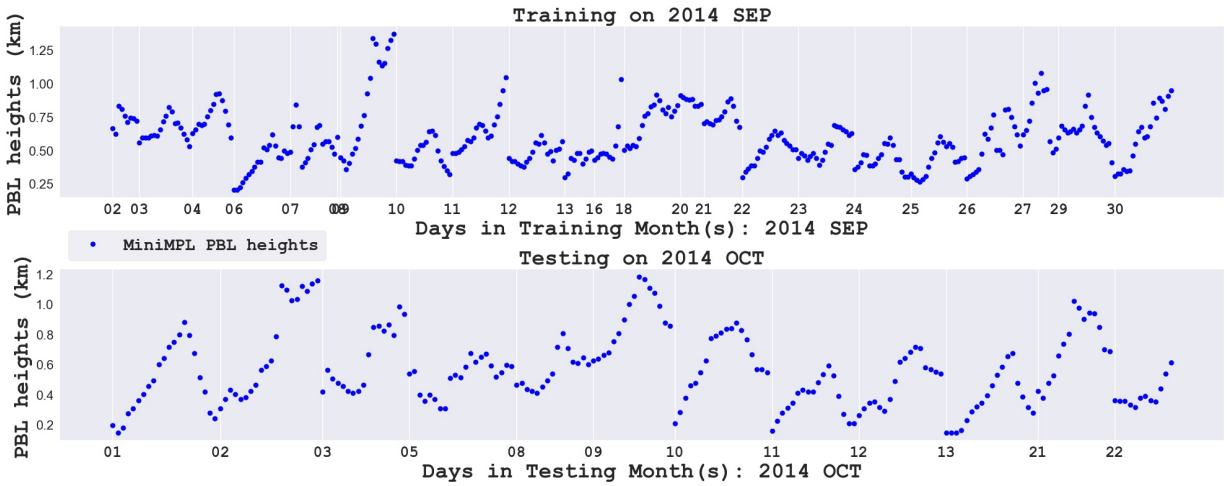

Figure 9: The time-series of the raw MiniMPL PBL heights.

5.1 Results

We train all the networks on the following 5 inputs: solar intensity, surface pressure, temperature, wind speed and the time of measurement. The data is taken from the Total Carbon Column Observing Network (TCCON), specifically from the Caltech TCCON weather station (Wunch *et al.* 2009).

The input data and the output data have two different time resolutions. For the PBL heights we have 20-30 data points a day at around 30 min apart. We have data from August 2012 to 2017 with large gaps scattered throughout. For the input data there are an inconsistent number of data available each day; it ranges from 0-400 with a resolution of 3 minutes. We have data from September 2012 to March 2017, also with large gaps scattered throughout. Due to the inconsistencies, the data is aligned within fifteen minute windows. These inconsistencies also cause gaps in the training and testing periods of the neural networks, sometimes only a few data points are available per day. As a result of this, dramatic shifts in the time-series of the PBL heights can be occasionally seen. These jumps and data gaps are clearly evident in Figure 9, in particular the testing period has a noticeable gap from day 13 to 21. There is little MiniMPL data available during the night and so little of the stable PBL data is plotted. This also causes jumps in the PBL height time series between days.

Before training, the input data is scaled by converting units so all the data has the same order of magnitude: 10^0. During experimentation it quickly became evident that the BNN gave superior quality results and was more useful in uncertainty quantification. The distributions used are all diagonal Gaussians:

$$p(\mathbf{w}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}/10), \quad p(\mathcal{D}|\mathbf{w}) \sim \mathcal{N}(||\mathbf{y} - \mathbf{t}||^2/|\mathcal{D}|, \mathbf{I}/10),$$
$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}) = E_{p(\mathbf{w}|\mathcal{D})}[\mathcal{N}(\mathbf{W}^H \mathbf{h}^{H-1} + \mathbf{b}^H, \mathbf{I})]$$

We only use one function sampled from the predictive distribution in the subsequent results.

5.1.1 Example Result : Winter

Two examples of how the BNNs perform in winter on the data are plotted below in Figures 10 and 11. The training period (top) and testing period (bottom) are both plotted. However, the training period is less relevant, we are most interested on the testing data which the network has not seen. Thus, we will mostly analyze the performance of the BNNs on the testing period where the BNN is making predictions forward in time. In Figure 10 the BNN is able predict the daily trend of the PBL in the testing period reasonably well but performs poorly on days 16, 17 and 19. On days 24 and 25 it captures the trend but is biased low later in the day. This is because during training the network has not fit well to days with a similar trend later in the day. In Figure 11 the network predicts fairly accurately the trend throughout the testing period, except for day 19 which it misses by half a kilometer. The success of BNN in the testing period of Figure 11 is surprising since the network is not closely optimized to the training data. The BNNs in Figure 10 and 11 miss the MiniMPL testing heights, on average, by 90 m and 100 m respectively.

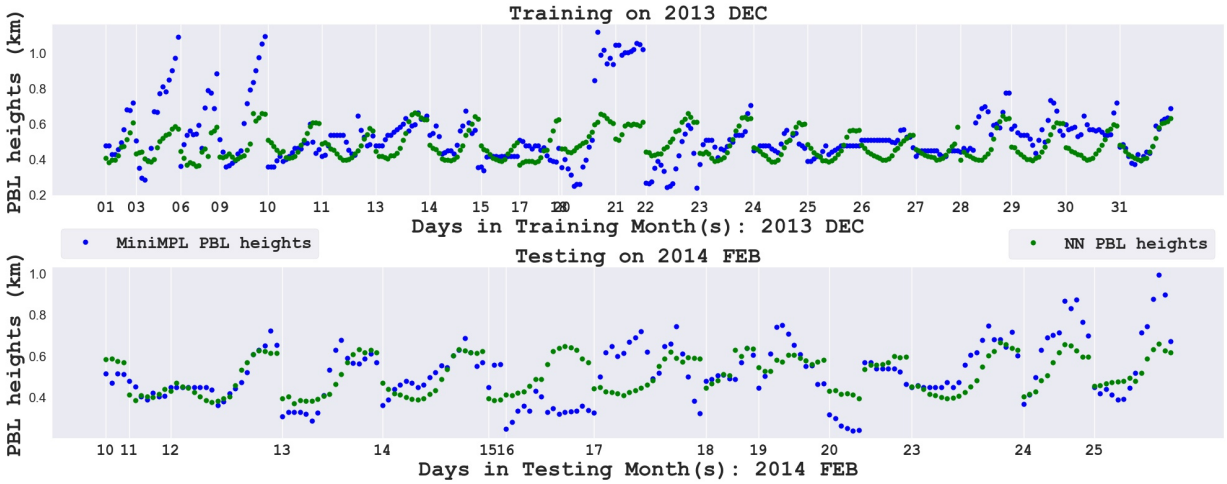

Figure 10: The TS of the MiniMPL and BNN PBLH, the BNN has two hidden layers with 36 units. $\bar{h}_{\text{diff}} = 90\text{m}$

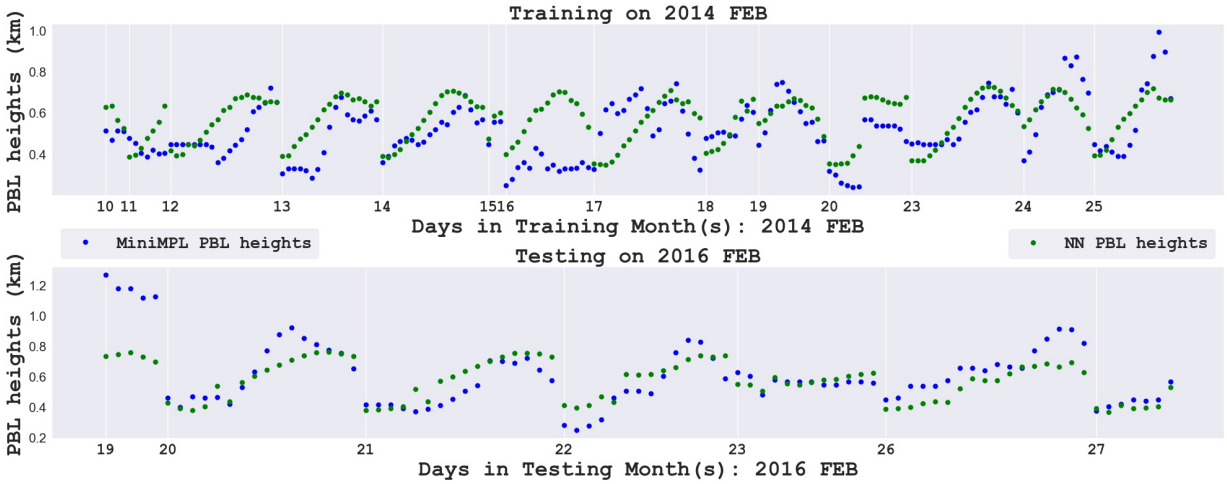

Figure 11: The TS of the MiniMPL and BNN PBLH, the BNN has two hidden layers with 21 units. $\bar{h}_{\text{diff}} = 100\text{m}$

5.1.2 Example Result : Spring and Summer

Another two examples of BNNs are plotted in Figures 12 and 13, the first is an example of training in early spring, the next is an example in late spring/early summer. The first example (Figure 12) captures most of the diurnal evolution of the PBL height during the testing period. However the BNN seems to fail to capture the trend later in the day, this is evident on days 3,7,8. The BNN also has poor performance throughout the entirety of days 22-24. The BNN also does not make any predictions above 800 meters while the actual heights exceeded 1 km. The next BNN example is later in spring. During the testing period the BNN quite accurately predicts the PBL depths, but only for the first nine days. After this, there is a three day gap (from days 9 to 12) in the data and then the network fails to accurately predict the PBL heights for three days. The network improves on the last 2 days, but is still inaccurate. Once again the BNN does not make predictions above 800 meters. The BNNs in Figure 12 and 13 miss the MiniMPL testing heights, on average, by 132 m and 119 m respectively.

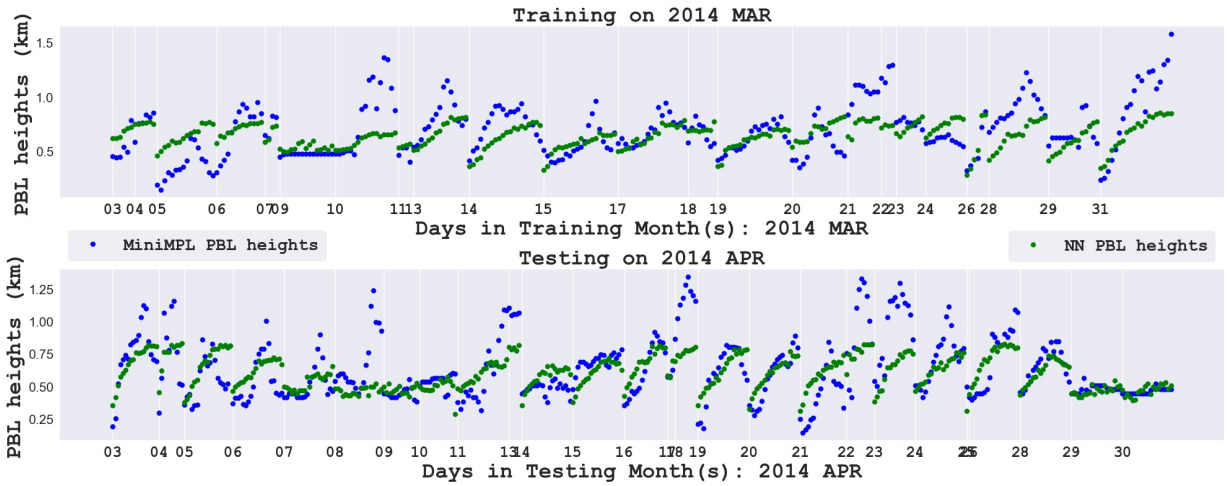

Figure 12: The TS of the MiniMPL and BNN PBLH, the BNN has two hidden layers with 12 units. $\bar{h}_{\text{diff}} = 132\text{m}$

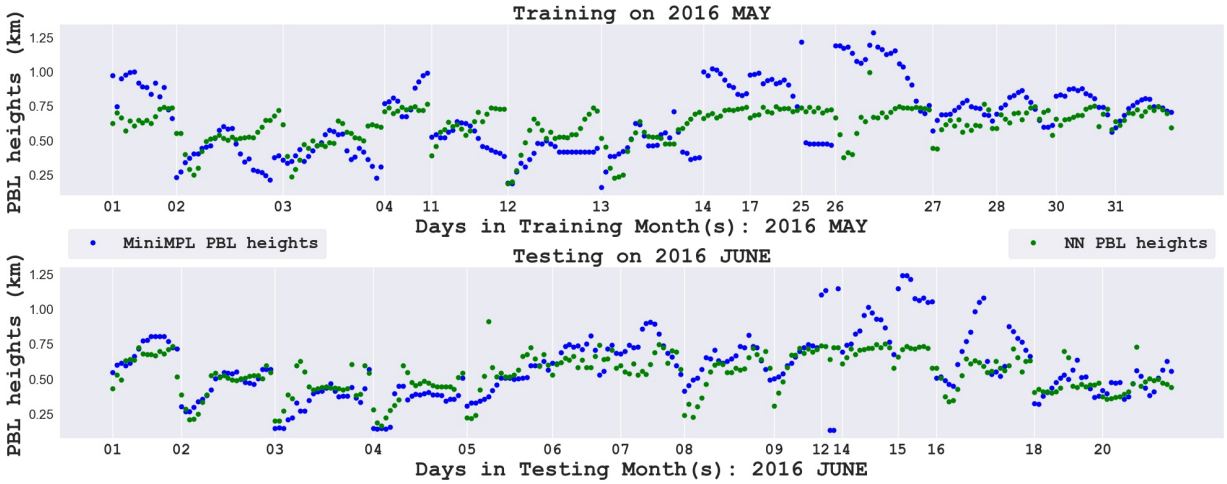

Figure 13: The TS of the MiniMPL and BNN PBLH, the BNN has two hidden layers with 15 units. $\bar{h}_{\text{diff}} = 119\text{m}$

5.1.3 Example Result : Fall

Fall had the most MiniMPL data with the highest concurrence scores. Just as before, Figure 14 and 15 have plots of two BNNs trained during the fall. Both are trained in September and tested in October, but during different years. In the testing period of Figure 14, the BNN accurately predicts most of the diurnal trend for days where the PBL depths have more moderate variation. Overall, the BNN does learn to predict the diurnal trend in the testing data, except for days 11 and 29. On days 2, 7, 8, 12, 13 and 30 the BNN predicts the trend but underestimates the rate the PBL heights rise during the day. The testing MiniMPL PBL heights are also biased low. For Figure 15, during the testing period the BNN accurately predicts both the trend and fairly closely matches the MiniMPL PBL heights for most days. Only on day 9 does does it underestimates the heights but still accurately predicts the trend. Training on this data, the BNN does make predictions close to the higher MiniMPL PBL heights at around 1 km. The BNNs in Figure 14 and 15 miss the MiniMPL testing heights, on average, by 149 m and 127 m respectively.

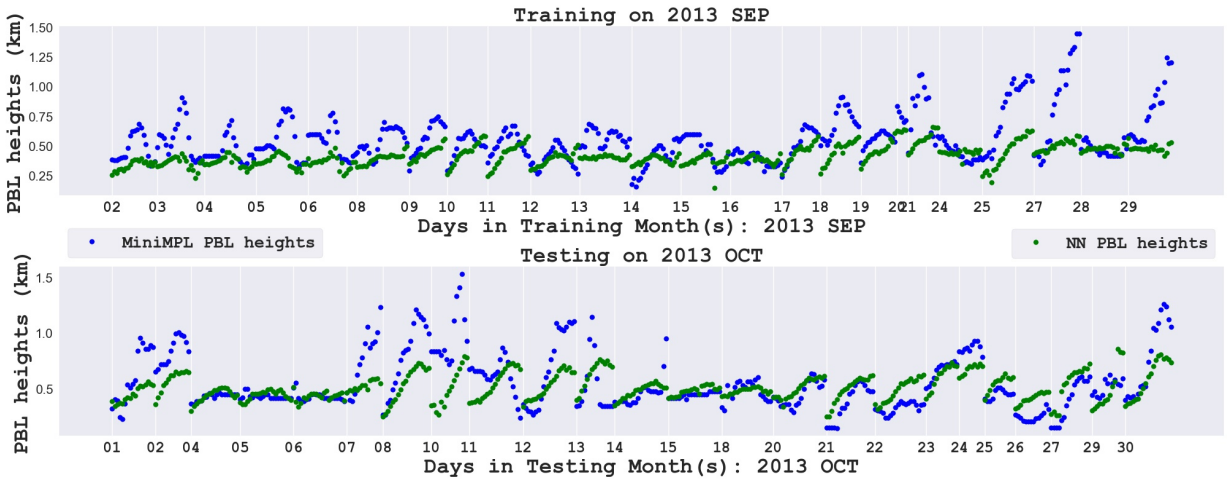

Figure 14: The TS of the MiniMPL and BNN PBLH, the BNN has two hidden layers with 16 units. $\bar{h}_{\text{diff}} = 149\text{m}$

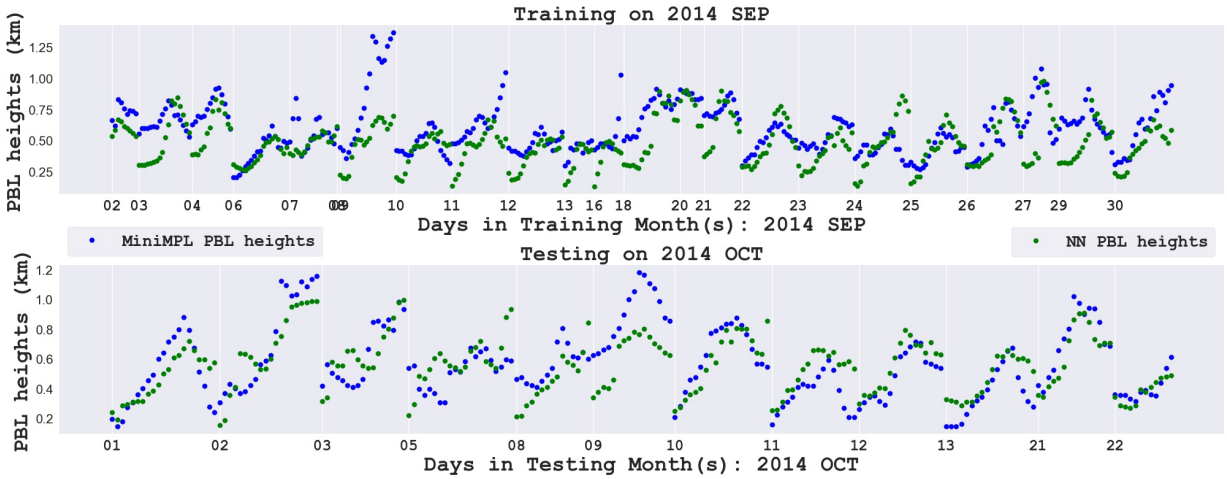

Figure 15: The TS of the MiniMPL and BNN PBLH, the BNN has two hidden layers with 36 units. $\bar{h}_{\text{diff}} = 127\text{m}$

6 Discussion of results and Conclusion

Discussion of Results

The results of the previous section demonstrate, with sufficient data we are able to train BNNs to do nonlinear regression on the PBL heights using around one month of solar intensity, external surface pressure, external temperature, wind speed and time of day as input data. The trained BNNs are able to predict the next month's PBL height using the same fields with an accuracy of, on average, less than 150 meters. The BNNs typically can predict the diurnal evolution of the PBL depth, although sometimes they make predictions that are biased low. We are able to make reasonable predictions with the BNNs all year round. In every seasons we can make predictions forward in time, however, the fall PBL height diurnal trends are better predicted than other seasons. This is due to the larger amount of quality data available.

Training on more than one month did not lead to improved results. The most successful network architectures had 2 hidden layers with 10-40 hidden units each. More hidden layers and units does not leads to improved performance. Training the BNN using 'stochastic' variational inference leads to a very noisy convergence of the approximate posteriors parameters $\boldsymbol{\phi}$ in $q_{\boldsymbol{\phi}}(\mathbf{w})$. This also causes the PBL height predictions to often oscillate around their optimal values before convergence.

Conclusion

This report describes machine learning methods to do nonlinear regression on the PBL height, which is the depth of the lowest, most turbulent, portion of the atmosphere. The main algorithm of choice was the Bayesian counterpart to the artificial neural network. We began the report with a discussion of what the PBL is, its forms, and its constituent layers. Then, briefly, methods to calculate the PBL height in its different forms were described. Section 2 and 3 introduces artificial neural networks and their Bayesian formulation, as well as their gradient based training strategies. Sections 4 and 5 detailed their use, on first the GEOS FP model output PBL height, then the PBL heights detected by the MiniMPL device. The first attempt proved futile and the model output could not be emulated by a neural network, Bayesian or otherwise. The later implementation proves reasonably successful. These results demonstrate that it is possible for a BNN to learn to predict the diurnal evolution of the planetary boundary layer depth over the period of a month using physical quantities commonly measured at weather stations. The BNNs are able to learn the relationship between the physical quantities used for inputs and the PBL height. The trained BNN are able to forecast next month's PBL heights with different degrees of accuracy depending on time of year and quality of data.

References

[1] Beljaars, A. C. M. and A. K. Betts, 1992: Validation of the Boundary Layer Representation in the ECMWF Model, Seminar Proceedings Validation of Models over Europe, Vol II, Reading, UK, 7-11 September 1992

[2] Blundell, Charles, Cornebise, Julien, Kavukcuoglu, Koray, and Wierstra, Daan. Weight uncertainty in neural networks. arXiv preprint arXiv:1505.05424, 2015.

[3] David Duvenaud, Ryan P. Adams Black-box stochastic variational inference in five lines of Python NIPS Workshop on Black-box Learning and Inference, 2015

[4] D. Wunch, G.C. Toon, J.-F.L. Blavier, R.A. Washenfelder, J. Notholt, B.J. Connor, D.W.T. Griffith, V. Sherlock, P.O. Wennberg. The Total Carbon Column Observing Network. Phil. Trans. R. Soc. A (2011) 369, doi:10.1098/rsta.2010.0240

[5] Ehret, G., A. Giez, C. Kiemle, K. J. Davis, D. H. Lenschow, S. P. Oncley, and R. D. Kelly (1996), Airborne water vapor DIAL and in situ observations of a sea-land interface, Contrib. Atmos. Phys., 69, 215228.

[6] EPA, 1995: A Users Guide for the CALMET Meteorological Model, EPA-454/B-95-002, U.S. Environmental Protection Agency, Research Triangle Park, NC

[7] Geoffrey E Hinton and Drew Van Camp. Keeping the neu- ral networks simple by minimizing the description length of the weights. In Proceedings of the 16th Annual Con- ference On Learning Theory (COLT), pages 513. ACM, 1993.

[8] Hanna, S. R., 1969: The thickness of the planetary boundary layer, Atmos. Envir. 3, 519-536

[9] Heffter, J. L., 1980: Transport Layer Depth Calculations, Second Joint Conference on Applications of Air Pollution Meteorology, New Orleans, LA (1980)

[10] Holtslag, A. A. M., E. van Meijgaard, W. C. De Rooy, 1995: A comparison of boundary layer diffusion schemes in unstable conditions over land, Boundary-Layer Met. 76, 69-95

[11] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. ICLR 2015

[12] Kingma, D and Welling, M Autoencoding Variational Bayes. ICLR 2014

[13] Lewis, J. R., E. J. Welton, A. M. Molod, and E. Joseph (2013), Improved boundary layer depth retrievals from MPLNET, J. Geophys. Res. Atmos., 118, 98709879, doi:10.1002/jgrd.50570.

[14] Nieuwstadt, F. T. M, 1981: The Steady-State Height and Resistance Laws of the Nocturnal Boundary Layer Compared with Cabauw Observations

[15] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. Cognitive modeling, 5, 1988.

[16] Stull, R. B. and A. G. M. Driedonks 1987: Applications of the transilient turbulence parameterization to atmospheric boundary-layer simulations Boundary-Layer. Met. 40, 209-239

[17] Sugiyama, Gayle and Nasstrom, John S. Methods for Determining the Height of the Atmospheric Boundary Layer

[18] Sykes, R. I. et al., 1996: PC-SCIPUFF Version 0.2 Technical Documentation, Titan Corporation, P. 0. BOX 2229, Princeton, NJ 08543

[19] Tennekes, H., 1973: A model for the dynamics of the inversion above a convective boundary layer, J. Atmos. Sci., 30, 558-567

[20] Vogelezang, D. H. P, and A. A. M. Holtsiag, 1996: Evolution and Model Impacts of Alternative Boundary Layer Height Formulations, Boundary-Layer Met. 81, 245-269

[21] J. Wallace, P. Hobbs Atmospheric Science : An Introductory Survey - (Elsevier, 2006) W.W.

[22] Ware, J., E. A. Kort, P. DeCola, and R. Duren (2016), Aerosol lidar observations of atmospheric mixing in Los Angeles: Climatology and implications for greenhouse gas observations, J. Geophys. Res. Atmos., 121, 98629878, doi:10.1002/2016JD024953.

[23] Wetzel, P. J., 1982: Toward Parameterization of the Stable Boundary Layer, J. AppE. Met. 21, 7-13