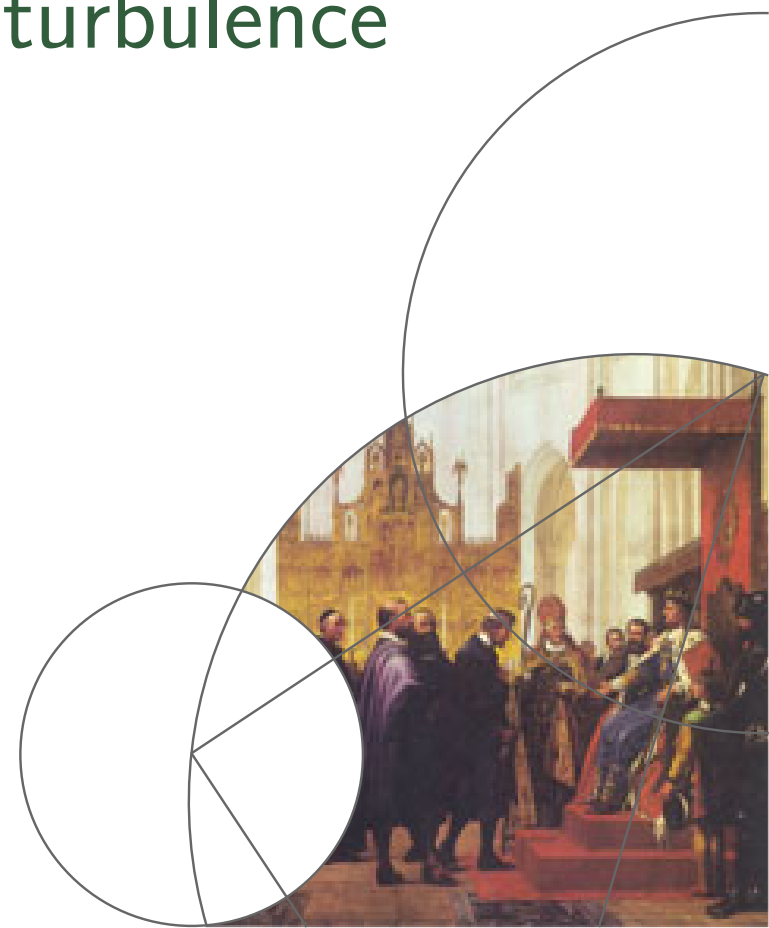
Development and implementation of a neural network based PBL turbulence parameterization scheme

Kasper Tølløse
Niels Bohr Institute

Supervisor:
Professor Eigil Kaas

March 27, 2020



Overview

- Introduction/motivation
- Planetary boundary layer
- Turbulence parameterization in NWP
- Artificial neural networks
- Development of neural network based model
- Implementation in WRF and comparison to other schemes
- Conclusion and outlook

Introduction and motivation

- Discretization of the governing equations for the atmosphere introduces a need for parameterization of dynamics on subgrid-scale.
- Turbulence parameterizations predicts turbulent flux terms - typically by assuming some mathematical relation with known mean field variables, and fitting constants to observational/simulated data.
- More advanced turbulence parameterizations are typically computationally expensive.
- A new method is proposed, where machine learning models are used to predict the turbulent fluxes. Thus, fewer assumptions are needed.

Reynolds averaging and turbulent fluxes

- Reynolds averaging distinguish between the "mean field variables" and the turbulent fluctuations.
- Methodology: Assume all variables can be written as a mean variable and a small randomly fluctuating part, $\tilde{A}' = A + a$. Then take the average of the equations to obtain (for a dry atmospheric boundary layer)

$$\frac{\partial \tilde{u}_i}{\partial x_i} = 0,$$
$$\frac{D\tilde{u}_i'}{Dt} = -\frac{1}{\rho_0}\frac{\partial \tilde{p}'}{\partial x_i} + g\delta_{3i}\frac{\tilde{\theta}'}{\theta_0} - 2\epsilon_{ijk}\Omega_j\tilde{u}_k' + \nu\nabla^2\tilde{u}_i',$$
$$\frac{D\tilde{\theta}'}{Dt} = \alpha\nabla^2\tilde{\theta}' - \frac{\tilde{\theta}'}{\tilde{\rho}c_p\tilde{T}}\frac{\partial \tilde{R}_i}{\partial x_i},$$

Reynolds averaging and turbulent fluxes

- Reynolds averaging distinguish between the "mean field variables" and the turbulent fluctuations.
- Methodology: Assume all variables can be written as a mean variable and a small randomly fluctuating part, $\tilde{A}' = A + a$. Then take the average of the equations to obtain (for a dry atmospheric boundary layer)

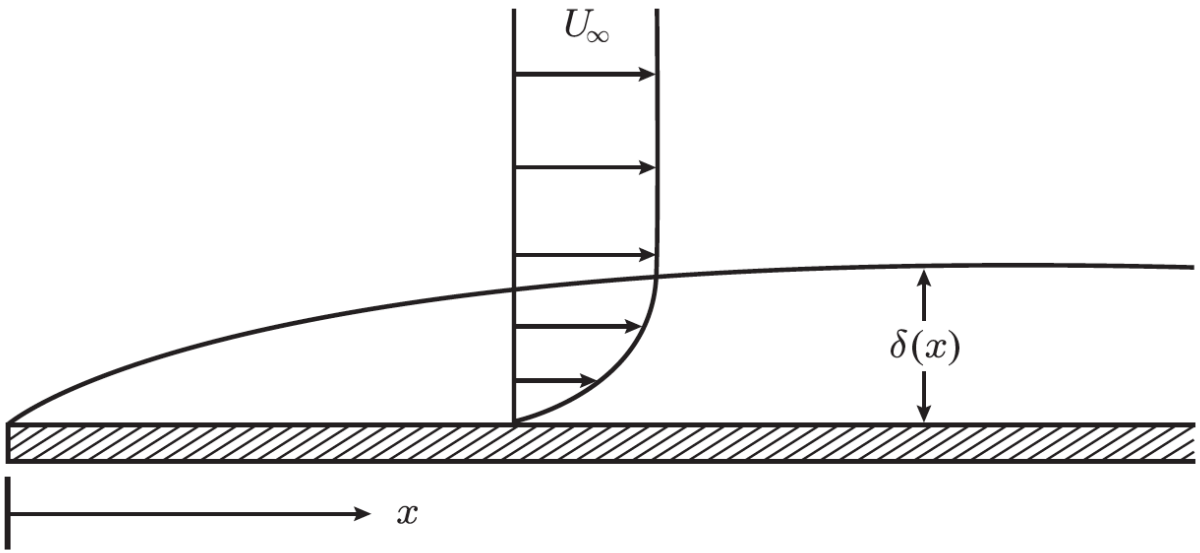
$$\frac{\partial U_i}{\partial x_i} = 0,$$

$$\frac{D_U U_i}{Dt} = -\frac{\partial}{\partial x_j}\overline{u_j u_i} - \frac{1}{\rho_0}\frac{\partial P}{\partial x_i} + g\delta_{3i}\frac{\Theta}{\theta_0} - 2\epsilon_{ijk}\Omega_j U_k + \nu\nabla^2 U_i,$$

$$\frac{D_U \Theta}{Dt} = -\frac{\partial}{\partial x_j}\overline{u_j\theta} + \alpha\nabla^2\Theta, -R \quad \text{where } R = \overline{\frac{\tilde{\theta}'}{\tilde{\rho}c_p\tilde{T}}\frac{\partial \tilde{R}_i}{\partial x_i}}$$

Planetary boundary layer

- The planetary boundary layer, *PBL*, is the lowest part of the atmosphere, which is "in contact with" the surface.
- In the PBL, turbulence is most likely to occur, and thus it tends to be well-mixed. Therefore, turbulence parameterization focus mainly on the PBL.



PBL structure

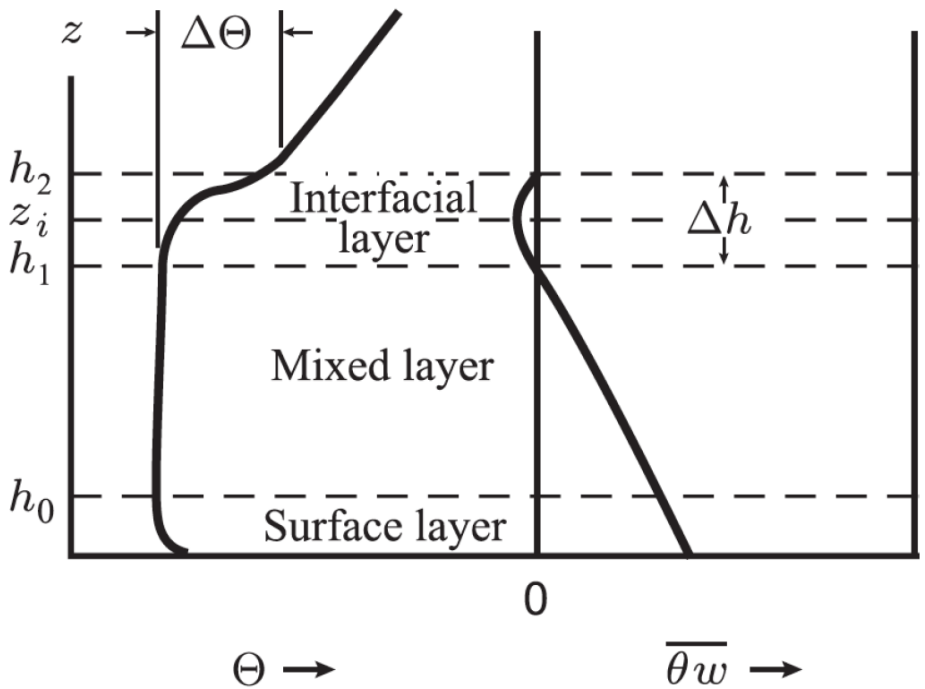
It is useful to further divide the PBL into sublayers:

- Surface layer, lowest part of the PBL, where turbulent fluxes can be shown to be approximately constant.
- An intermediate layer, where the characteristics depend strongly on the static stability.
- An interfacial layer, marking the boundary between the turbulent PBL and the free, laminar atmosphere above.

PBL structure

The *turbulent structure* of the PBL depend mainly on two components: surface static stability and vertical wind shear.

Example of unstable/convective boundary layer.



Parameterization of turbulence in NWP

- First order parameterization methods assume diagnostic relations with known prognostic variables.
- Second order methods use prognostic equations for (some or all) turbulent fluxes and parameterize third order terms.
  This can be computationally demanding and may account for a substantial part of the of the computation time in a NWP model.
- In both cases closure constants must be determined by fitting to observed/simulated data.

Mixing length hypothesis and K-closure

Idea behind Ludwig Prandtl's mixing length hypothesis:

- Consider an air parcel vertically displaced by Δz. Its temperature deviation from the surroundings is then

$$\theta = \Theta(z) - \Theta(z + \Delta z) \approx -\Delta z \frac{\partial \Theta}{\partial z}$$

- Assume that an air parcel moved by the turbulent wind field u_i will carry its properties some characteristic distance, d, before it mixes with the surrounding air.
- The magnitude of the temperature deviation, θ, must then related to the characteristic length scale, d, such that

$$\theta \sim -d \frac{\partial \Theta}{\partial z} \quad \Rightarrow \quad \overline{w\theta} \sim -K \frac{\partial \Theta}{\partial z}, \quad \text{where} \quad K = \overline{wd}$$

- Note that only the vertical component is considered. Horizontal gradients are negligible and can be ignored.

MYNN2.5 parameterization scheme

The Mellor-Yamanda-Nakanishi-Niino level 2.5 parameterization scheme is a second order scheme, using a prognostic equation for TKE:

$$\frac{\partial q^2}{\partial t} = -\frac{\partial}{\partial z}\left(\overline{wq^2} + 2\frac{\overline{wp}}{\rho_0}\right) - 2\left(\overline{wu}\frac{\partial U}{\partial z} + \overline{wv}\frac{\partial V}{\partial z}\right) + 2\frac{g}{\Theta_0}\overline{w\theta_v} - 2\varepsilon, \quad (1)$$

where $q^2 = \sum_i \overline{u_i u_i} = 2TKE$.

The remaining covariance terms are parameterized as follows:

$$\overline{wu} = -K_m\frac{\partial U}{\partial z}, \qquad \overline{w\theta_l} = -K_h\frac{\partial \Theta_l}{\partial z}, \qquad \varepsilon = \frac{q^3}{B_1 L},$$

$$\overline{wv} = -K_m\frac{\partial V}{\partial z}, \qquad \overline{wq_w} = -K_h\frac{\partial Q_w}{\partial z},$$

$$\overline{wq^2} + 2\frac{\overline{wp}}{\rho_0} = -3K_m\frac{\partial q^2}{\partial z}, \qquad \overline{w\theta_v} = \beta_\theta\overline{w\theta_l} + \beta_q\overline{wq_w}, \quad (2)$$

where q_w is the total water content and θ_l is the liquid water temperature.

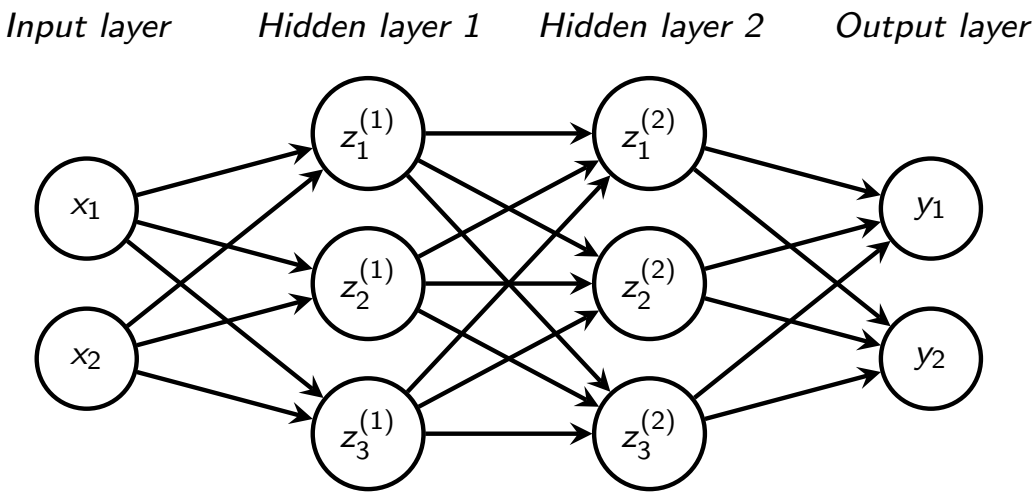
Machine learning models for regression

- One advantage of using a machine learning model is that we do not need to assume *how* the variables depend on the relevant inputs.
- Disadvantage: many model parameters may require a large dataset to avoid overfitting.
- Training may be slow, but evaluation is typically fast and can easily be parallelized.
- Artificial neural networks, *ANN*, is a class of machine learning models suitable for both classification and regression problems.

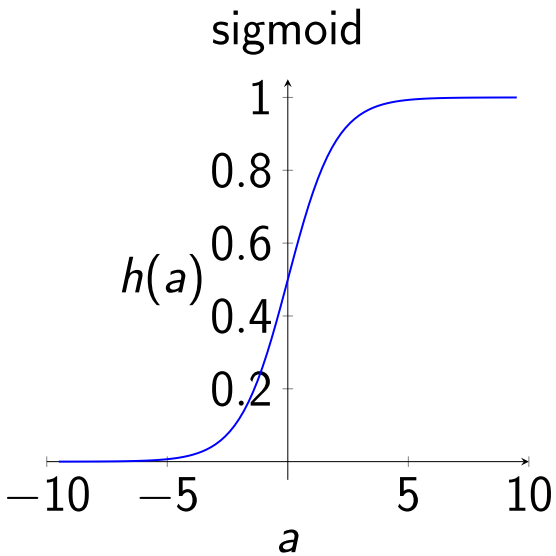
Feed forward neural networks

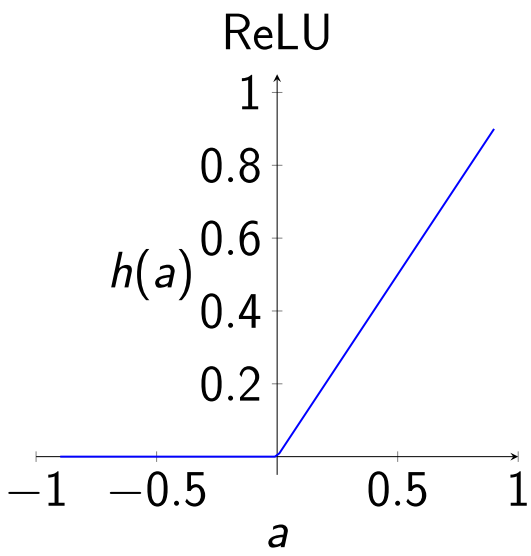
Given input vector x_i, the neural network prediction $\hat{y}_j$ can be written as the recurrence relation

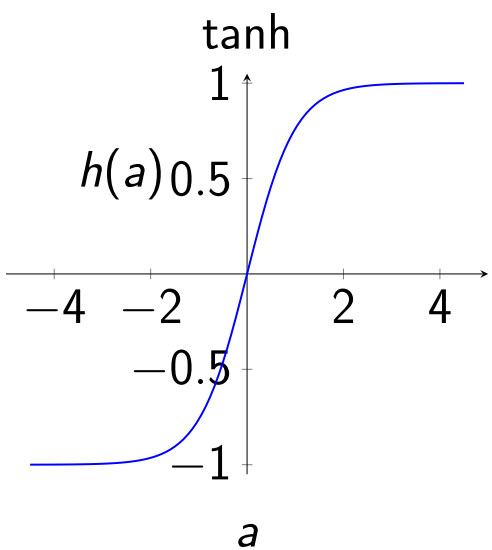
$$\begin{aligned}
\hat{y}_j &= w_{ji}^{(N)} z_i^{(N-1)} + b_j^{(N)}, \\
z_j^{(n)} &= h\left(a_j^{(n)}\right) \quad \text{for} \quad n = 1, 2, ..., N-1\ , \quad \text{where} \\
a_j^{(n)} &= w_{ji}^{(n)} z_i^{(n-1)} + b_j^{(n)}, \\
z_i^{(0)} &= x_i .
\end{aligned}$$

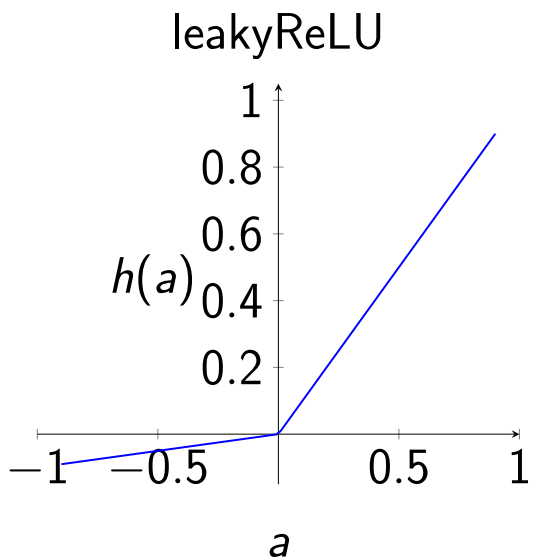


Role of activation function









Training neural networks

Training consist of determining the optimal model parameters (weights $w_{ji}^{(n)}$ and biases $b_j^{(n)}$) - minimize the error on predictions.

Equally important is optimizing the hyperparameters:

- Loss function (mean squared error, mean absolute error, etc.)
- Optimization algorithm (e.g. stochastic gradient descent)
- Batch size
- Learning rate (step size in gradient descent)
- Activation function
- Network size (number of layers and nodes per layer)
- Type of input/output normalization/scaling

Illustration of learning curves

It is helpful to monitor "learning curves" during training to avoid overfitting.

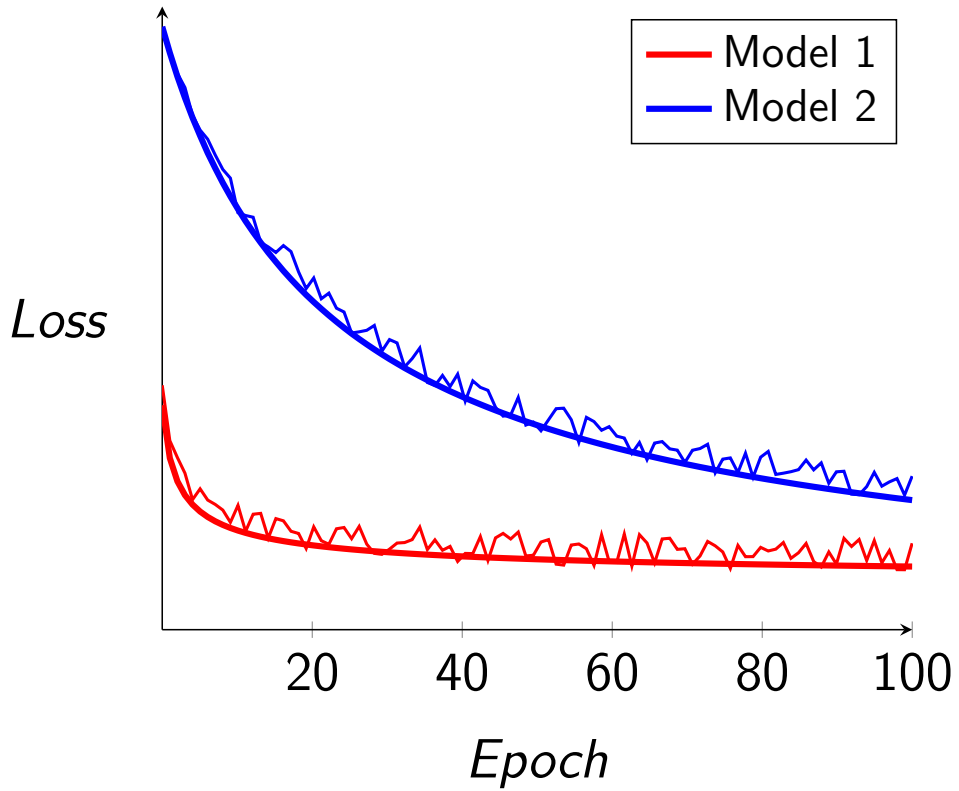
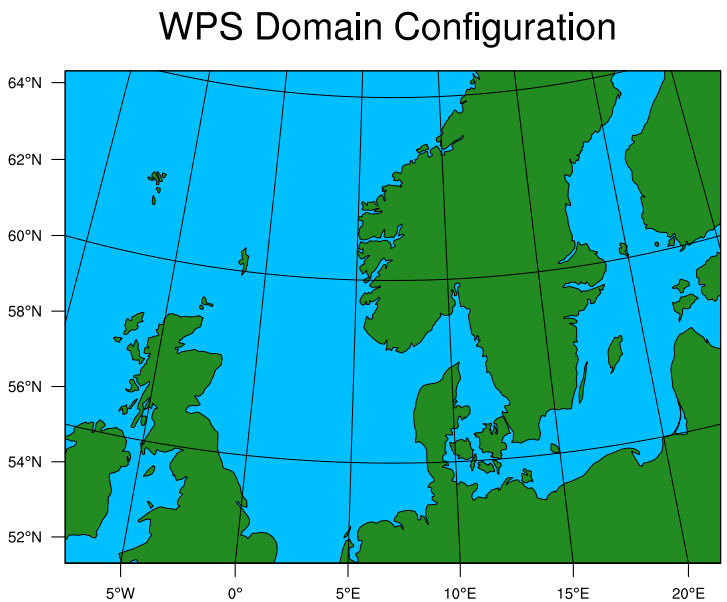


Figure: Sketch of learning curves for two different models. The thick solid lines show the loss on the training data for each of the two models, while the thin solid lines show the loss on the validation data.

Construction of dataset

- Six 24-hour (+6 hours spin-up) simulations with the Weather Research and Forecast model, *WRF*
- 10x10 km horizontal resolution
- 41 η-levels, lowest level at $\sim$ 10 m and 14 levels within the lowest km
- Model output every 60 minutes. Each time, 500 randomly selected air columns are added to the training dataset and the validation set



Model construction

Recall, the MYNN2.5 parameterization:

$$\overline{wu} = -K_m \frac{\partial U}{\partial z}, \qquad \overline{w\theta_l} = -K_h \frac{\partial \Theta_l}{\partial z}, \qquad \varepsilon = \frac{q^3}{B_1 L},$$

$$\overline{wv} = -K_m \frac{\partial V}{\partial z}, \qquad \overline{wq_w} = -K_h \frac{\partial Q_w}{\partial z},$$

$$\overline{wq^2} + 2\frac{\overline{wp}}{\rho_0} = -3K_m \frac{\partial q^2}{\partial z}, \qquad \overline{w\theta_v} = \beta_\theta \overline{w\theta_l} + \beta_q \overline{wq_w},$$

Considerations:

- Should the model compute the turbulent quantities layer by layer? Or column by column?
- Should the model compute the tendencies (gradients of fluxes)? The fluxes? Or the Diffusivities?
- Should the prognostic *TKE* equation be kept? Or should the model predict the *TKE* as well? Or should *TKE* just be left out?

Model construction

Recall, the MYNN2.5 parameterization:

$$\overline{wu} = -K_m \frac{\partial U}{\partial z}, \qquad \overline{w\theta_l} = -K_h \frac{\partial \Theta_l}{\partial z}, \qquad \varepsilon = \frac{q^3}{B_1 L},$$

$$\overline{wv} = -K_m \frac{\partial V}{\partial z}, \qquad \overline{wq_w} = -K_h \frac{\partial Q_w}{\partial z},$$

$$\overline{wq^2} + 2\frac{\overline{wp}}{\rho_0} = -3K_m \frac{\partial q^2}{\partial z}, \qquad \overline{w\theta_v} = \beta_\theta \overline{w\theta_l} + \beta_q \overline{wq_w},$$

Do not use TKE as prognostic variable.
Predict fluxes directly

$$\overline{wu}, \overline{w\theta_l}, \overline{wv}, \overline{wq_w}, \overline{wq^2} = f(\textit{input arguments})$$

Model construction

Recall, the MYNN2.5 parameterization:

$$\overline{wu} = -K_m \frac{\partial U}{\partial z}, \qquad \overline{w\theta_l} = -K_h \frac{\partial \Theta_l}{\partial z}, \qquad \varepsilon = \frac{q^3}{B_1 L},$$

$$\overline{wv} = -K_m \frac{\partial V}{\partial z}, \qquad \overline{wq_w} = -K_h \frac{\partial Q_w}{\partial z},$$

$$\overline{wq^2} + 2\frac{\overline{wp}}{\rho_0} = -3K_m \frac{\partial q^2}{\partial z}, \qquad \overline{w\theta_v} = \beta_\theta \overline{w\theta_l} + \beta_q \overline{wq_w},$$

Keep *TKE* as prognostic variable.
Predict fluxes directly

$$\overline{wu}, \overline{w\theta_l}, \overline{wv}, \overline{wq_w}, \overline{wq^2}, \overline{wq^2} + 2\frac{\overline{wp}}{\rho_0}, \varepsilon = f(\textit{input arguments})$$

Model construction

Recall, the MYNN2.5 parameterization:

$$\overline{wu} = -K_m \frac{\partial U}{\partial z}, \qquad \overline{w\theta_l} = -K_h \frac{\partial \Theta_l}{\partial z}, \qquad \varepsilon = \frac{q^3}{B_1 L},$$

$$\overline{wv} = -K_m \frac{\partial V}{\partial z}, \qquad \overline{wq_w} = -K_h \frac{\partial Q_w}{\partial z},$$

$$\overline{wq^2} + 2\frac{\overline{wp}}{\rho_0} = -3K_m \frac{\partial q^2}{\partial z}, \qquad \overline{w\theta_v} = \beta_\theta \overline{w\theta_l} + \beta_q \overline{wq_w},$$

Keep *TKE* as prognostic variable.
Predict diffusivities

$$K_m, K_h, L, \overline{w\theta_v} = f(\textit{input arguments})$$

Combine with traditional method for solving *TKE* equation and the diffusion problem.

Determine model input

- Divide dataset based on local static stability - allow for differences in variable dependence.
- Assume a set of "base variables".
- Iteratively add potentially relevant variables as input to the neural network.
- Base variables:

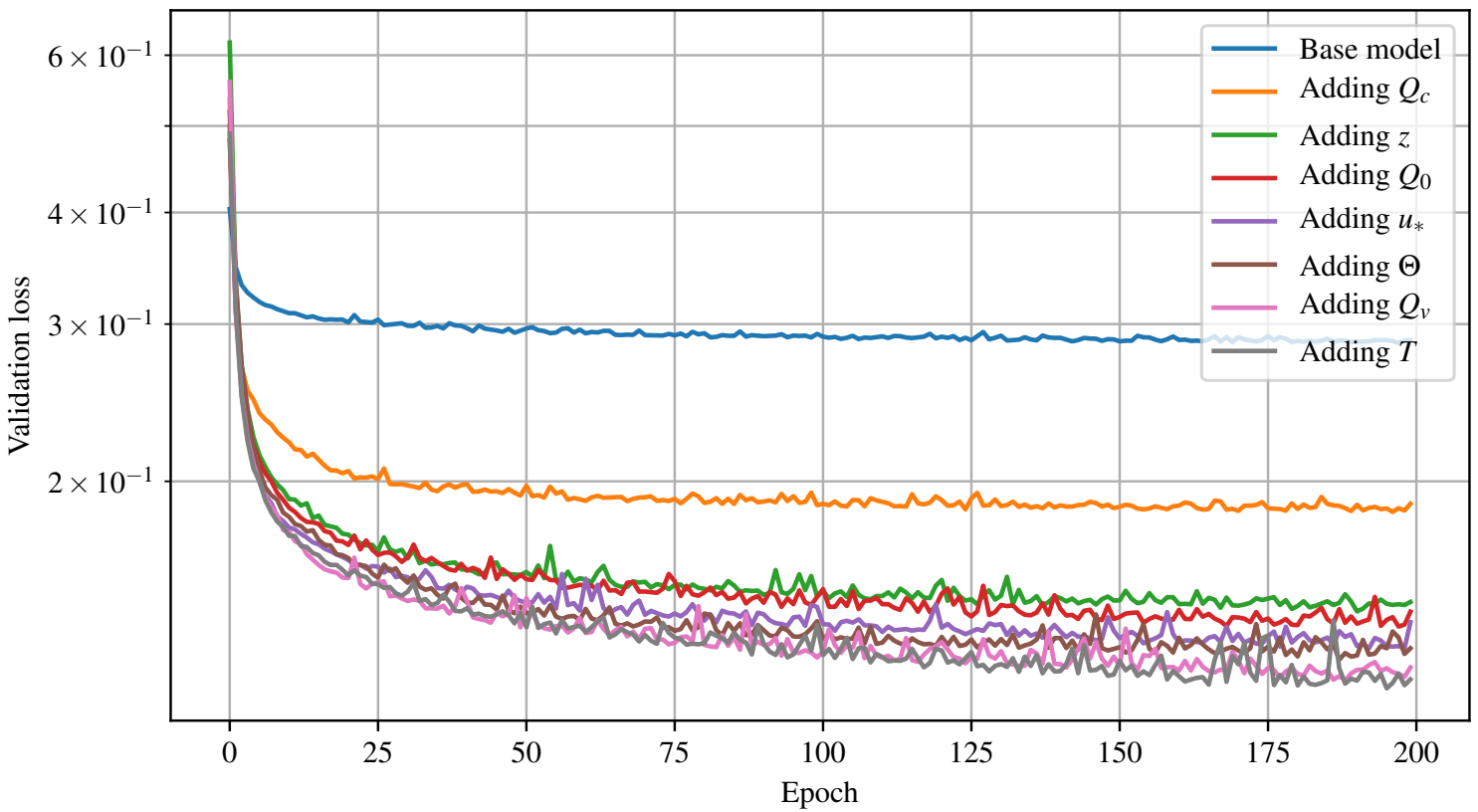
$$q = \sqrt{\overline{uu} + \overline{vv} + \overline{ww}}, \quad B = -\frac{g}{\Theta_0}\frac{\partial \Theta_v}{\partial z}, \quad S = \left(\frac{\partial U}{\partial z}\right)^2 + \left(\frac{\partial V}{\partial z}\right)^2$$

- Additional variables to be tested:

$$z, Q_0, u_*, \Theta, T, Q_v, \text{ and } Q_c.$$

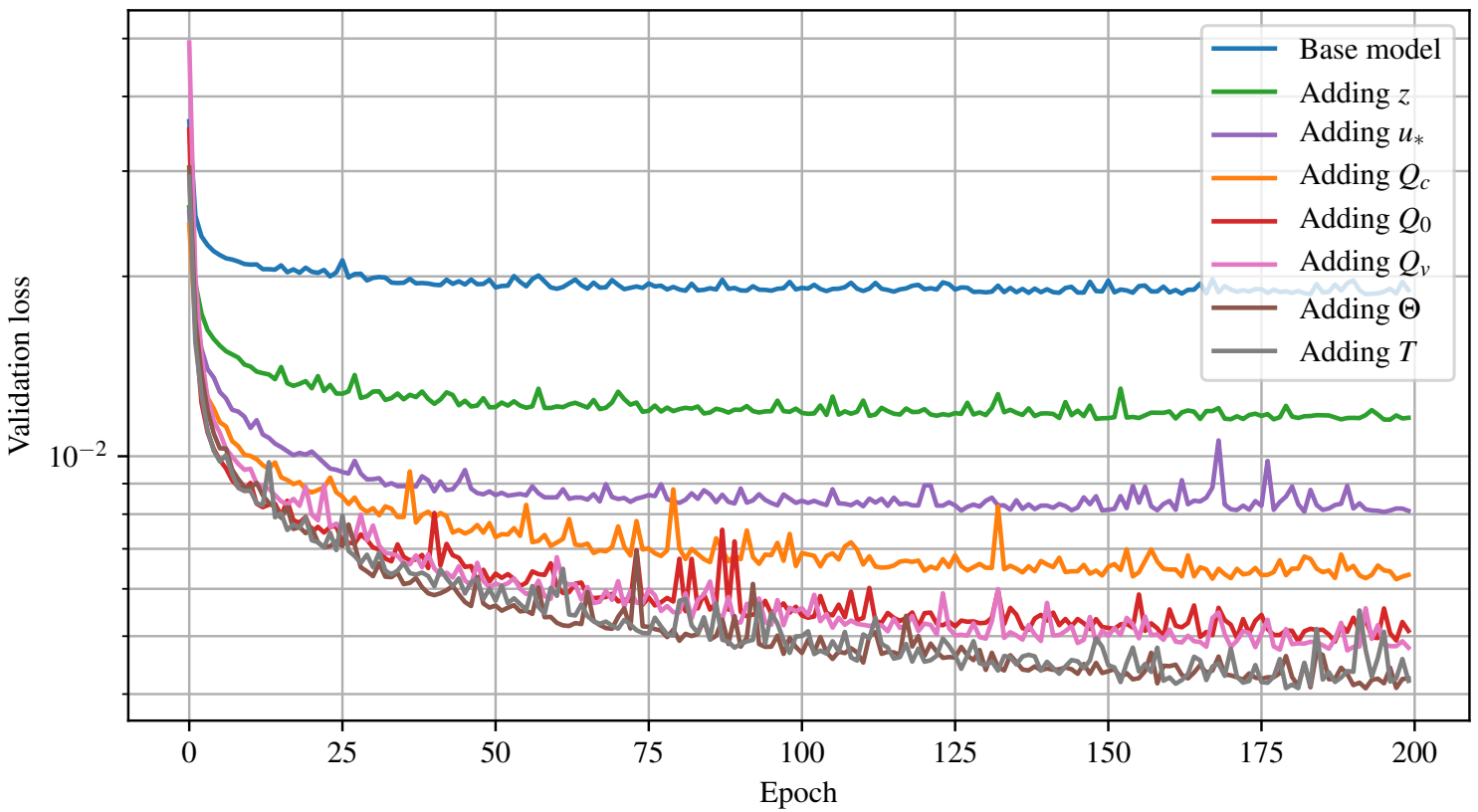
Determine model input

For statically stable model levels (shows loss on validation set)



Determine model input

For statically unstable model levels (shows loss on validation set)



Comment on input variables

Based on the iterative method applied, one could conclude that all variables except T.

However, it turned out to give a better result to only use following variables as input:

$$q, B, S, z, Q_0, u_*$$

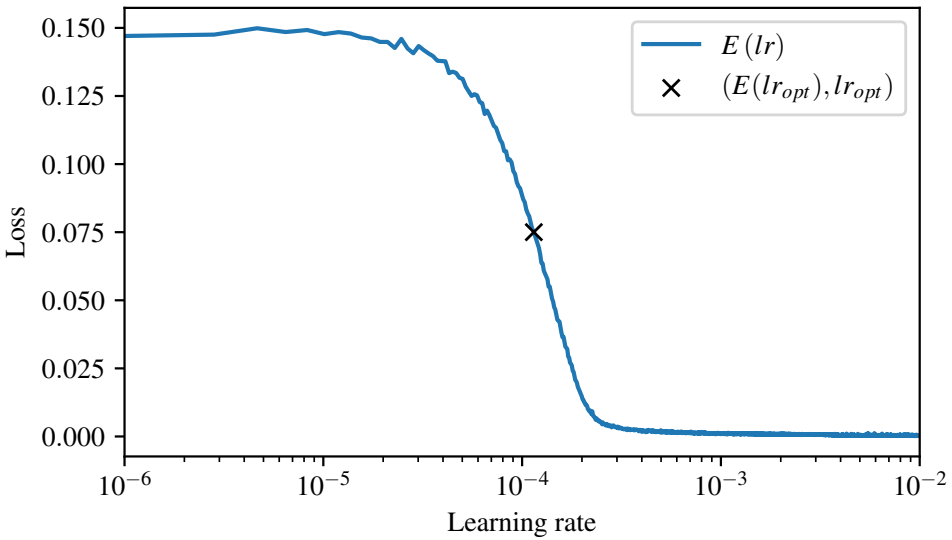
The reason was most likely overfitting, because the validation dataset was not sufficiently independent (sampled from the same simulations).

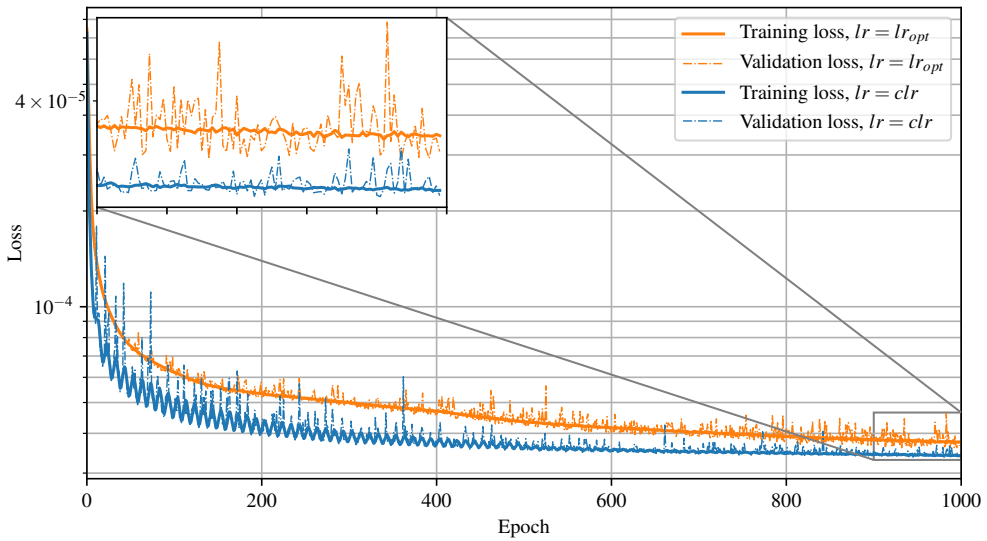
Model optimization

- All models are trained using Tensorflow and Keras in Python.
- The model parameters are optimized using the stochastic gradient descent based optimization algorithm Adaptive Moment Estimation, *Adam*.
- Learning rate is determined before all trainings start by using a linear "learning rate scanner". Further, a cyclic learning rate schedule is used.
- Additional hyperparameters are optimized using grid search in different "steps".

Model optimization

Illustration of learning rate optimization.





Model optimization

Hyperparameter optimization, first part:

Table: First column is the model number. Second column tells us, how the data is categorized: *None* means one model is used for all samples. *Stability* means that two different models are used for stable/unstable samples. *Stability and TKE* means stable samples are further divided into samples with/without TKE. Third column shows the data processing, and the fourth column shows the loss function used. *mse* is mean squared error, while *mae* is mean absolute error.

	Data categorization	**Pre- and postprocessing**	**Loss function**
Model 1	None	Linear scaling	*mse* on physical values
Model 2	Stability	Linear scaling	*mse* on physical values
Model 3	Stability	Logarithmic scaling	*mse* on log-scaled values
Model 4	Stability and TKE	Logarithmic scaling	*mse* on log-scaled values
Model 5	Stability and TKE	Logarithmic scaling	*mse* on physical values
Model 6	Stability and TKE	Logarithmic scaling	*mae* on physical values

Model optimization

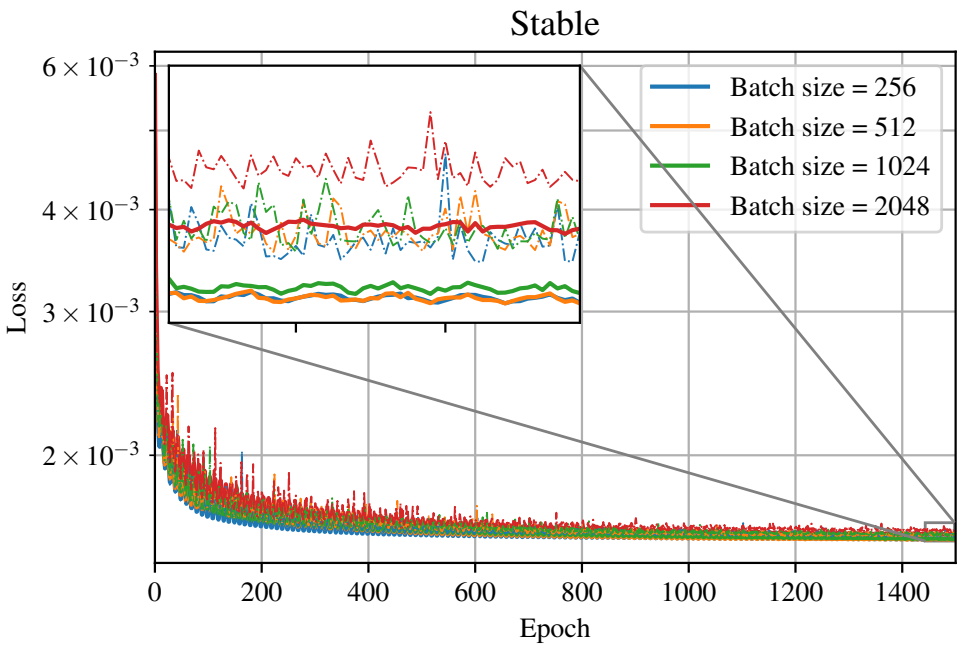
Hyperparameter optimization, first part (result):

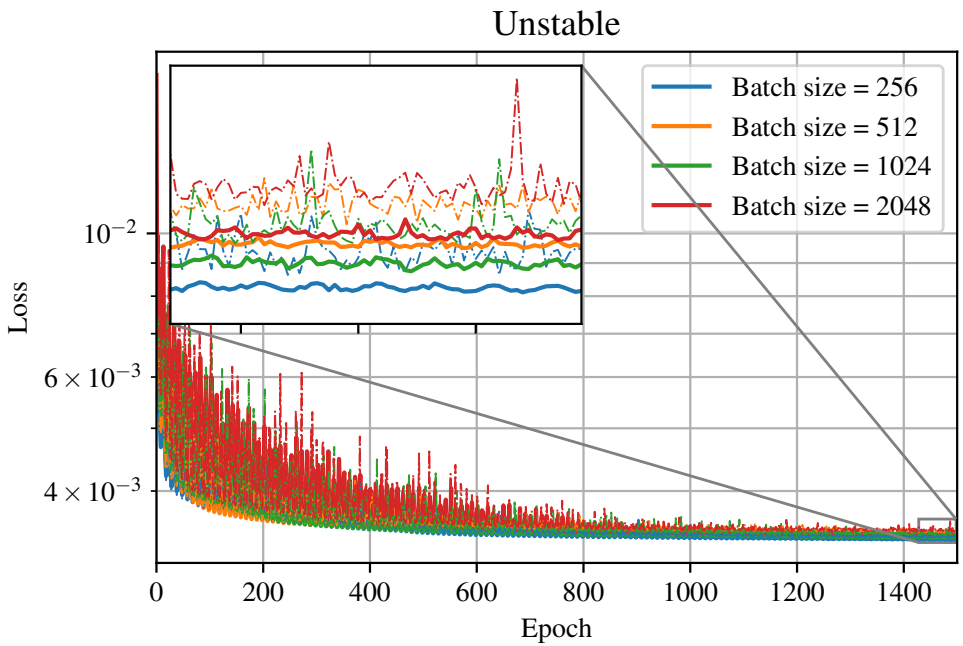
Table: Performance of the models tested in the first step of the model optimization. The model specifications are explained in Table 1. For each variable and each error measure, the best performance is highlighted with red color, and the worst is highlighted with blue.

	Stat	K_h	K_m	L	B_p	$-K_h B$	$K_m\sqrt{S}$	q^3/L
Model 1	l_1	0.2713	0.2861	0.1261	0.2835	0.4481	0.9068	0.2849
	l_2	0.2986	0.3154	0.1598	0.2303	1.011	5.289	4.877
	r	0.9508	0.9438	0.9841	0.9733	0.6927	0.1277	0.3667
Model 2	l_1	0.2275	0.3206	0.1120	0.2261	0.5495	1.218	0.2263
	l_2	0.2250	0.2762	0.1476	0.1915	0.6040	8.798	0.7186
	r	0.9727	0.9580	0.9864	0.9820	0.8598	0.2382	0.8736
Model 3	l_1	0.06222	0.04390	0.1168	0.05586	0.07348	0.02881	0.07455
	l_2	0.1188	0.1136	0.1843	0.1082	0.1712	0.06836	0.1198
	r	0.9924	0.9930	0.9793	0.9943	0.9853	0.9978	0.9927
Model 4	l_1	0.05847	0.04058	0.1124	0.04618	0.06845	0.02351	0.05988
	l_2	0.1096	0.08420	0.1555	0.06163	0.1595	0.03453	0.1123
	r	0.9935	0.9962	0.9850	0.9981	0.9872	0.9994	0.9939
Model 5	l_1	0.08943	0.07602	0.1212	0.09739	0.1960	0.4034	0.1119
	l_2	0.1191	0.09794	0.1545	0.1364	0.2993	0.7136	0.1742
	r	0.9924	0.9948	0.9851	0.9907	0.9627	0.9837	0.9844
Model 6	l_1	0.04779	0.03089	0.09572	0.03554	0.05976	0.01854	0.05179
	l_2	0.1053	0.06758	0.1360	0.05390	0.1589	0.01868	0.09625
	r	0.9941	0.9975	0.9887	0.9985	0.9873	0.9999	0.9954

Model optimization

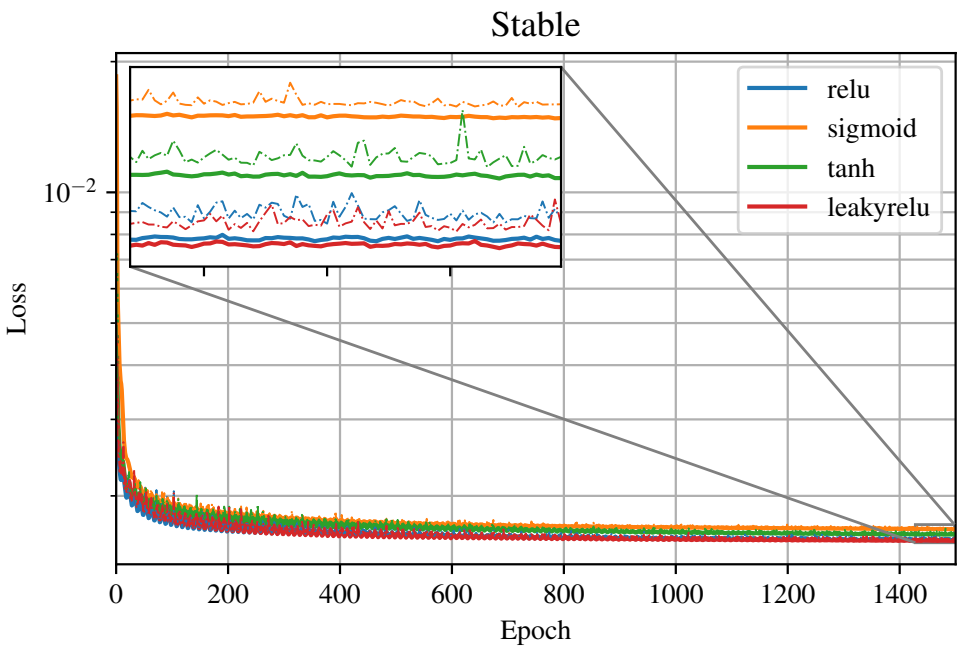
Batch size was found to have little or no impact

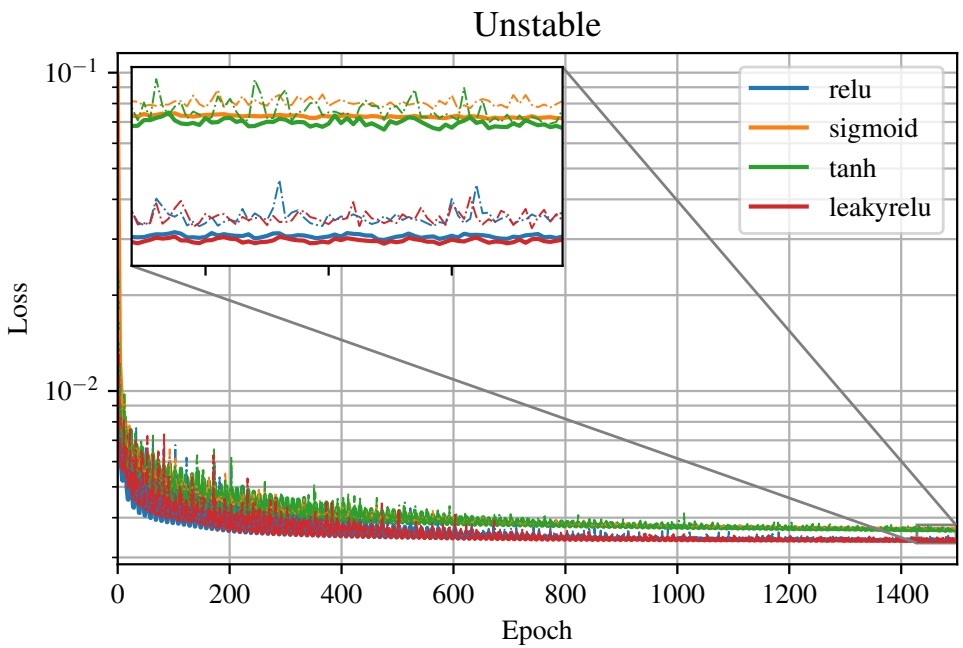




Model optimization

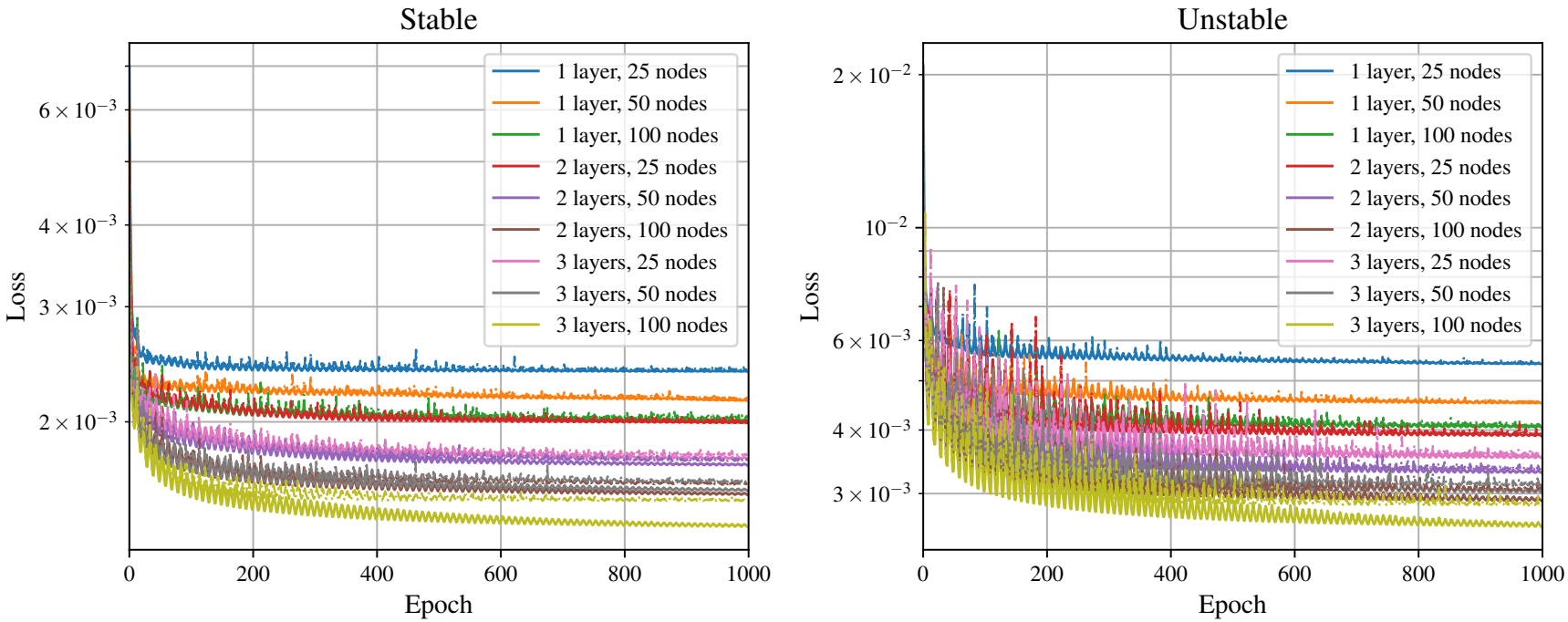
Activation functions were found to have little impact: ReLU and leakyReLU seem to work slightly better than tanh and sigmoid.





Model optimization

Optimizing the network size.

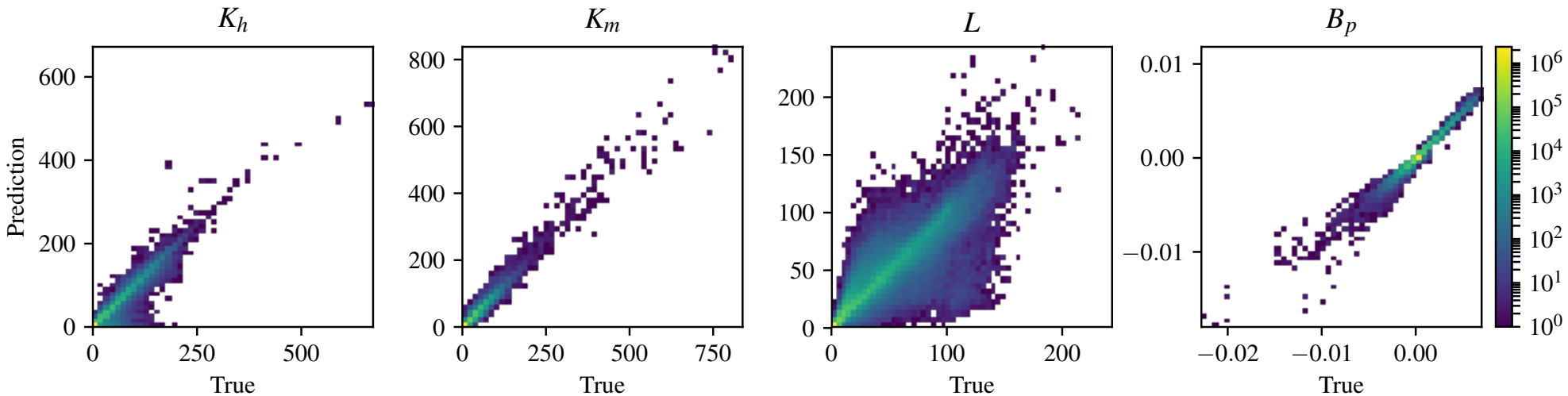


Due to lack of independence between training/validation, the optimal network couldn't be estimated.

The chosen "optimal" network sizes were:
1 layer and 50 nodes per layer (stable samples)
2 layers and 25 nodes per layer (unstable samples)

Predictions of "optimal" model

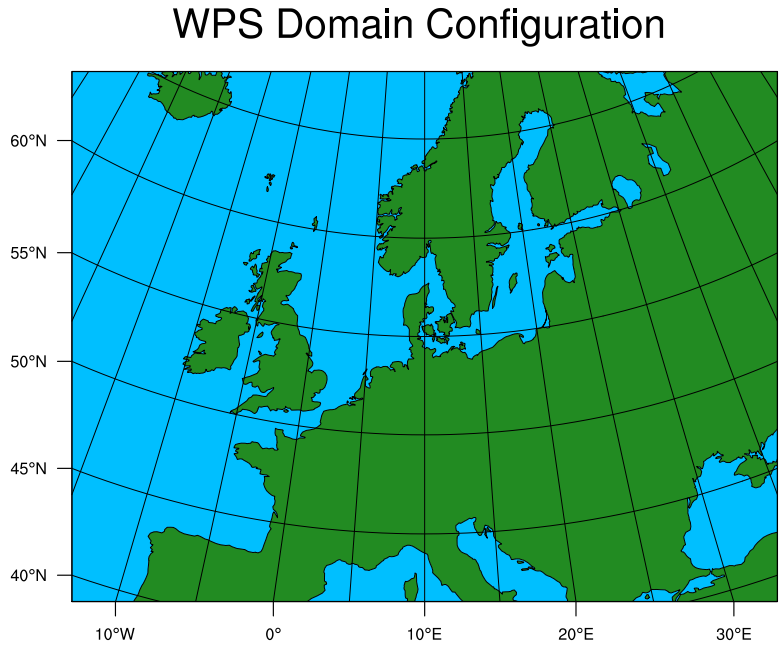
Plots of ANN predictions as function of true values.



Implementation and test

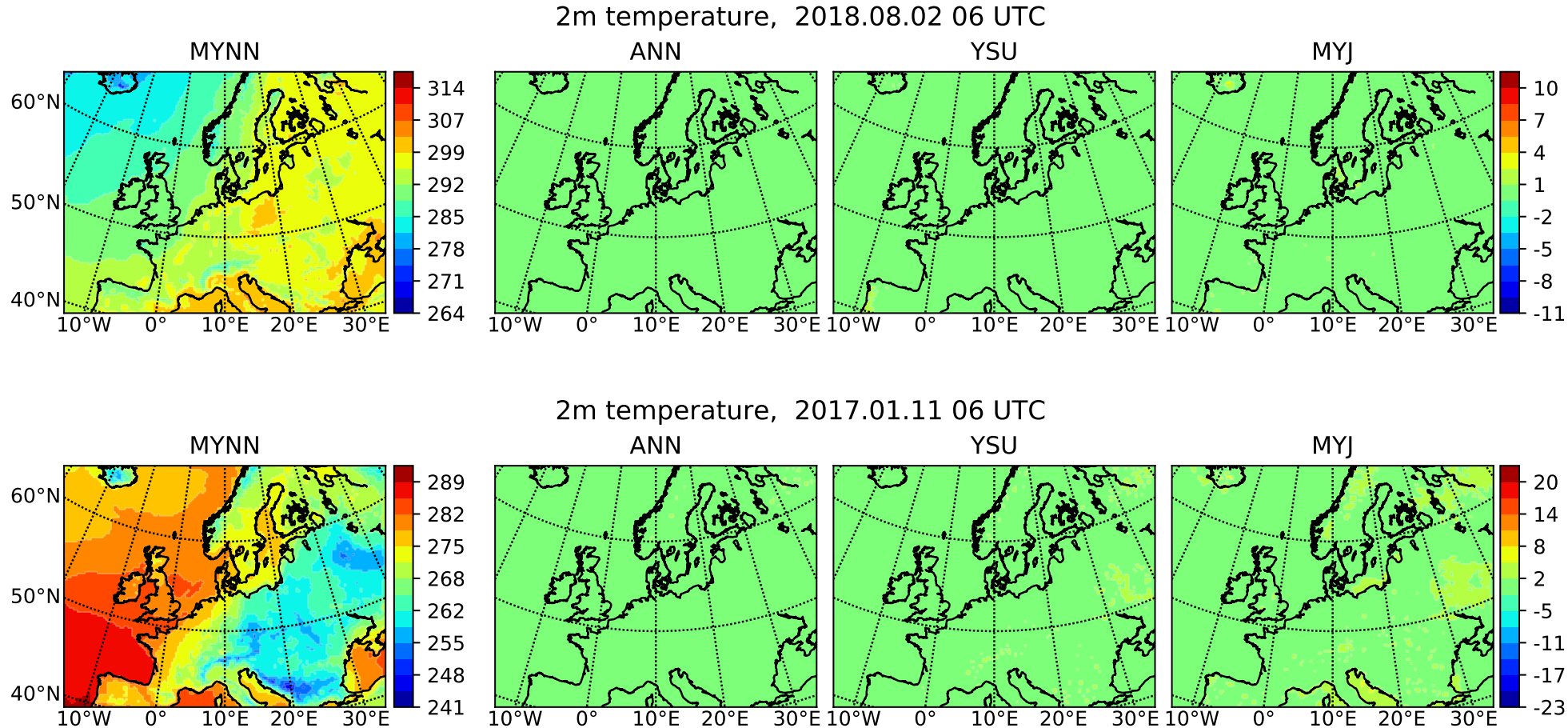
Methodology:

- two 72 hour simulations made with both the MYNN2.5 and the ANN schemes (summer/winter)
- horizontal and vertical resolution same as for training data
- larger domain than for training
- control run with MYNN2.5 scheme
- for comparison, runs are made with two additional PBL schemes: YSU and MYJ



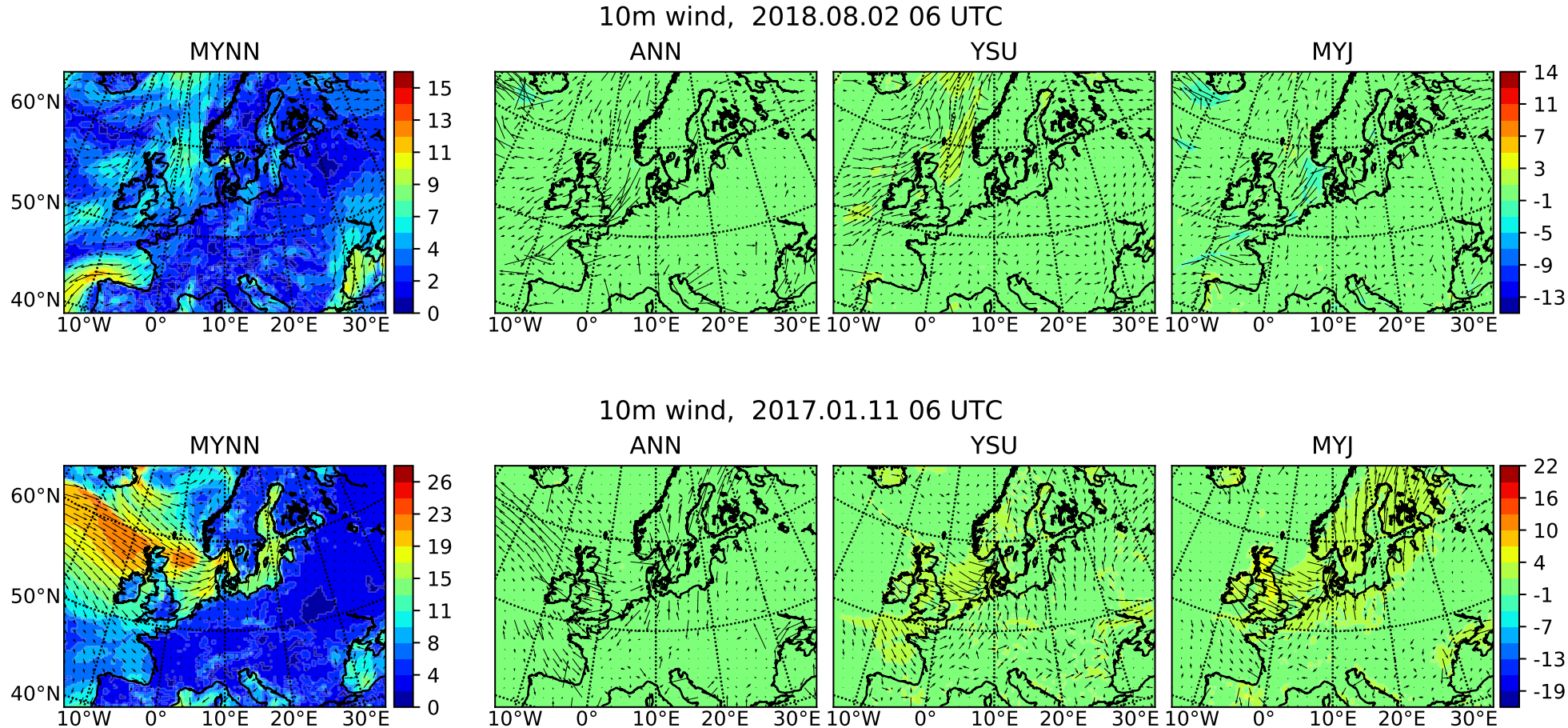
Results example

2-meter temperature (summer and winter)

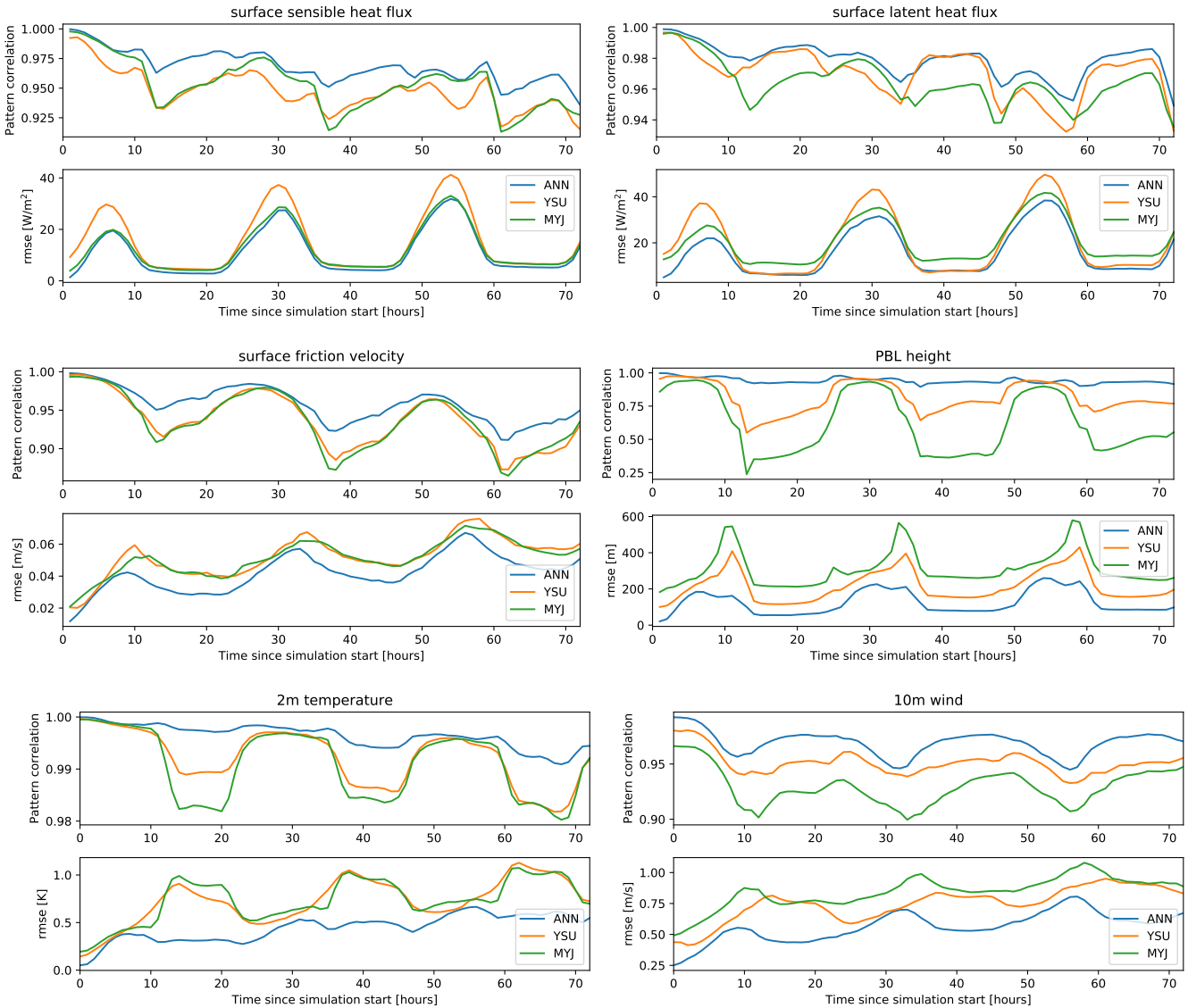


Results example

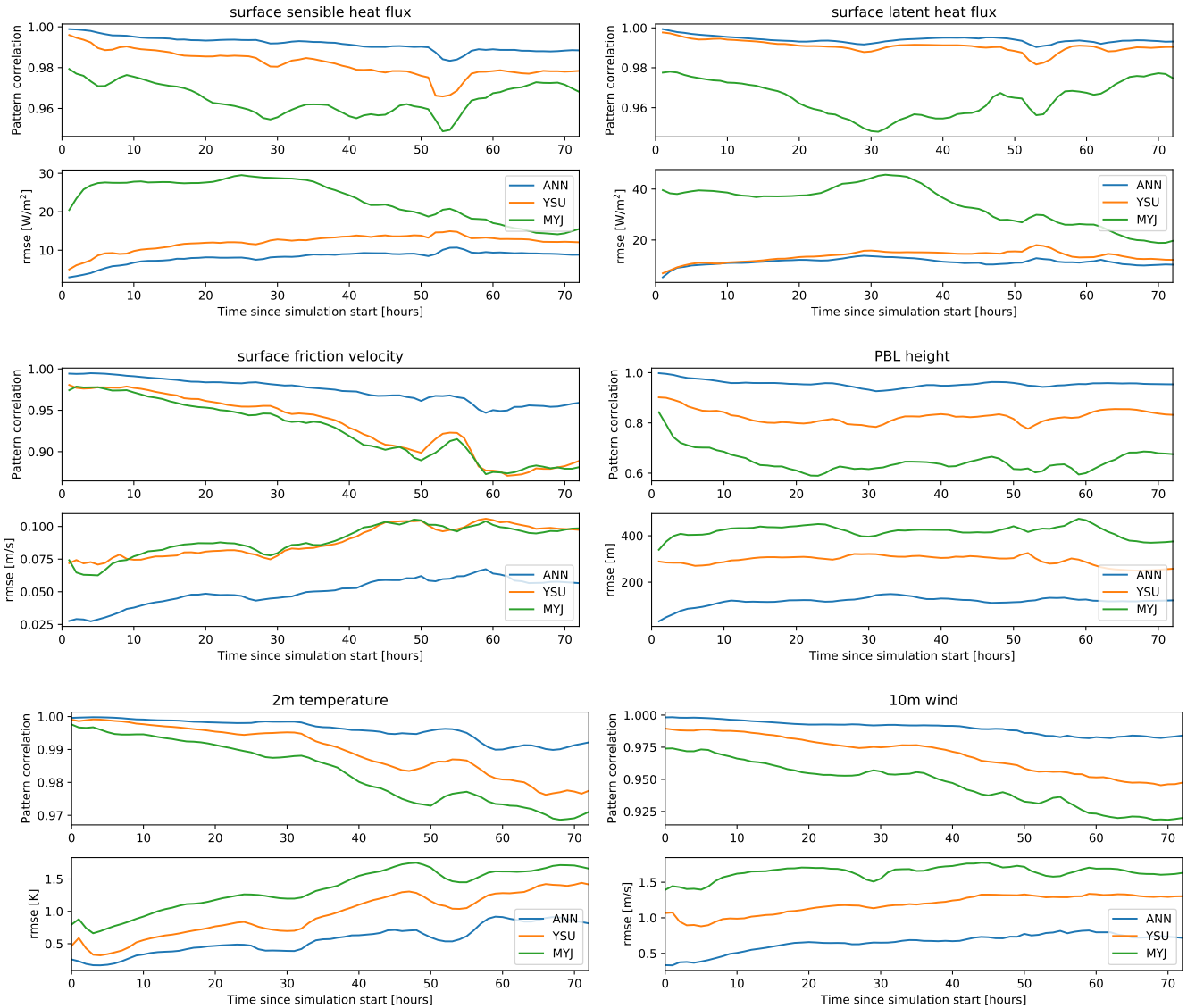
10-meter wind (summer and winter)



Results, error vs time (summer)



Results, error vs time (winter)



Results, efficiency

Scheme	Time spend on the PBL scheme	% of total run time
MYNN	19.50 s	16.9%
ANN (matmul)	10.89 s	10.1%
ANN (sgemm)	9.49 s	9.1%
YSU	3.61 s	3.7%
MYJ	6.44 s	6.1%

Scheme	Subroutine	Time spend on the PBL scheme	% of total run time
MYNN	Computing K_m, K_h and L	5.53 s	4.8%
	Condensation scheme	4.31 s	3.7%
	Computing tendencies	4.09 s	3.6%
	Solving TKE equation	0.70 s	0.6%
	Other	4.87 s	4.2%
ANN (matmul)	Neural network prediction	1.58 s	1.5%
	Computing tendencies	3.77 s	3.5%
	Solving TKE equation	0.55 s	0.5%
	Other	4.99 s	4.6%
ANN (sgemm)	Neural network prediction	0.85 s	0.8%
	Computing tendencies	3.72 s	3.6%
	Solving TKE equation	0.64 s	0.6%
	Other	4.28 s	4.1%

Conclusions

- It is possible to create an accurate and robust turbulence closure model using neural networks.
- Demonstration of important results related to scaling/normalization of inputs and outputs.
- Neural network based turbulence parameterizations has potential of being an efficient alternative to expensive second order PBL schemes while retaining the second order accuracy.

Outlook

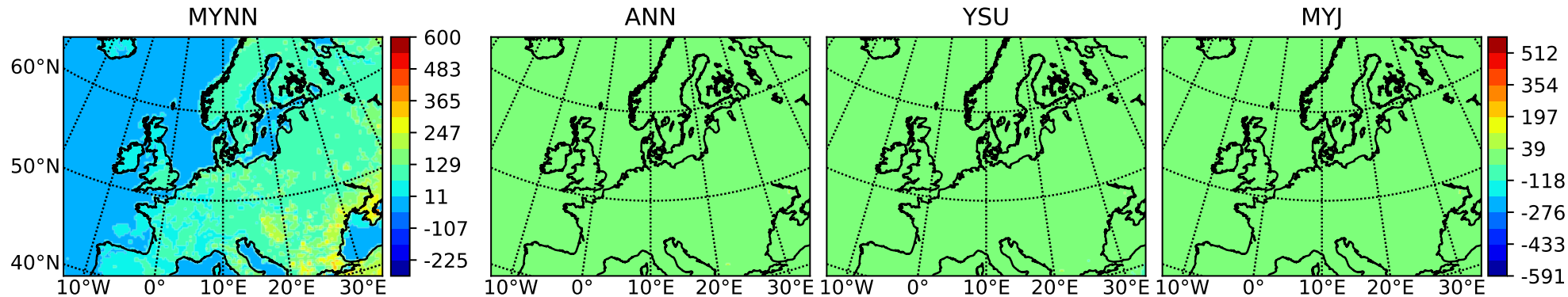
- Neural network optimization and training should be repeated with a new and better dataset - ensure validation set is independent.
- Perhaps test other types of machine learning models, e.g. random forests or boosted decision trees.
- Train a machine learning model on a data from high resolution models, e.g. LES.

Thanks for listening!

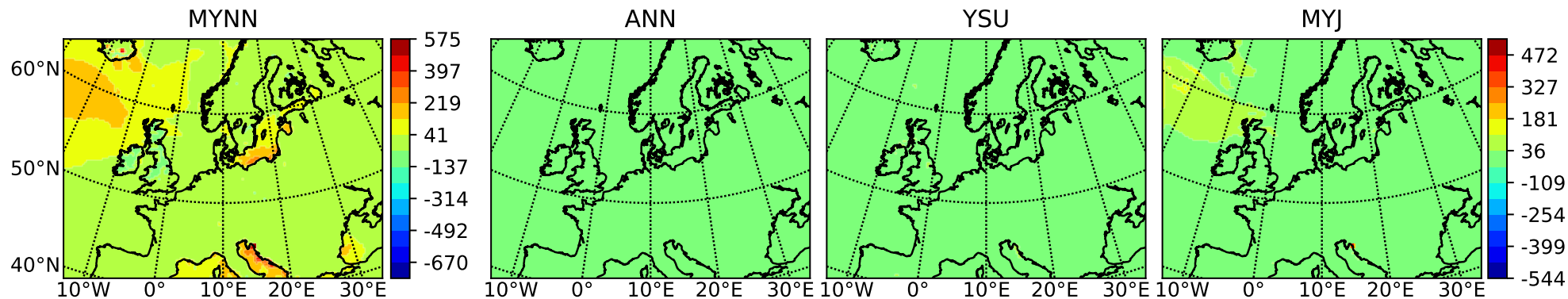
Results example (extra slides)

Surface sensible heat flux (summer and winter)

surface sensible heat flux, 2018.08.02 07 UTC



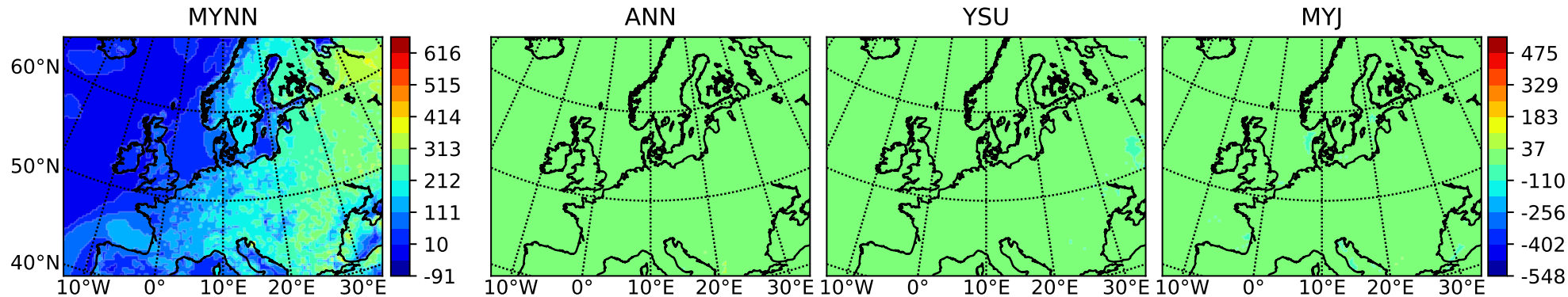
surface sensible heat flux, 2017.01.11 07 UTC



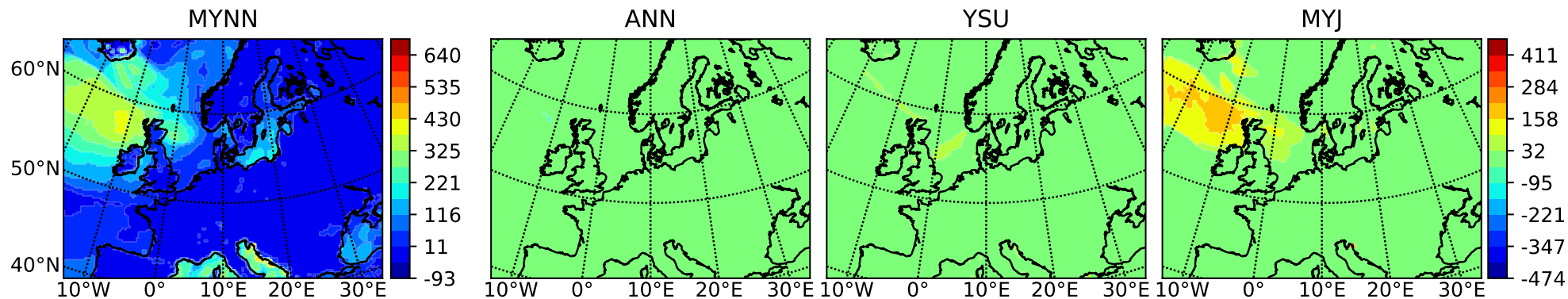
Results example (extra slides)

Surface latent heat flux (summer and winter)

surface latent heat flux, 2018.08.02 07 UTC

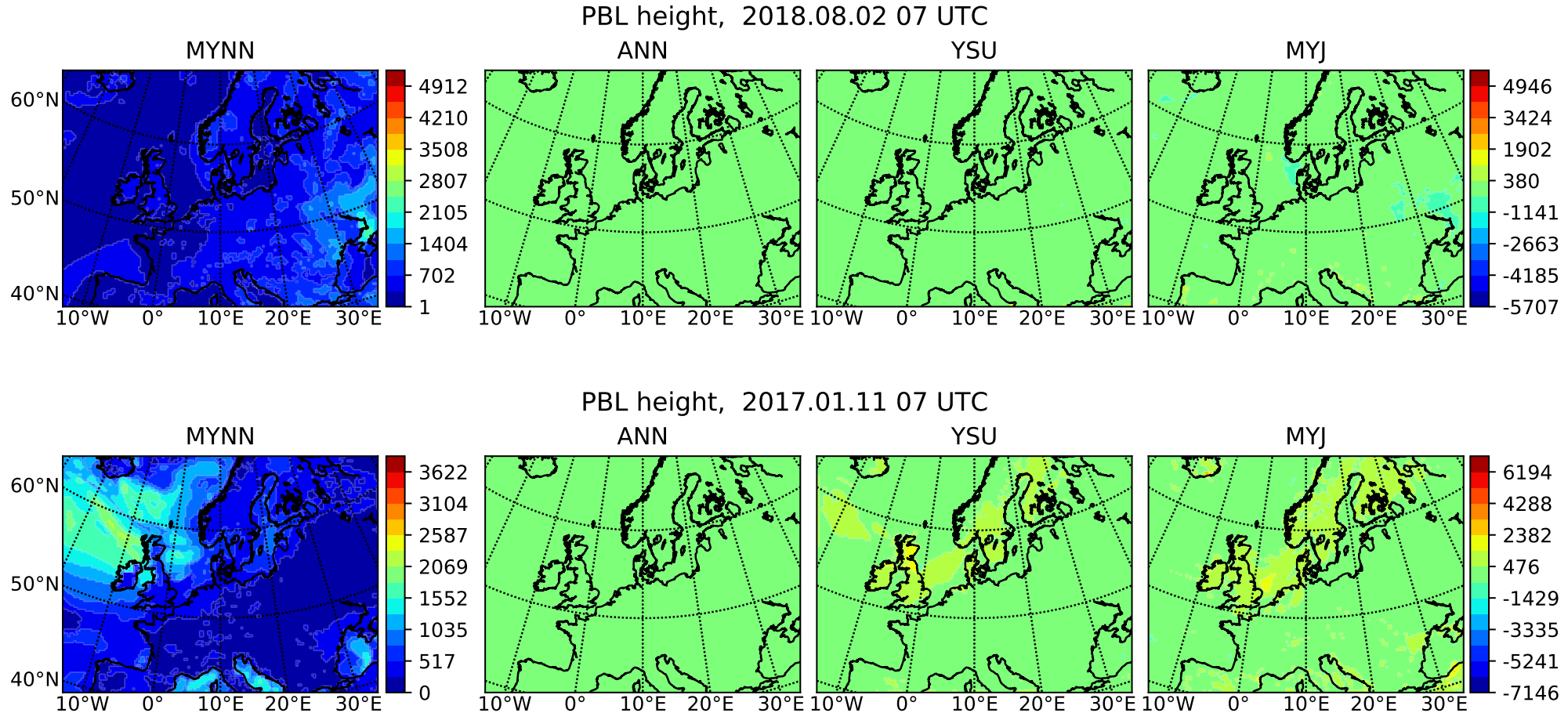


surface latent heat flux, 2017.01.11 07 UTC



Results example (extra slides)

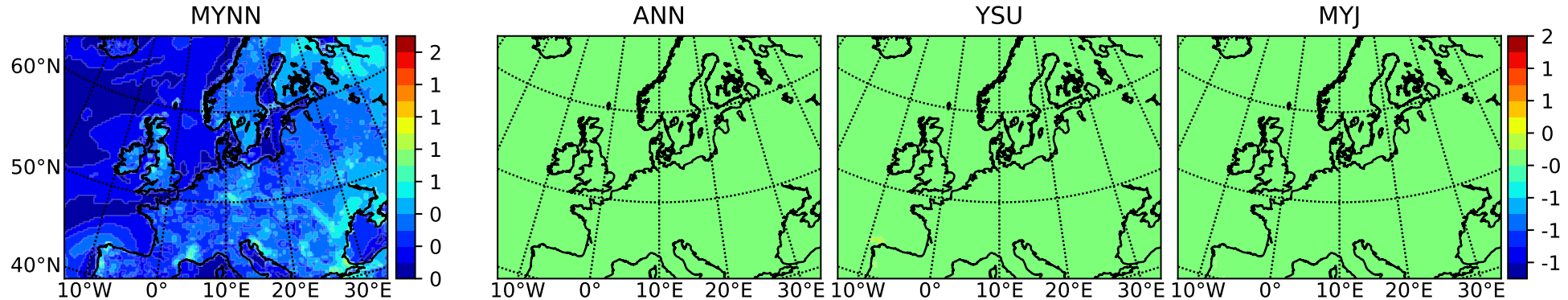
PBLH (summer and winter)



Results example (extra slides)

U_* (summer and winter)

surface friction velocity, 2018.08.02 07 UTC



surface friction velocity, 2017.01.11 07 UTC