Co-ordinated by

MAchinE Learning for Scalable meTeoROlogy and climate

MAELSTROM

Report on tests with a tangent-linear and adjoint version of ML emulators with 4DVar

Matthew Chantry, Ana Prieto Nemesio, Peter Dueben

www.maelstrom-eurohpc.eu

MAELSTROM

D1.5 Report on tests with a tangent-linear and adjoint version of ML emulators with 4DVar

Author(s):	Matthew Chantry (ECMWF)
	Ana Prieto Nemesio (ECMWF)
	Peter Dueben (ECMWF)

Dissemination Level:	Public
Date:	25/03/2024
Version:	0.2
Contractual Delivery Date:	31/03/2024
Work Package/ Task:	WP1/ 1.5
Document Owner:	ECMWF
Contributors:	Organisations
Status:	Draft/ **for Review**/ Final

MAELSTROM

Machine Learning for Scalable Meteorology and Climate

Research and Innovation Action (RIA)
H2020-JTI-EuroHPC-2019-1: Towards Extreme Scale Technologies and Applications

Project Coordinator:	Dr Peter Dueben (ECMWF)
Project Start Date:	01/04/2021
Project Duration:	36 months

Published by the MAELSTROM Consortium

Contact:
ECMWF, Shinfield Park, Reading, RG2 9AX, United Kingdom
Peter.Dueben@ecmwf.int

The MAELSTROM project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955513. The JU receives support from the European Union's Horizon 2020 research and innovation programme and United Kingdom, Germany, Italy, Luxembourg, Switzerland, Norway

Contents

Figures

Tables

1 Executive Summary

This document reports on the prospects for the use of machine learning within data assimilation algorithms for weather forecasting. This is a special application of the work delivered in Application 3 as part of Work Package 1. This application focused on the creation of emulators for components of weather forecasting models. Accurate emulators were created as documented in Deliverable 1.6. The emulators can, in principle, not only be used for the non-linear forecast model, but also as linearised versions (gradient versions) for forward and backward propagation in the 4DVar data assimilation framework at ECMWF. Gradient versions of these machine learning based emulators are here automatically generated using machine learning libraries, and their accuracy and stability are assessed. We document the motivation, results and prospects for future exploitation, and investigate whether the gradient version that is automatically generated from the machine learning libraries is sufficient for the high level of accuracy that is required in the data assimilation framework. If the gradient version turns out to be useful for 4DVar data assimilation, this opens another important use case for machine learning emulators in numerical weather predictions, additionally to the straight-forward use in the forward models.

2 Introduction

2.1 About MAELSTROM

To develop Europe's computer architecture of the future, MAELSTROM will co-design bespoke compute system designs for optimal application performance and energy efficiency, a software framework to optimise usability and training efficiency for machine learning at scale, and large-scale machine learning applications for the domain of weather and climate science.

The MAELSTROM compute system designs will benchmark the applications across a range of computing systems regarding energy consumption, time-to-solution, numerical precision and solution accuracy. Customised compute systems will be designed that are optimised for application needs to strengthen Europe's high-performance computing portfolio and to pull recent hardware developments, driven by general machine learning applications, toward needs of weather and climate applications.

The MAELSTROM software framework will enable scientists to apply and compare machine learning tools and libraries efficiently across a wide range of computer systems. A user interface will link application developers with compute system designers, and automated benchmarking and error detection of machine learning solutions will be performed during the development phase. Tools will be published as open source.

The MAELSTROM machine learning applications will cover all important components of the workflow of weather and climate predictions including the processing of observations, the assimilation of observations to generate initial and reference conditions, model simulations, as well as post-processing of model data and the development of forecast products. For each application, benchmark datasets with up to 10 terabytes of data will be published online for training and machine learning tool-developments at the scale of the fastest supercomputers in the world. MAELSTROM machine learning solutions will serve as blueprint for a wide range of machine learning applications on supercomputers in the future.

2.2 Scope of this deliverable

2.2.1 Objectives of this deliverable

To document the prospects for the use of automatically generated gradients of machine learning emulators within gradient-based data assimilation algorithms.

2.2.2 Work performed in this deliverable

Machine learning emulators developed in previous deliverables in for Application 3 of Work Package 1 have been tested to assess the behaviour of automatically generated gradient information. The smoothness and stability of these gradients have been measured.

2.2.3 Deviations and counter measures

No significant deviations occurred and no counter measures were required.

3 Background

3.1 4D-var data assimilation

3.1.1 Introduction

In weather forecasting, data assimilation refers to the process of determining the current model state of the atmosphere to utilise it as the initial conditions for subsequent weather predictions. This is a minimisation problem, where one seeks to find the state of a weather forecasting model that best fits observations over a window of time. Multiple algorithms are currently used by operational weather forecasting centres in their daily activities, particularly prevalent are ensemble Kalman filter (EnKF, Evensen 2003) approaches and variational data assimilation methodologies, including 3D- and 4D-var. At ECMWF, an incremental (ensemble) 4D-var technique is used (Bonavita et al. 2016). In ECMWF's implementation of incremental 4D-var, atmospheric state gradients are propagated forward and backward through the IFS, ECMWF's operational (physical) weather forecasting model. These provide the necessary information to increment the initial estimate of the atmosphere to better match the observations. Via an iterative approach, the initial state is incremented multiple times towards a more optimal estimate of the current atmospheric state. To operate this method, additional versions of the IFS are required to propagate these gradients. For the components of the IFS, e.g. for each parameterised physical process, tangent-linear (TL) and adjoint (AD) versions of the nonlinear components must be derived. The TL calculates how an increment to the input atmospheric state increments the output. The adjoint model acts as an inverse to the TL model, and when provided with an initial state of the atmosphere, and an increment to the output state, provides the increment to the input state.

Currently, at ECMWF, the derivation and maintenance of TL and AD models is a task managed by hand, with humans deriving and coding these models. For some components, where strong nonlinearities exist, reformulation is required to ensure a necessary level of smoothness, e.g. via "IF" statements in the code. This derivation and maintenance has a significant cost, often resulting in further simplifications/approximations. For example, at the UK Met Office, a simplified set of physics schemes is used in data assimilation (Rawlins et al. 2007). At ECMWF, some areas of TL and ADcode have not been kept up with developments for the nonlinear forecasting model, meaning that advances in physical modelling are not being fully utilised in data assimilation. Radiative transfer is one example of this, where the older, less accurate, Morcrette scheme is the basis for the TL/AD versions within 4D (Morcrette 1991). By contrast the more recent TripleClouds scheme (Shonk and Hogan 2008) within the ecRad radiation scheme is used in the forecasting model. An upgrade of the TL/AD code to match the more complex ecRad scheme would almost certainly lead to improvements in forecast scores. However, the generation of such TL/AD code by hand is time consuming, and represents a significant amount of work.

Here lies the opportunity for machine learning. Within MAELSTROM, highly accurate emulators of the TripleClouds radiative transfer scheme have been trained using neural networks. In Deliverable 1.6, we document that emulators are stable and accurate even when coupled to the full IFS weather forecasting model. These emulators are written in Python-based machine learning frameworks, here Tensorflow, which provide auto-differentiability functionality. Such auto-differentiability is normally used to train the emulators, but can be deployed to generate TL and AD models, even for complex neural networks. The task of auto-differentiation is natural for machine learning tools such as

Tensorflow as it is also required to perform the backpropagation during the optimisation of weights in the neural networks.

In this deliverable, we demonstrate how auto-differentiability may be used for the derivation of TL and AD models, and test offline the behaviour of these models. We will begin by exploring this topic in the setting of a simpler parametrised physics scheme, the non-orographic gravity wave drag. We will then move onto the topic of radiative transfer.

3.1.2 TL and AD equations

For a model, $\mathcal{F}$, with $\mathbf{K}(x)$ denoting the Jacobian matrix of partial derivatives of $\mathcal{F}$ evaluated at x. Then the TL is written as

$$\delta y = \mathbf{K}(x)\,\delta x$$

The adjoint is written

$$Adjoint = \mathbf{K}(x)^T \delta y$$

where T denotes the transpose. These equations must be calculated for the parametrised physics schemes.

3.1.3 TL and AD tests

For stable and accurate convergence, two properties are required from the TL and AD models and their relationship to the nonlinear model. These are necessary but not sufficient conditions for successful use within data-assimilation.

First, for small increments to the state, the TL should be close to a finite difference estimation. Mathematically, for a model, $\mathcal{F}$, with the corresponding TL version, **F,** the following expression

$$\lim_{a \to 0} \mathcal{F}\ (x_0 + \alpha\delta x) - \mathcal{F}(x_0) - \mathbf{F}(x_0)\alpha\delta x] = 0$$

should hold for plausible perturbations δx. This test essentially measures the accuracy of the TL model as a first-order Taylor approximation to the nonlinear model.

Secondly, the symmetry between the TL and adjoint models should be preserved.

$$[\,\mathbf{F}(x_0)\delta x\,]^T \delta y = \delta x^T\,[\,\mathbf{F}^T(x_0)\delta y\,]$$

where δy is an arbitrary vector.

In this deliverable, we primarily assess these two tests, where the TL and AD models are calculated using auto-differentiability of a neural network emulator.

4 Non-orographic gravity wave drag

In work preceding MAELSTROM, a machine learning emulator was developed for the non-orographic gravity wave drag (NOGWD) process (Chantry et al. 2021). This process is important for stratospheric predictability, in particular for the quasi-biennial oscillation (QBO). For non-orographic gravity wave drag, a simple machine learning solution was sufficient, requiring a multi-layer perceptron. For this process, both TL and AD models already exist within the IFS, meaning that the accuracy of the ML-based TL/AD estimation method can be directly compared with the human-derived equivalents that are used daily in operational forecasting. The work described below was published in Hatfield et al 2021, where further details can be found.

4.1 Deriving TL and AD models for NOGWD

For simple neural networks, the automatic differentiation feature of machine learning frameworks is not required, as manual derivation of these is relatively straightforward, as detailed in Hatfield et al., 2021. Using the manual derivation approach enables their integration in Fortran, the programming language used for the IFS. Code for calling the TL and AD versions of the neural network was written in Fortran, and tested in the IFS. Despite the fact that a human remains in this process, it is still an instructive test, as it measures the smoothness of machine learning based emulators to perturbations in the input and output state.

4.2 Testing TL and AD properties

Firstly, we test the necessary conditions for stable and accurate convergence in Section 3.1.3 that describe the relationship between the TL and the finite difference estimate, as well as the TL and AD.

Figure 1 (taken from Hatfield et al 2021) shows the change in accuracy of the TL test, compared with physics-based nonlinear and TL models. In the easy test, machine learning is used for both the nonlinear model and TL model. This test therefore measures the accuracy of the neural network's estimated derivative. In the "hard" test the physics-based nonlinear model is used for the finite-difference estimate, but the neural network is used to estimate the TL. In both tests, the neural network-based TL behaves similarly to the physics-based TL, with the "easy" test slightly outperforming the full physics solution and the "hard" test providing a minor degradation. Overall the neural network derivative shows a comparable accuracy to the physics-based hand derivation.

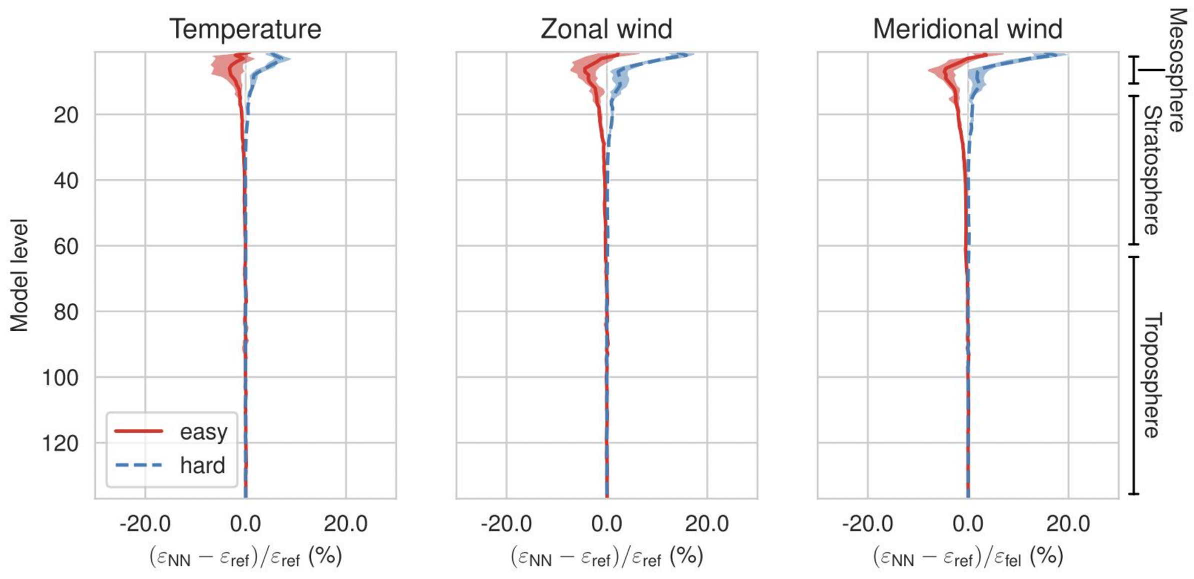

Figure 1: Comparison between the neural network TL and hand-derived TL in remaining close to a Taylor approximation of the nonlinear model (see Hatfield et al. 2021 for further technical details). Negative numbers indicate a better fit for the neural network, and positive a worse fit. For the "easy" test, where the neural network derivative is compared to the neural network itself the fit is equal or improved. When the neural network derivative is used to estimate the physical nonlinear model, the fit is somewhat degraded at the top of the atmosphere.

When assessing the second test described in section 3.1.3, the adjoint test, we measure the symmetry in a wide number of scenarios of input and output perturbations to the neural network state. In summary, we find that this symmetry property is maintained to at least 9 decimal places across all tests (see Hatfield et al. 2021 for further details).

The neural network-based TL and ADmodels both are considered to have successfully passed the offline tests, meaning that these models can be tested within a full data assimilation minimisation process.

4.3 Testing TL and AD models within data assimilation

As described above, the tests in Section 4.2 are considered necessary but not sufficient for success in data assimilation. Here, we test the full use of the derivative models within the IFS's data assimilation algorithm. We assessed the effectiveness of our neural network-based TL and AD models in data assimilation through a series of 4D-Var data assimilation experiments. The experiment ran continuously throughout the winter season, from December 1st, 2018 to February 28th, 2019. We implemented 4D-Var data assimilation cycles every 12 hours, and 10-day forecasts were generated from the final analysis produced by each cycle. Forecasts were run at TCo399, approximately equivalent to a 25 km resolution. This was used for both the reference and test runs. For further details on the configuration of the tests carried out see Hatfield et al. 2021.

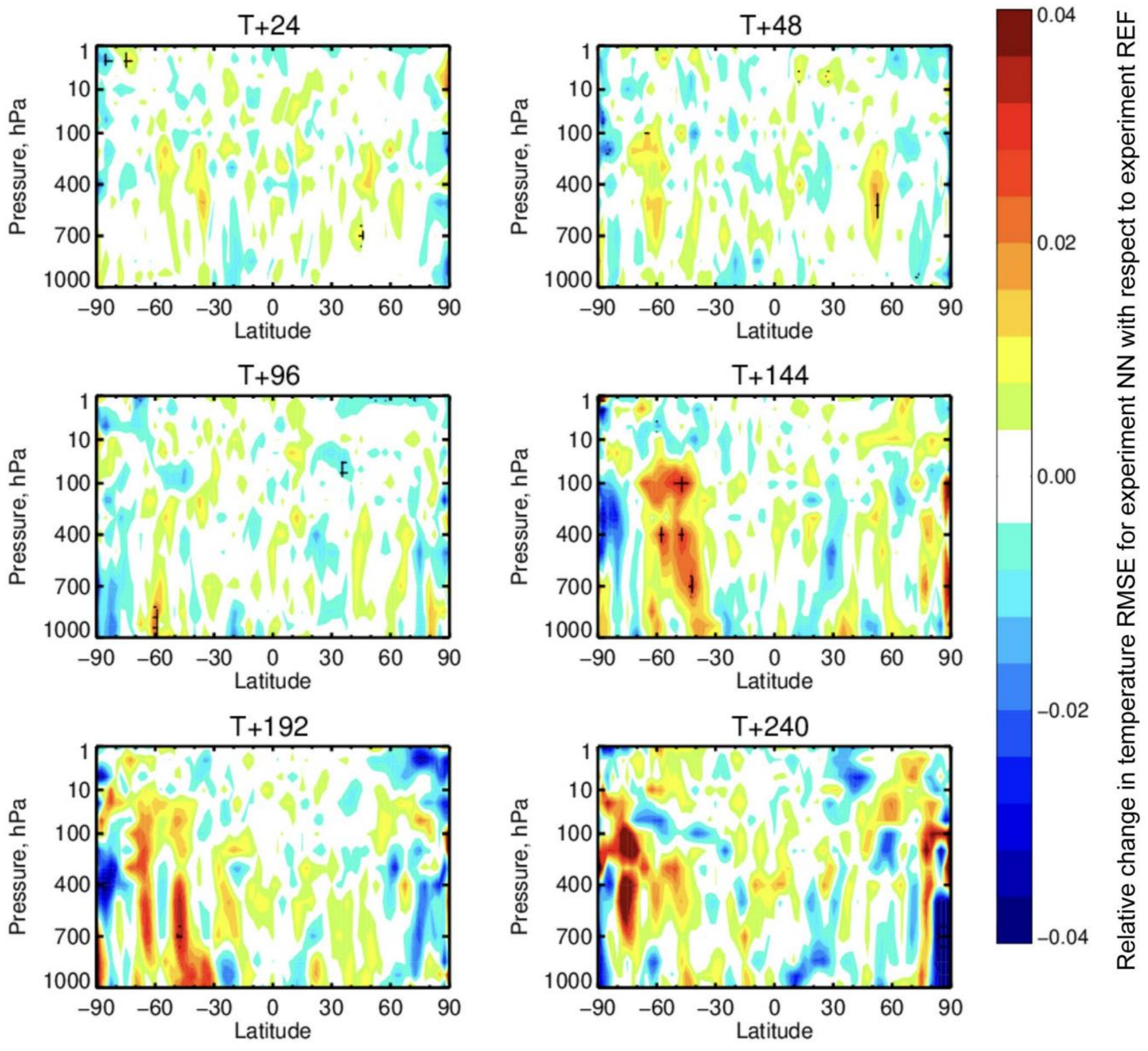

Figure 2: The relative differencein root-mean-square error of temperature of experiment NN compared with the reference experiment (REF, blue indicates that NN is better than REF) averaged in the zonal direction and across all forecasts in the three-monthly experimental period, for a number of forecast lead times (indicated in each subfigure title by "T+" with the number of hours after the initial time). The relative difference is computed by dividing by the error of REF. Hatching indicates that differences are significant with 95% confidence, and the sparsity of hatching indicates that the error difference is for the most part not statistically significant. (Figure taken from Hatfield et al. 2021.)

Figure 2 shows the change in accuracy of historical weather forecasts when using the NOGWD neural networks for the TL and AD models. Overall the accuracy changes are minimal, with almost no statistically significant degradation shown. This can be therefore viewed as a successful test.

4.4 Summary of NOGWD

In both offline and online tests, the neural network emulator-based TL and AD models were successful in providing gradient information, enabling accurate data assimilation to be carried out. However, it should be emphasised that non-orographic gravity wave drag is one of the simpler and less impactful parameterisation schemes in a weather forecasting model. This does not impact the validity of the results, but should provide caution in extrapolating to other, more nonlinear and more impactful parametrisation schemes. We therefore moved on to test radiative transfer, which will be a more challenging test of the approach.

5 Radiative transfer

5.1 Background on radiative transfer

The radiative transfer processes within a weather forecasting model capture longwave and shortwave heating and cooling, involving interactions with clouds, aerosols and the land surface. These processes are physically well understood, but their computation is too expensive to fully capture them in a weather or climate forecasting model. Therefore approximations are made to create computationally affordable, yet accurate, estimates of the processes. This is an area of continual development, with new approximations and algorithms to achieve better estimates within a computational budget. Ideally, for operational centres using variational approaches for data assimilation, new TL and AD versions would be required whenever an update to the nonlinear radiation scheme is operationalised. Unfortunately, due to the laborious work and time involved in hand-deriving these models, this is not always possible, and thus, there are inconsistencies between models used in the forecasting model and those used within the data assimilation minimisation.

In MAELSTROM, focus has been on learning an emulator of the Tripleclouds solver within ecRad, which is used operationally in the IFS. As reported in Deliverable 1.6, these emulators have been tested not only in isolation, but also when coupled to the IFS for weather forecast experiments. In all tests, the emulator is exceptionally accurate, with minimal deviations seen when compared with the reference model. Bespoke LSTM-based solutions were created, which mimic the information flow of the physical system.

Here, we will focus on the shortwave model, which constitutes one of the two trained and tested models. Similarly to the NOGWD results, we wish to test the accuracy of the TL and AD models. However, due to the significantly increased complexity of these models when compared with the fully-connected neural networks used in NOGWD, a hand-derivation of the neural network gradients is far less appealing. Instead we used the automatic differentiation provided by the Tensorflow framework to construct these models.

5.2 Deriving TL and AD models using Tensorflow

The radiative transfer emulator is implemented using Tensorflow (https://www.tensorflow.org), an open-source and widely used deep learning framework. The code for training the neural network emulators used here can be found under https://git.ecmwf.int/projects/MLFET/repos/maelstrom-radiation. Tensorflow provides access to different features and methods that facilitate the definition of the model architecture and its training. A fundamental part of training and optimising a neural network relies on the backpropagation algorithm. Using the chain rule, backpropagation performs a backward pass adjusting the model's parameter after each forward pass.

While most of the complexity of this algorithm is abstracted away from the user when using Tensorflow, it is still possible to watch and access the gradients computed across the model, using the API '`tf.GradientTape`'[1]. This API provides easy access to automatic differentiation so that it first records any relevant operations executed within the API context into a "tape" and then uses reverse mode differentiation to compute the gradients about some specified inputs. This concept of GradientTape is specific to Tensorflow, but it is worth pointing out that other common deep learning frameworks like Pytorch or JAX also provide access to similar APIs to access the gradients.

As already introduced in previous sections, the derivation of the TL model can be performed by constructing the Jacobian matrix. To obtain the Jacobian with `tf.GradientTape` one can use the method `tf.GradientTape.jacobian.` Below we provide a pseudocode example to better understand how this method works and how it can be used in the context of TL model derivation using neural networks:

```
trained_model_path = 'trained_model.h5'
model = tf.keras.models.load_model(trained_model_path)

x= tf.random.uniform([3,6]) # Input Variable

with tf.GradientTape() as tape:

    y=model(x)

J = tape.jacobian(y, x)
```

Assuming we have an already trained model stored under a given path, the model is loaded with TensorFlow to generate a new prediction for a given input x. To correctly track the gradients across the operations that take place inside the model, the inference step needs to be defined within the `GradientTape` context. Assuming the input vector x has a shape m, and output vector y has a shape of n, the resulting Jacobian matrix J will have a shape equal to $n \times m$. Summing over the output's dimension, it is possible to obtain the gradients that would have been calculated using `tf.GradientTape.gradient.`

By default, a tape allows one to compute one set of gradients. To compute multiple gradients or higher-order gradients like the Hessian Matrix, the tape has to be created using `persistent=True.` When working with machine learning models, it is also very common to define your dataset in samples of a given batch size. Following the example from above, we would have the input vector x with a shape (b, m) where b would refer to the batch size and the output vector y would have a shape (b, n), and a Jacobian matrix J with shape (b, n, b, m). If there is independence between $x[i,:]$ *and* $y[j,:]$ *for* $j \neq i$, then there is a more efficient computation of the Jacobian available via `tf.GradientTape.batch_jacobian` that would return a matrix of shape (b, n, m) that stores the diagonal values as rows.

[1] https://www.tensorflow.org/api_docs/python/tf/GradientTape

When dealing with multiple inputs and multiple outputs, it is still possible to use the `GradientTape` API to compute the Jacobian as the input x can take other data types like for example dictionaries, where each entry of the dictionary refers to one of the input variables. Output variables can equally be a dictionary where each entry refers to a different variable. However in that case, `tf.GradientTape.jacobian` would be required to compute the Jacobian for each combination of output-input variables.

Approaching the TL via the construction of the Jacobian is not the most computationally efficient approach, but means this cost is still estimated to be small when leveraging GPUs. The advantage of this approach is that the AD model can be trivially generated via the transpose of the Jacobian, and the symmetry property will be satisfied to the level of numerical precision used.

5.3 Deriving TL and AD models for the radiative transfer emulator

The emulation radiative transfer process for short-wave and long-wave radiations represents a multiple input/multiple output problem where a deep learning model predominantly based on RNN architecture, specifically using LSTM blocks, is used to propagate information in the vertical dimension. In particular, the model receives input profiles of atmospheric state and composition that can be divided into full model levels (137 levels in the IFS model), model half-levels (138 levels), model interfaces (136 levels) & surface variables (scalars). As outputs, the model provides downwards and upwards fluxes as well as the resulting heating rates that can be derived from those fluxes.

Using an already trained model for short-wave radiation, we aim to assess if the TL model can be derived using the nonlinear model's automatically generated gradients. Since the AD model formulation here uses the transpose of the Jacobian matrix, the symmetry between TL and AD will be naturally satisfied to numerical precision. Focus will be spent in evaluating the TL property, but the AD model can easily be generated with this approach.

The heating rates are obtained from the fluxes according to the equation:

$$\frac{dT}{dt} = \frac{g}{c_p} \frac{F^n_{i+1/_2} - F^n_{i-1/_2}}{p_{i+i-1/_2} - p_{i+i-1/2}}$$

where F^n denotes the net flux (downwards minus upwards), p is the pressure, g is the gravitational constant, c_p is the specific heat at constant pressure of most air, and i refers to the vertical index. As such, the heating rate TL and AD models can be ignored.

Generally, the trained model can be written as a vector-valued function that maps an input x to an output y:

$$y = \boldsymbol{F}(x)$$

Here, the TL of the model is given by:

$$\delta y = \boldsymbol{F}(x, \delta x) = \boldsymbol{K}(x)\delta x$$

In case of r the radiative transfer the input x contains a set of different variables which can be decomposed as follows;

$$\delta y \;=\; \mathbf{F}(x,\delta x) \;= \mathbf{K}(x)_{SURF}\delta x_{SURF} + \mathbf{K}(x)_{FL}\delta x_{FL} + \mathbf{K}(x)_{HL}\delta x_{HL} + \mathbf{K}(x)_{I}\delta x_{I}$$

Here, $SURF$ refers to input surface variables (skin temperature, solar zenith angle and albedos), FL refers to the input full-level variables (humidity, cloud fraction, etc), HL to input half-level variables (temperature and pressure) and I refer to the interface fields.

5.4 Testing methodology

Here, we present results for a perturbation of one input field to the radiative transfer scheme. This is presented to provide easier and more concise analysis, as the dimensionality of the input state for either shortwave or longwave radiative transfer is large. The results shown here, for humidity perturbations, are representative of the other variables tested.

Once a Jacobian has been constructed, the last remaining piece is testing the models with plausible increments to the physical state. It is important that the perturbations tested are, to some level, physical. Otherwise the neural network may be asked to generalise to completely unseen physical states, which may lead to irregular gradient information. Ideally, one would use increments from the data assimilation process, to give a true test of the application. Unfortunately the full state of increment information used during data assimilation at ECMWF is not stored, so this cannot be achieved without significant overhead. In its place, we instead generate plausible perturbations by calculating the spatial increment between a column and a nearby neighbour column. We tested different variations to compute the spatial increment. Although the results were similar among those tests, we opted to define the perturbation profile as the difference between a reference column (generally column 0) and a close-by column (column 4) to obtain a profile with perturbation magnitudes that were neither too extreme nor too simple. By selecting this approach, both the original input state and the perturbed state come from realistic states of the atmosphere. Figure 3 shows the spatial variation in humidity inputs and how different amplitudes of perturbation will affect the humidity field.

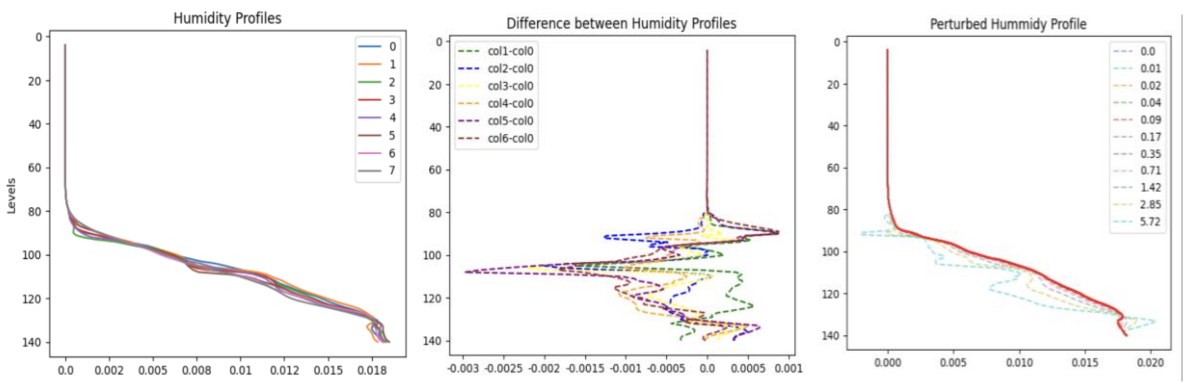

Figure 3: Computation of spatial perturbation between adjacent columns. The figure on the left shows the vertical profile for humidity of 8 different columns located in South Africa at midday. The figure in the middle displays the spatial increment for humidity where column 0 is used as a reference. Based on the magnitude of the humidity differences, we choose 'col4-col0' as a plausible perturbation. The right figure shows how the selected perturbed profile is applied on top of the original humidity profile for a given column for a range of alphas.

5.5 Results

Figure 4 shows results from a single (illustrative) column, comparing the neural network and the finite difference TL models. Here various values of alpha are used that represent the perturbation amplitudes to the humidity profile. The sensitivity of the downwards and upwards models have differing sensitivities, as could be expected from the physical processes. For downwards fluxes, sensitivity is only expected below the level where humidity takes an active role in the radiative transfer. This is seen for both neural network and finite difference versions. For upwards fluxes, perturbations to the humidity in the troposphere change the amount of energy that is reflected back upwards. Hence the outgoing fluxes show sensitivity higher in the atmosphere, at and above the levels where humidity is perturbed. Near the top of the IFS model's atmosphere, the sensitivity of upwards fluxes to humidity changes is near constant. Again this is captured by both gradient models. Only for the largest amplitude perturbations of humidity,we see significant differences between neural network and finite difference estimates.

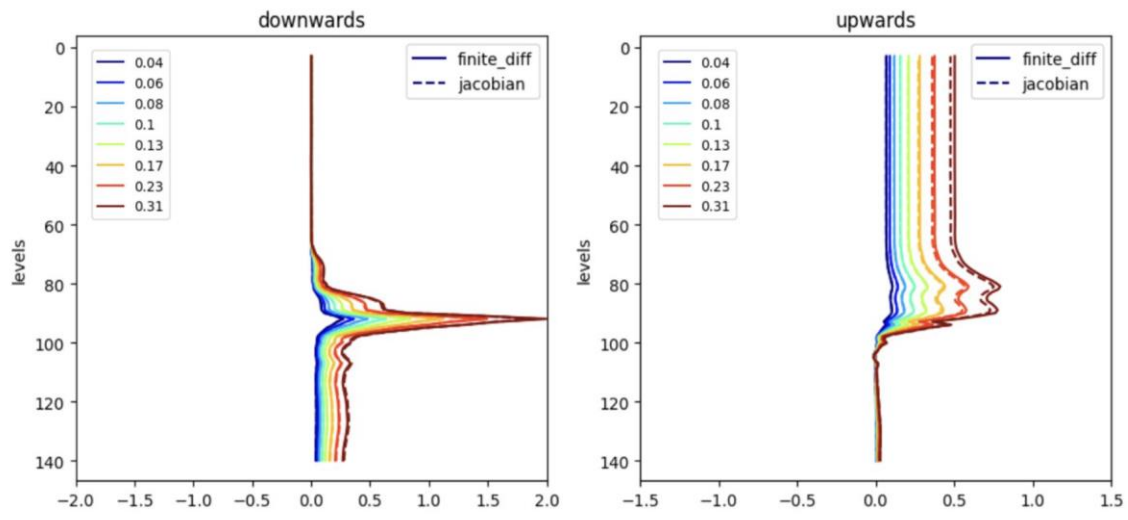

Figure 4: Comparison between the finite difference and Jacobian tangent-linears for the downwards (left) and upwards (right) fluxes when perturbations to humidity are made. For the largest alphas, differences are detectable in the upwards fluxes, but otherwise both outputs are indistinguishable.

In Figure 5 and 6,the ratio of the Jacobian-based neural network TL over the finite difference estimate is plotted. For both figures, 8 representative columns are shown, to show the variation in the behaviour. All 8 are neighbours in the atmosphere, but testing highlighted that these columns were representative of the general behaviour. Here green shading represents a ratio of one, i.e. the neural network and finite difference versions satisfy the desired relationship. Figure 5 shows the behaviour of the downward fluxes for humidity perturbations, Figure 6 illustrates the same but for upward fluxes. This type of plot can quickly highlight where the two approaches for TL generation differ significantly.

Due to the directionality of the shortwave process, and lack of humidity in the upper 60 layers of the atmosphere, there is little sensitivity to humidity perturbations for downwards fluxes. The finite different TL amplitudes are small, and thus, the use of a relative error becomes problematic and the ratio becomes noisy. Lower in the atmosphere, and for the full profile of upwards fluxes, there is overall a wide range of α for which the ratio is close to one, and hence the two TL models are comparable. However, significant deviations from one are apparent for a small number of layers in some example columns for both, the downwards and upwards fluxes.. One example is visible near model level 120 for the downwards fluxes in example column number 6.

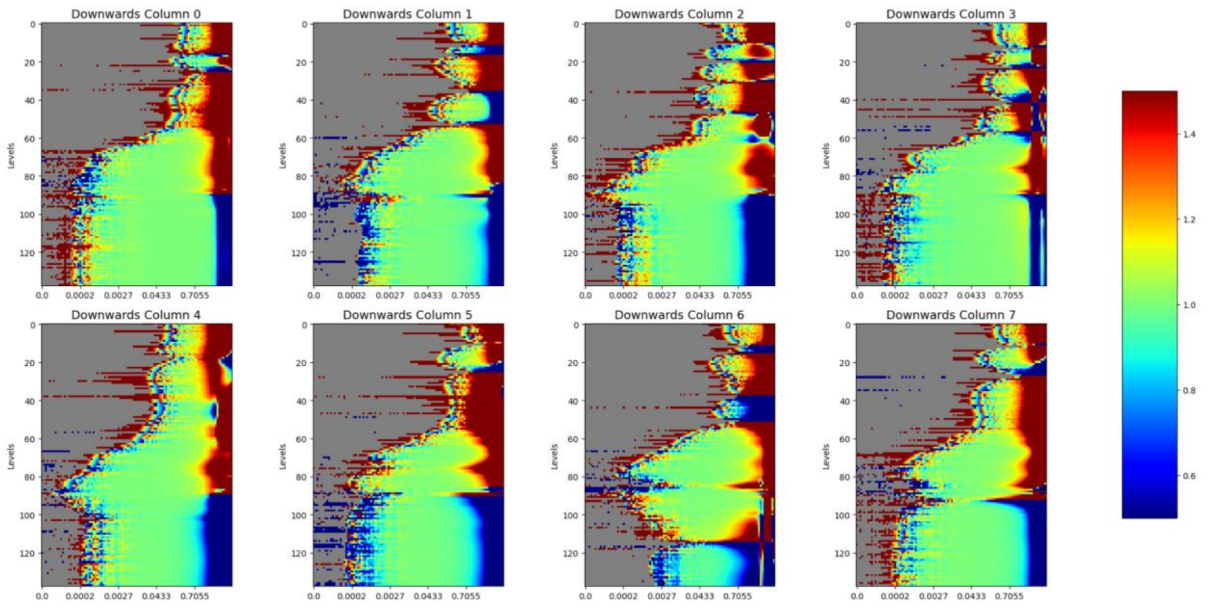

Figure 5: Ratio of the neural network Jacobian-based TL estimate, over the finite difference estimate. Green shows the ideal ratio of one. The x-axis denotes the value of alpha used, i.e. the amplitude of the perturbation (see the first equation in 3.1.3). The y-axis shows this ratio for different model levels, i.e. heights in the atmospheric model. Grey denotes where the denominator, i.e. the finite-difference estimate, has a value less than 10^{-4}. The 8 subplots denote 8 columns, chosen over Africa during midday.

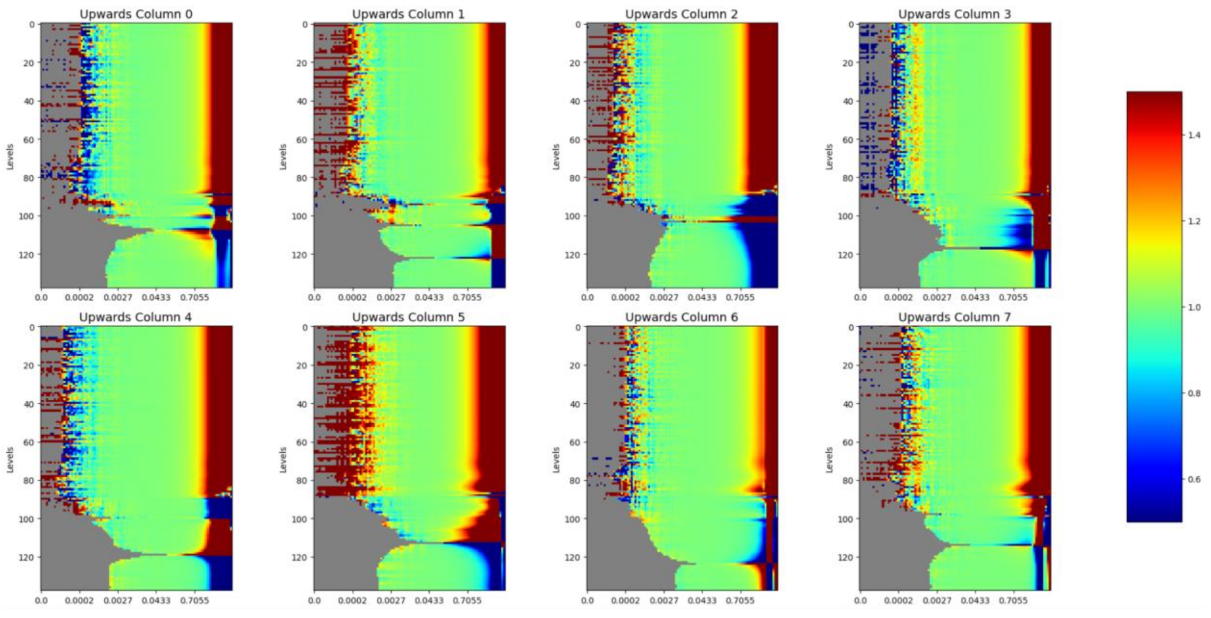

Figure 6: As with figure 5 but for upward fluxes.

Examination of other locations, as well as perturbations to other input variables, provides consistent results with those presented here for humidity. For the majority of the levels, and for a wide range of amplitudes of perturbations, the neural-network TL behaves similarly to the finite difference estimate. The quality of the match is considered sufficiently good that online testing constitutes a worthwhile endeavour.

6 Conclusions and exploitation

Here we have examined the validity of using neural networks to generate tangent-linear and adjoint models of parametrised physics, for use within variational data assimilation in numerical weather prediction.

For the relatively simple process of non-orographic gravity wave drag, offline and online testing has been carried out with success. Compared against manually derived models, which are considered as a benchmark, there is almost no statistically significant degradation found when using neural networks. This proof of concept motivated further testing with the more challenging radiative transfer parameterisation scheme.

For radiative transfer, featuring far more complex neural network solutions, offline testing of the smoothness of automatically derived tangent-linear models has been carried out. For a wide range of perturbation amplitudes, the neural network-based TL model is close to the finite difference version. There are only rare exceptions where the two deviate.

Several avenues exist to further improve the quality of the neural network TL models. Firstly, the radiative transfer shortwave model was not trained with weight decay, which encourages non-active elements of the neural network to be switched off. Such an approach may help to create even smoother models. Secondly, the use of reduced numerical precision should be explored to understand its possible role in the noisy estimates.

The broadly positive outcome of this testing is a necessary but not sufficient condition for success within variational data assimilation. The results are considered positive enough to invest in online testing of the approach, which will happen after the end of MAELSTROM. For this purpose, the library Infero[2] must be expanded in functionality, to expose the creation of Jacobians within the Fortran interface.

Success in online testing would be a major result for variational data assimilation methods. As stressed in the introduction, simplications and inconsistencies are currently present in variational data assimilation resulting from the complexity of human-generated tangent-linear and adjoint models. If this could be automated, via the creation of training datasets, training of emulators, and then the use of automatic differentiation, it would provide a blueprint for future successful integration into operational weather forecasting model chains..

[2] https://github.com/ecmwf/infero

7 References

Bonavita, M., Hólm, E., Isaksen, L. and Fisher, M., 2016. The evolution of the ECMWF hybrid data assimilation system. *Quarterly Journal of the Royal Meteorological Society*, *142*(694), pp.287-303.

Chantry, M., Hatfield, S., Dueben, P., Polichtchouk, I. and Palmer, T., 2021. Machine learning emulation of gravity wave drag in numerical weather forecasting. Journal of Advances in Modeling Earth Systems, 13(7), p.e2021MS002477.

Evensen, G., 2003. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, *53*, pp.343-367.

Hatfield, S., Chantry, M., Dueben, P., Lopez, P., Geer, A. and Palmer, T., 2021. Building tangent-linear and adjoint models for data assimilation with neural networks. Journal of Advances in Modeling Earth Systems, 13(9), p.e2021MS002521.

Morcrette, J.J., 1991. Radiation and cloud radiative properties in the European Centre for Medium Range Weather Forecasts forecasting system. Journal of Geophysical Research: Atmospheres, 96(D5), pp.9121-9132.

Rawlins, F., Ballard, S.P., Bovis, K.J., Clayton, A.M., Li, D., Inverarity, G.W., Lorenc, A.C. and Payne, T.J., 2007. The Met Office global four-dimensional variational data assimilation scheme. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, *133*(623), pp.347-362.

Shonk, J.K. and Hogan, R.J., 2008. Tripleclouds: An efficient method for representing horizontal cloud inhomogeneity in 1D radiation schemes by using three regions at each height. Journal of Climate, 21(11), pp.2352-2370.

Document History

Version	Author(s)	Date	Changes
	Name (Organisation)	dd/mm/yyyy	
1.0	Matthew Chantry	05/03/2024	Initial draft
1.1	Ana Prieto Nemesio	13/03/2024	Added section 5.2 and 5.4
1.2	Matthew Chantry	22/03/2024	Results section added
1.3	Peter Dueben & Ana Prieto Nemesio	25/03/2024	Additional content, remarks and corrections
1.4	Matthew Chantry	26/03/2024	Changes recommended by review

Internal Review History

Internal Reviewers	Date	Comments
Michael Langguth (FJZ)	26/03/2024	Minor changes requested

Estimated Effort Contribution per Partner

Partner	Effort
ECMWF	1PM
Total	**1PM**