MLP: Machine learning for weather prediction

G003 (s1611423, s1652610, s1660124)

Abstract

Widely available weather forecasts are created using Numerical Weather Prediction (NWP) models that are highly complex and also a source of uncertainties due to the chaotic nature of the equations that govern the atmosphere. In our work we focus on the performance of machine learning models that have a potential to outperform NWP models. Specifically, they attempt to circumvent the limits of chaotic equations by using a data-driven approach. We evaluate the performance of three different architectures by comparing them to a simple CNN baseline network, and also to the state-of-the-art numerical models. We find that the ConvLSTM network achieves the best performance, beating both the U-Net and VGG-16 architectures, which struggle to improve upon the baseline. However, the NWP models still outperform our models by a large margin on a three day forecast.

1. Introduction

Weather forecasting has been around since the 19th century when a heavy storm caused the loss of a ship named *Royal Charter*. This inspired 2 navy officers to set up a communication chain of telegraphs to transmit daily information about weather serving as a gale warning service to prevent similar disasters. Nowadays, we use Numerical Weather Prediction (NWP) to generate the daily forecasts you see all around you.

The atmosphere is a fluid which allows us to construct complex mathematical models using fluid dynamics and thermodynamics, which given the precise state of the atmosphere at the current time are able to calculate an "accurate" weather forecast (Richardson, 1922). However, even these accurate models have a flaw, where it's impossible to solve certain partial differential equations that govern the atmosphere exactly due to its chaotic nature (measurement and processing noise), and so even the best models with correct inputs will not be reliable for long term predictions.

Machine learning weather prediction tries to work around this uncertainty by using a data-driven approach to construct the model, mapping exogenous input to a target output without requiring a complex understanding of the underlying physical processes built into the model. We will be working with deep learning models as they engineer their own features during training time (LeCun, 2015), therefore we do not require domain knowledge that would be necessary to hand-craft meaningful features.

2. Task and data

2.1. Original MIDAS dataset

Initially, we were planning to use the *Met Office Integrated Data Archive System (MIDAS)* dataset collection[1], which contains 17 datasets of daily, hourly, and sub-hourly weather observations across the UK. While this dataset conveniently contained the ground truth and a large time range of data, our evaluation would be limited to our own baseline. The data was also only labeled by a weather station id, not a geographical location of the station, and not all years contained the data for all stations. For us to be able to use geographically connected data, then, a lot of preprocessing would have been required with uncertain results.

2.2. ERA5

Instead, we chose to use the same dataset as (Rasp et al., 2020) in order to be able to compare our models with their baselines. The raw data comes from the ERA5 climate reanalysis archive[2] produced by ECMWF, and contains hourly data on several weather parameters from the year 1979 onward.

Reanalysis means that archive weather observations have been *re-analysed* using ECMWF's forecast models and data assimilation systems to produce global data for the entire world, including the gaps with no weather stations. This largely solves the issue we observed with MIDAS since it only provided discrete weather station data. ERA5, on the other hand, leaves no gaps in the data for the entire Earth, but rather provides an estimate of uncertainty, seeing as the data cannot technically be considered ground truth. This is very important for using the data as input for a CNN, which relies on seeing the full scope of the data rather than just some parts.

Since downloading the raw data takes weeks, we opted for using the already pre-processed datasets provided by (Rasp et al., 2020) hosted alongside their WeatherBench project [3]. The data resolution is given in degrees of latitude and longitude, determining the size of the grid that covers the data for the entire Earth. While the original ERA5 data has

[1] https://catalogue.ceda.ac.uk/uuid/220a65615218d5c9cc9e4785a3234bd0

[2] https://climate.copernicus.eu/climate-reanalysis

[3] https://github.com/pangeo-data/WeatherBench

resolution of 0.25°×0.25° and most atmospheric parameters on 37 pressure levels, this takes up terabytes of disk space, making it unfeasible for us to work with. Instead, the pre-processed data we used had a 5.625°×5.625° resolution and most atmospheric parameters on only 11 pressure levels. This meant the data was laid out in a 64×64 grid instead of 1440×1440, resulting in over a 500-fold decrease in resolution and a convenient power-of-two size, which can slightly improve the performance of the model. Even with such a significant reduction in resolution, though, many datasets took up over 20GB of disk space.

2.3. Research question

We set out to see if it was a viable strategy to predict weather using machine learning and if so, which architectures and architectural choices would be the best, as well as how close we are able to get to the results of the physical NWP models employed by most weather forecasting services currently.

2.4. Evaluation using RMSE

For the evaluation metric we chose root mean squared error (RMSE) because, as (Rasp et al., 2020) states, it "is easy to compute and mirrors the loss used for most ML applications". More importantly, we chose it because the original WeatherBench model and the operational models Rasp et al. use have already been evaluated using RMSE. Using the same metric allows us to easily and confidently compare our models with theirs to see if we have managed to improve upon them or not.

3. Methodology

3.1. Baselines description

For our primary baseline we use the direct CNN model from (Rasp et al., 2020). It is a 5-layer CNN, where the hidden layers have 64 channels, kernel size of 5, an ELU activation with an Adam optimizer and a learning rate of 0.0001. The input and output layers have only two channels each, representing the Z500 (geopotential) and T850 (temperature) data used for training, which will be further discussed in section 3.5. With minor modifications discussed in section 4.1, we use this model as the base for most of our experiments.

We compare our models performance with fully convolutional baseline models as well as numerical models described in (Rasp et al., 2020). The simplest baseline is persistence forecast, which predicts the forecast for every day as the weather for the previous day. The all-time climatology uses a mean of of all training data to predict forecasts for all days. The weekly climatology computes means for each of the 52 calendar weeks and uses these to predict forecasts.

Lastly, we also compare our model's performance with an operational NWP model and two modified IFS (Integrated Forecast System) models. The operational model is evaluated using the TIGGE[4] forecast archives. The other two IFS models are IFS42 and IFS63, respectively, which are physical models that were run with coarser resolution by Rasp et al. to be more in line with the computational resources of the neural network model.

3.2. LSTM model

Given our simple CNN baseline we had 2 options to improve baseline network, the more traditional approach of adding convolutional, pooling and up/down-sampling layers to increase the network's size and complexity (which we did in the U-Net experiment, section 4.6) or introduce a new type of layer to improve the capabilities of the network (which we did in the ConvLSTM experiment, section 4.3).

Because the input is 2D time-series data, it is textbook example for convolutional long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997; Gers et al., 2000) layer, as LSTMs are ideally suited for problems with complex time-dependent structure, such as weather prediction. This is because LSTM units contain a cell state, which is propagated forward in time, modulated by gates (learned weights), which leverage how much to "forget" from the cell state at the previous time step and how much new information to "input" to the cell from the data at the current time step. This enables the cells to make short-term modulations to their state while maintaining long-term behavior (Weyn, 2019).

(Weyn, 2019) used a single ConvLSTM layer at the start of a their CNN network (Figure 1) to implement LSTM network, we have decided to use this approach because it is easy to implement and has been shown to work. This required us to change the data loader we use to process input to our network. The CNN used 4D input (batch size, width, height, channels), we had to extend it a 5D input (batch size, time, width, height, channels). As previously mentioned, we are working with 2D time-series data points, where width and height are correlated and hence we decided to use a ConvLSTM layer instead of a simple LSTM one, as the former was specifically designed to be able to capture spatio-temporal correlation (Shi et al., 2015) present in our weather data.

3.3. Lead time selection

The baselines NWP models from (Rasp et al., 2020) are compared on performance of lead time of 3 and 5 days. To make our experiments comparable also keeping in mind out time constraints we have chosen to use the 3-day lead time as the basis for our experiments.

Originally, we intended to attempt lead time of 14 days as this is known to be limit of numerical models (Stern. & Davidson, 2015). However, due to tape drive failure[5] at the ECMWF we were unable to access the data for the baseline models.

[4] https://confluence.ecmwf.int/display/TIGGE
[5] ticket J0144200

Table 1
Neural Network Architecture for DLWP as a Sequence of Operation Layers

Layer	Filters	Filter size	Dilation	Output shape[a]	Trainable params[b]
input				(vt, 36, 144)	
ConvLSTM2D[c]	$2v$	3×3	2	(t, $4v$, 36, 144)[d]	736
Conv2D	32	3×3	2	(32, 36, 144)	608
MaxPooling2D		2×2		(32, 18, 72)	
Conv2D	64	3×3	1	(64, 18, 72)	18,496
MaxPooling2D		2×2		(64, 9, 36)	
Conv2D	128	3×3	1	(128, 9, 36)	73,856
UpSampling2D		2×2		(128, 18, 72)	
Conv2D	64	3×3	1	(64, 18, 72)	73,792
UpSampling2D		2×2		(64, 36, 144)	
Conv2D	32	3×3	2	(32, 36, 144)	18,464
Conv2D	v	5×5	1	(vt, 36, 144)	1602

Note. The parameter v represents the number of distinct variable/level pairs, while t represents the size of the time dimension. The layer names correspond to the names in the Keras library. DLWP = Deep Learning Weather Prediction; LSTM = long short-term memory.
[a]Output shape is (channels, y, x). [b]Approximate number of learned parameters for $t = 2$, $v = 1$. [c]Only the LSTM variants include the ConvLSTM2D layer. [d]The ConvLSTM2D layer has a separate time dimension which is subsequently reshaped into $4vt$ output channels. For variants with the ConvLSTM2D layer the inputs and outputs are also reshaped to (t, v, 36, 144).

Figure 1. Original CNN architecture from (Weyn, 2019).

3.4. Training procedure

We have modified the WeatherBench data loader to load only 3 years at a time for reasons described in section 4.1. For this reason, rather than re-computing the means and standard deviations for the purposes of normalization many times throughout the training, we pre-computed both and then loaded them from saved files. Data for years 1979 to 2015 were used for training, year 2016 for validation, and years 2017 and 2018 for testing. Each experiment ran for 100 epochs and we used early stopping with patience of 50 epochs.

3.5. Geopotential and temperature

The primary variables we chose for prediction and model evaluation were geopotential at 500 hPa pressure level (further referred to as Z500), which corresponds to about 5.5km above sea level, and temperature at 850 hPa pressure level (further referred to as T850), which corresponds to about 1.5 km above sea level. These two variables are often used in similar applications for various reasons, such as because the temperature at 850 hPa is not affected by the surface (N., 2015). Our primary reason for using specifically these variables was so that we could compare our models to the WeatherBench baselines by Rasp et al., as well as other publications using the same or similar variables, such as (Weyn, 2019) or (Düben & Bauer, 2018).

4. Experiments

In order to answer our research question, we designed a number of experiments. Each experiment used the WeatherBench framework designed by Rasp et al. as its base and then made several adjustments, in order to identify the effects of these adjustments on the performance measure. These varied from using more or different data, through varying the configuration of the network and adding new elements such as an LSTM block, to using a different network architecture altogether and only using the WeatherBench framework to evaluate our model.

4.1. Modified baseline

The WeatherBench framework implements a data loader which loads the entirety of the selected dataset (40 years) into memory at once. This approach works fine for 5.625° resolution and using a single level of each geopotential and temperature, which requires only about 6 GB of RAM. However, all pressure levels for temperature alone require about 25 GB of RAM, for instance, which poses some problems with running the experiments.

Firstly, we were advised not to request more than 12 GB of RAM for the MLP cluster jobs so we would not hog up resources for other groups and be able to move up in the queue. This was important because it often took us 1-2 days to get a job running even with 12 GB RAM, let alone with a much higher number. It also meant we had to resort to using Google Cloud for some of our experiments, where more RAM meant more credit cost. Lastly, we were originally imagining experiments with datasets close to 60 GB in size, making it further unlikely we would ever get in queue on the cluster and raising the Google Cloud costs significantly.

Our solution was to modify the data loader to only load 3 years at a time instead of 40, resulting in significantly lower RAM usage. This meant we could comfortably run our experiments on both the MLP cluster and Google Cloud. However, it also meant that since we were loading the data differently from the original baseline, this could have adverse effects on the model's performance, which would invalidate the original baseline provided by WeatherBench.

We modified the original baseline by making it use our data loader instead of the original one and re-ran the experiment with no further changes. As shown in table 1, the largest RMSE difference between the original and modified baselines was mere 3.48% for temperature at 850 hPa. Therefore, while the resulting model did not have exactly the same performance as the original baseline, it was very close and we could continue with experimenting. All further experiments used our modified data loader for consistency.

	Z500 [m^2 s^{-2}]	**T850 [K]**
Original baseline	626.40	2.87
Modified baseline	641.12	2.97
RMSE Difference	+2.35%	+3.48%

Table 1. Comparison of 3-day forecast baseline RMSE for geopotential at 500 hPa (Z500) and temperature at 850 hPa (T850) between original and modified baselines.

4.2. CNN for isolated variables with all pressure levels

4.2.1. Motivation

We were initially surprised that Rasp et al. downloaded and pre-processed 11 pressure levels of geopotential and temperature data without using all of them. Seeing as we had this data available, it seemed like a waste not to use

them, especially since combining a lot of data for training did not seem to be popular amongst the related works we have analysed and we wanted to see why. But despite the fact that most have focused on single-level geopotential, sometimes along with single-level temperature, some have used different combinations of data. For instance, Weyn used 500 hPa geopotential in combination with 700-300 hPa thickness in order to better predict the amplification and decay of weather systems.

Our hypothesis for using all 11 available pressure levels for geopotential and temperature, respectively, was that the dynamics of individual levels depend on each other to an extent, hence the knowledge of the dynamics of all pressure levels could give us better understanding of Z500 and T850.

4.2.2. Description

We built two models, one for each variable, and trained them on all the pressure levels available for each variable. Each model was then able to predict values for all supplied pressure levels but was evaluated only on Z500 and T850.

	Z500 [m^2 s^{-2}]	**T850 [K]**
Baseline	641.12	2.97
Isolated models	621.61	3.02
RMSE Difference	−3.04%	+1.68%

Table 2. Comparison of 3-day forecast RMSE for isolated models with all pressure levels of their respective variable compared to the baseline.

4.2.3. Interpretation and discussion

Unfortunately, as we see in table 2, the isolated-variable models did not perform significantly better than the baseline but instead managed to perform worse in the case of temperature prediction. Furthermore, because each model used about 10 times more data than the baseline, each training and validation step took about 10 times as long, resulting in a terrible performance-cost ratio. It is therefore likely that the models were not able to capture the related dynamics of all pressure levels or that they are simply not significantly pronounced hence each level is independent of the others.

The only upside of using these two models was that each model is able to predict not only Z500 and T850, respectively, but all pressure levels for their respective variable. This, however, is not particularly useful simply because most of the other pressure levels are not very useful.

4.3. CNN with ConvLSTM

4.3.1. Motivation

As explained in-depth in section 3.2, one approach to improve the performance of the forecast network is to add a ConvLSTM layer to capture spatiotemporal correlations present in our dataset.

4.3.2. Description

We trained a new model by adding a ConvLSTM layer at the start of the baseline CNN architecture, and changed the input from 4D to 5D by adding a time step dimension. Since the output from ConvLSTM is 4D no additional transformation is required. Our model used time steps size of 8 for each sample, meaning 8 hours of continues images were fed as input with 3 day lead time forecast being the target.

	Z500 [m^2 s^{-2}]	**T850 [K]**
CNN Baseline	641.12	2.97
ConvLSTM model	556.77	2.73
RMSE Difference	−13.16%	−9.19%

Table 3. Comparison of 3-day forecast RMSE for geopotential at 500 hPa (Z500) and temperature at 850 hPa (T850) between CNN baseline and ConvLSTM model.

4.3.3. Interpretation and discussion

Our reasoning seems to have been correct as just by adding a single layer at the start of the network we were able to improve the Z500 baseline by 13% and T850 by 9%. These results are in-line with (Weyn, 2019), where their best performing model included a ConvLSTM layer, albeit their CNN structure was different.

A notable downside to this approach, however, is the model training and validation time, which increased significantly when using an LSTM layer. While a validation step in the CNN-only baseline (section 4.1) took only about 74 seconds, with the added LSTM layer, this increased to around 717 seconds. Although this is almost a 10-fold increase in time, we would argue that unless minimizing the training time is of the essence, the noticeable improvement that comes with using LSTM is worth the extra cost.

4.4. Varying ConvLSTM network step size with constant amount of data

4.4.1. Motivation

We wanted to discover whether the performance of the model improves when the time step data fed to the model for training is not continuous (continuous meaning every hour from 1pm to 8pm) but rather non-continuous, with gaps between each step (e.g. first data point at 1pm, second at 5pm, third at 9pm and so on). Our reasoning to conduct this experiment was that having a larger step size could lead to the trained network being able to capture the underlying processes better as each sample would have a longer period from which the data is collected, but the overall amount of data would be the same.

4.4.2. Description

We compared two models, both using the same LSTM architecture described in 3.2. The continuous model was trained using 8 continuous time steps as input with a 3 day lead time forecast as the target (experiment 4.3). The non-

continuous model was also trained using 8 data points, but the time gap between time steps was 4 hours instead of just 1 hour, with the target being the same as in the first model.

	Z500 [m^2 s^{-2}]	T850 [K]
ConvLSTM Baseline	556.77	2.73
Non-continuous model	702.02	3.16
RMSE Difference	+26.09%	+15.75%

Table 4. Comparison of 3-day forecast RMSE for geopotential at 500 hPa (Z500) and temperature at 850 hPa (T850) between models with continuous time steps and non-continuous time steps.

4.4.3. Interpretation and discussion

The non-continuous model performed significantly worse, even worse than the CNN baseline, meaning the model was not able to learn the underlying patterns in our dataset. We think that this is due to the fact that the atmosphere can sometimes significantly change even in the span of 1 hour. This means that only providing information about it in 4 hour intervals is not sufficient to be able to capture the trends and processes that are happening.

4.5. Varying the depth and configuration of the LSTM model

4.5.1. Motivation

Although not always true, a deeper model with a similar configuration will usually give better results as long as problems like vanishing gradients are dealt with. In this experiment, we wanted to test if this holds true for our specific scenario too. Since a deeper model means longer training due to a higher number of learnable parameters, we also wanted to find out what the trade-off between performance and cost would be in our case, and if training a very deep model would be worth it or not.

4.5.2. Description

We have used 3 models for this experiment: the baseline ConvLSTM model from experiment 4.3, and two modified versions of it. The only thing we varied was the number of convolution layers within each model, which was 5 for the baseline and 7 and 12 for the deeper models.

Model (# of Layers)	Z500 [m^2 s^{-2}]	T850 [K]
ConvLSTM Baseline	556.77	2.73
ConvLSTM (7)	550.99	2.68
ConvLSTM (12)	521.57	2.58
RMSE Difference 7	−1.04%	−1.93%
RMSE Difference 12	−6.33%	−5.50%

Table 5. Comparison of 3-day forecast RMSE for geopotential at 500 hPa (Z500) and temperature at 850 hPa (T850) between 3 ConvLSTM models each with increasing depth.

4.5.3. Interpretation and discussion

Table 5 shows us that depth does indeed matter seeing as the model with 12 convolution layers outperformed the baseline by about 6% in both Z500 and T850. The mid-sized model with 7 layers showed a minor improvement as well, but this was relatively marginal and insignificant.

Model (# of Layers)	# of Parameters	Time [s/step]
ConvLSTM Baseline	835,714	717
ConvLSTM (7)	1,040,642	744
ConvLSTM (12)	1,552,962	820
Difference 7	+24.52%	+3.77%
Difference 12	+85.83%	+14.37%

Table 6. Comparison of the number of parameters and the average time per validation step between 3 ConvLSTM models each with increasing depth.

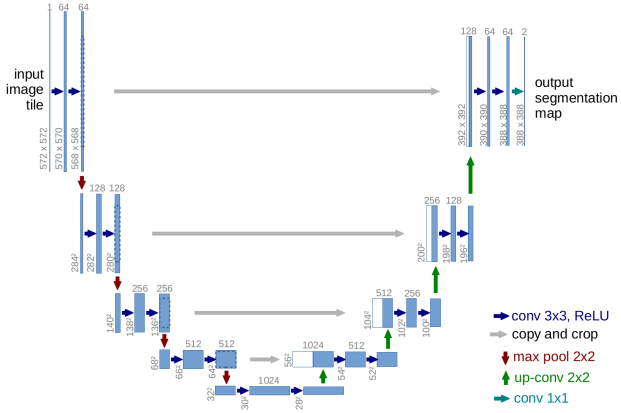

Figure 2. Original U-Net architecture from (Ronneberger et al., 2015). The architecture of (Larraondo et al., 2019) instead uses Conv2D with kernel size 3x3 layers of filter sizes 32, 32, 64, 64, 128, 128, 256, 256, 512, 512 in the down sampling part.

In terms of the performance-cost ratio, the largest model took only about 14% longer per step on average compared to the baseline despite its significantly larger number of learnable parameters, as shown in table 6. While these times are not entirely representative of the whole training time and they should be taken with a grain of salt, they show that the increase in time cost is not significant enough to ignore the improvement in performance. This is especially the case when we compare these time differences to the nearly 10-fold increase mentioned in section 4.3 upon introducing an LSTM layer. In summary, using a deeper network proved to give improved performance with a relatively insignificant increase in costs.

4.6. U-Net

4.6.1. Motivation

(Larraondo et al., 2019) compared three different auto-encoders models at precipitation prediction and showed that the U-Net (Ronneberger et al., 2015) architecture performed the best. We have decided to test their findings and use their model architecture, specifically U-Net and VGG-16, on our task.

4.6.2. Description

The U-Net (Ronneberger et al., 2015) architecture is encoder-decoder type of network. The encoders network learn compressed representation of the input space and how to decompress it. For weather prediction we hope that the model learns dynamics of the weather. We use almost identical version of U-Net to (Larraondo et al., 2019) with the exception of using root mean square error instead of mean absolute error for the loss function.

The original model used three layers of geopotential to predict likelihood of precipitation. We, on the other hand, use one pressure level of geopotential and one pressure level of temperature to predict 3 days forecast of these levels.

	Z500 [m^2 s^{-2}]	**T850 [K]**
CNN Baseline	641.12	2.97
U-Net	689.69	3.10
RMSE Difference	+7.5%	+4.32%

Table 7. Comparison of 3-day forecast RMSE for geopotential at 500 hPa (Z500) and temperature at 850 hPa (T850) between the baseline and U-Net.

4.6.3. Interpretation and discussion

Training this model proved to be challenging because initially the loss function during training exploded to infinity. However, restarting the training with a different random seed allowed us to circumvent this problem.

The model was not able to outperform our CNN baseline but it did have very similar performance without any architecture tweaking. It is possible that, for example, with deeper architecture it would be possible to achieve better results. This architecture was, however, already 26 times bigger than our baseline in terms of network parameters.

(Weyn et al., 2020)[6] showed that is possible to use a U-Net-like architecture to outperform the CNN baseline and even the IFS T42 numerical weather model. Notably, their U-Net architecture was much smaller than the one that we have used.

4.7. U-Net with ConvLSTM

4.7.1. Motivation

Goal of this experiment was to verify whether adding Convolutional LSTM layer will improve the performance of the network.

(Azad et al., 2019) showed that their variant, "Bi-Directional ConvLSTM U-Net with Densley Connected Convolutions" (BCDU-Net) can outperform U-Net on certain tasks. We have tried using the BCDU-Net network on our dataset but this had resulted in very high levels of RMSE both for Z500 (2434.49, that is 3.80 times higher than the CNN baseline) and T850 (10.64, 3.58 times higher than the baseline) for 72 hours forecast.

[6]This paper was published on March 30, after we have trained our model.

Therefore we have decided to extend the U-Net architecture by adding a convolutional LSTM 2D layer at the beginning of the model. In previous experiments we have seen that this layer increased performance of the fully convolutional baseline therefore our hypothesis is that this approach will work for U-Net as well.

4.7.2. Description

We add ConvLSTM layer at the start of the U-Net model, otherwise the experiment design remains identical to previous U-Net experiment.

	Z500 [m^2 s^{-2}]	**T850 [K]**
U-Net	689.69	3.10
U-Net ConvLSTM	685.50	3.10
RMSE Difference	−0.6%	0%

Table 8. Comparison of 3-day forecast RMSE for geopotential at 500 hPa (Z500) and temperature at 850 hPa (T850) between the U-Net and U-Net with LSTM.

4.7.3. Interpretation and discussion

The RMSE for T850 is identical to previous U-Net results and in Z500, there was only a marginal decrease from 689.70 to 685.50. Therefore, there was no significant improvement compared to pure U-Net. On the contrary, this model was 7 times slower to train, so it did not prove to be useful for our application.

4.8. VGG-16

4.8.1. Motivation

VGG-16 model is a well recognised model for image recognition (Simonyan & Zisserman, 2015). It was the second model (Larraondo et al., 2019) used for precipitation prediction In spite of the results in that work showing that VGG-16 was outperformed by U-Net, we have decided to test this model on our task.

4.8.2. Description

VGG-16 and other auto-encoders learn compressed representation of data and then use deconvolution layers to expand this representation back into original data (Hinton & Salakhutdinov, 2006).

In our task we use auto-encoder design not to learn identity of the input but rather to learn a future state. This is a harder task because it forces the network to learn the representation and the transformation of the data at the same time.

	Z500 [m^2 s^{-2}]	**T850 [K]**
CNN Baseline	641.12	2.97
VGG-16	3130.57	14.11
RMSE Difference	+388.3%	375.1%

Table 9. Comparison of 3-day forecast RMSE for geopotential at 500 hPa (Z500) and temperature at 850 hPa (T850) between the baseline and VGG-16.

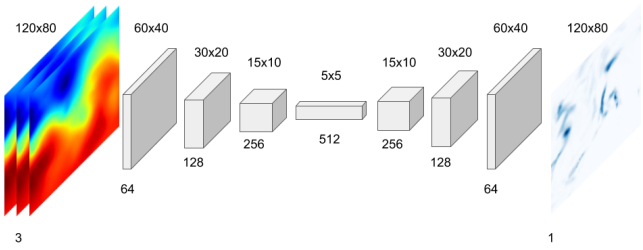

Figure 3. VGG-16 architecture from (Larraondo et al., 2019). The network has to learn a compressed representation of input that fits the mid-block.

4.8.3. Interpretation and discussion

Clearly this model was not able to capture the dynamics of our data. This is interesting because both models are use in (Larraondo et al., 2019), where VGG-16 performed worse than U-Net but not by such a huge margin.

However, as stated before, our task and data were different from (Larraondo et al., 2019). VGG-16 model works by learning reduced representation of state and then upsampling this representation to its original shape. This method might not be optimal for our task. We speculate that U-Net performs better because it uses the copy operations to preserve the identity of the previous layers. Thus, the spatial reconstruction might be easier for U-Net.

5. Related work

State of the art methods for weather prediction use numerical methods, so there have only been a couple of papers that try to directly use machine learning for weather prediction. We suspect this might be because numerical methods are highly tuned and optimized and thus hard to out-perform.

There are some works that try to combine machine learning with numerical models. (Rodrigues et al., 2018) uses convolutional neural networks to upsample low resolution output of numerical methods into high resolution predictions. (Scher & Messori, 2018) used CNN to predict uncertainty of the forecasts.

(Scher, 2018) showed that CNN networks can learn dynamics of general circulation models (GCM) that together with numerical models are widely used for weather prediction. The input of their network was the state of GCM and their output was the next state of GCM. This work suggest that CNNs have the ability to process the dynamics of complex weather systems.

In (Weyn, 2019) they compared basic CNN networks with and without using LSTM layers to the state of the art physical model known as Climate Forecast System. Their results show that including an LSTM layer improved the performance when compared to a basic CNN model. However, they were still largely lacking when it came to comparing with CFS, a baseline for NWP models.

(Mehrkanoon, 2019) compared a basic LSTM model to a 1D and 2D CNN based model on the task of predicting temperature from 1 to 10 days in the future. Contrary to (Weyn, 2019), the non-LSTM architecture came out on top, with the 2D network achieving the best performance, where the dimensionality was increased by convolving learned kernels over additional stations, allowing for information sharing between the neurons responsible for every station.

A similar research question to ours was also asked and discussed by (Düben & Bauer, 2018), who go deeper into why deep learning could and could not be used for weather prediction. They are also one of the several papers to use very similar data to ours (ERA5, geopotential at 500 hPa) and arrive at a similar conclusion that NWP models simply still out-perform machine learning solutions.

(Larraondo et al., 2019) was an inspiration for two of the models that we have used. Their task was, however, quite different from ours - they model the relation between geopotential and precipitation rather than forecasting.

(Weyn et al., 2020) have built the currently best-performing model on the WeatherBench benchmark – their model managed to outperform a scaled down version of an NWP model. Compared to our version of U-net they use a shallower version. Moreover, they use cubic convolution that are better suited for 3D data.

6. Conclusions

	Z500 RMSE [$m^2\ s^{-2}$]	T850 RMSE [K]
VGG-16	3131	14.11
Climatology	1075	5.51
Persistence	936	4.23
Weekly Climatology	816	3.50
Linear Regression	714	3.19
Non-continuous ConvLSTM	702	3.16
U-Net	690	3.10
U-Net ConvLSTM	686	3.10
CNN Baseline	641	2.97
CNN Isolated all levels	622	3.02
ConvLSTM Baseline	556	2.72
ConvLSTM 7 Layers	551	2.68
ConvLSTM 12 Layers	**522**	**2.58**
IFS T42	489	3.09
IFS T63	268	1.85
Operational IFS	**154**	**1.36**

Table 10. Comparison of all baselines and trained models for 3 days forecast time at 5.625° resolution. Models are ordered by performance and best machine learning and physical models are in bold.

Table 10 summarizes the results from all our experiments as well as previously established baselines from (Rasp et al., 2020). While the individual results of each experiment were discussed in section 4, here we can see how our machine learning models compare across the board to the industry NWP models.

Starting from the CNN baseline, our improvements were achieved by adding an initial ConvLSTM layer as well as increasing the amount of convolutional layers in the network, with the best network (ConvLSTM 12 Layer) achieving a 17% lower RMSE on the geopotential Z500

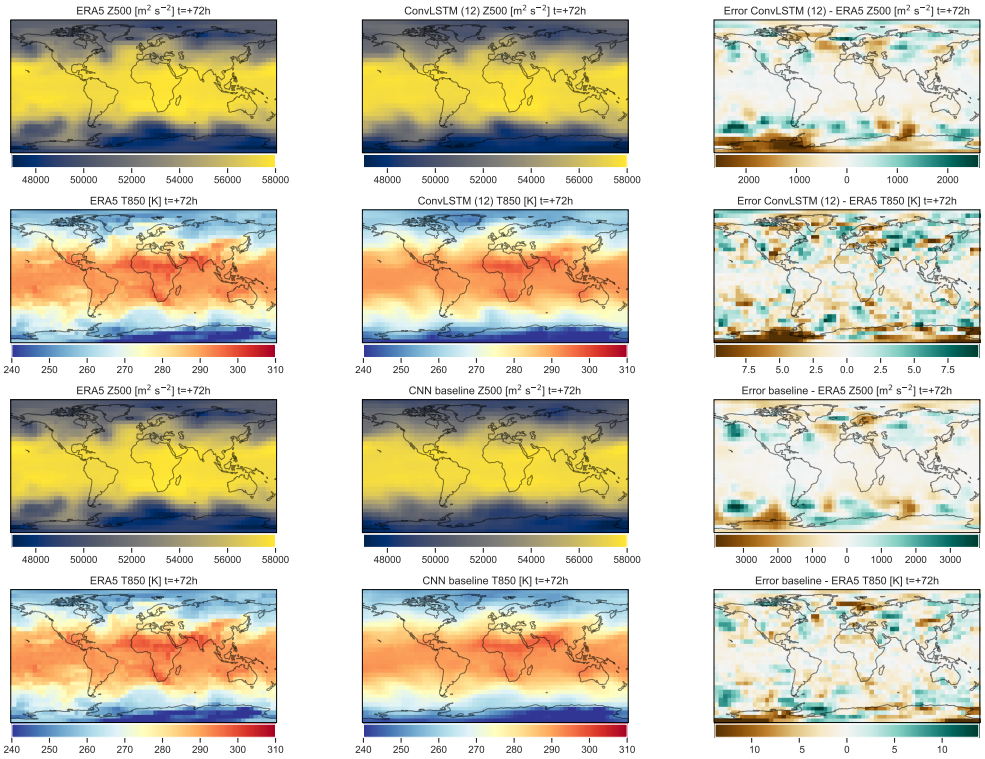

Figure 4. Visualization of sample prediction made by ConvLSTM (12) at 3 day lead time. We can see that model managed to learn general pattern of the data, however it is much smoother than ground truth.

data and an 11% improvement on the temperature T850 data.

Figure 4 shows the improvements of the best performing model compared to the baseline. Specifically results of CNN baseline in the bottom 2 rows were much "smoother" than the ConvLSTM ones in the top 2 rows, indicating the latter model was able to learn the spatiotemporal correlations at a much finer level, thereby achieving a better performance.

We have also found that training on all data levels (from 1 hPa to 1000 hPa) provided no apparent benefit (but increased training time tenfold), which was surprising result, as we initially hoped that providing neighboring levels would improve the model accuracy. Even without any direct correlations we hoped that more data independent data points would enable the network to learn better. But from the results it is clear that there was no benefit.

Experimenting with gaps between time-steps also showed that for a 3 day prediction task, providing spread out data over 32 hours was no better than using 8 consecutive hours, indicating that per hour changes are important for the performance of predicting this short time frame.

The VGG-16 architecture was not able to capture any of the dynamics present in our data and was by far the worst performing model. U-Net showed promise, achieving similar performance to the CNN baseline with the default architecture, indicating a more fine tuned network would be able to achieve good performance. This was confirmed in the very recently published paper (Weyn et al., 2020), that used an altered U-Net with a cubed sphere representation achieving results better than IFS T42 NWP.

While our best network came closer to the IFS T42 benchmark, it was still very far in terms of performance compared to the operational IFS model. This confirms that deep learning weather prediction is not yet able to compete with the numerical weather prediction methods that have been developed and improved for over half a century. However, with further research is done in the field and more fine tuned models and architectures are created we hope to see machine learning models that are able to achieve comparable performance to its numerical counterparts.

References

Azad, Reza, Asadi, Maryam, Fathy, Mahmood, and Escalera, Sergio. Bi-directional convlstm u-net with densley connected convolutions. 08 2019.

Düben, Peter and Bauer, Peter. Challenges and design choices for global weather and climate models based on machine learning. *Geoscientific Model Development*, 11: 3999–4009, 10 2018. doi: 10.5194/gmd-11-3999-2018. URL https://doi.org/10.5194/gmd-11-3999-2018.

Gers, Felix, Schmidhuber, Jürgen, and Cummins, Fred. Learning to forget: Continual prediction with lstm. *Neural computation*, 12:2451–71, 10 2000. doi: 10.1162/089976600300015015.

Hinton, G E and Salakhutdinov, R R. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006. doi: 10.1126/science.1127647. URL http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&uid=16873662&cmd=showdetailview&indexed=google.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. Neural Computation, 1997. URL https://www.bioinf.jku.at/publications/older/2604.pdf.

Larraondo, Pablo, Renzullo, Luigi, Inza, Inaki, and Lozano, Jose. A data-driven approach to precipitation parameterizations using convolutional encoder-decoder neural networks. 03 2019.

LeCun, Y. Deep learning. Nature, 2015. URL https://www.nature.com/articles/nature14539.pdf.

Mehrkanoon, Siamak. Deep shared representation learning for weather elements forecasting. *Knowledge-Based Systems*, 179:120 – 128, 2019. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2019.05.009. URL http://www.sciencedirect.com/science/article/pii/S0950705119302151.

N., MAIER. Variations of air temperature at 850 hpa in north-west romania. *Aerul şi Apa: Componente ale Mediului*, 2015, 03 2015. doi: 10.17378/AWC2015_25. URL https://doi.org/10.17378/AWC2015_25.

Rasp, Stephan, Dueben, Peter D., Scher, Sebastian, Weyn, Jonathan A., Mouatadid, Soukayna, and Thuerey, Nils. WeatherBench: A benchmark dataset for data-driven weather forecasting. *arXiv e-prints*, art. arXiv:2002.00469, Feb 2020.

Richardson, L.F. Weather prediction by numerical process. Cambridge, The University press, 1922. URL https://archive.org/details/weatherpredictio00richrich/page/n7/mode/2up.

Rodrigues, Eduardo, Oliveira, Igor, Cunha, Renato, and Netto, Marco. Deepdownscale: a deep learning strategy for high-resolution weather forecast. 08 2018.

Ronneberger, O., P.Fischer, and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pp. 234–241. Springer, 2015. URL http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a. (available on arXiv:1505.04597 [cs.CV]).

Scher, S. Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning. *Geophysical Research Letters*, 45 (22):12,616–12,622, 2018. doi: 10.1029/2018GL080704. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018GL080704.

Scher, Sebastian and Messori, Gabriele. Predicting weather forecast uncertainty with machine learning. *Quarterly Journal of the Royal Meteorological Society*, 144(717): 2830–2841, 2018. doi: 10.1002/qj.3410. URL https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.3410.

Shi, X., Chen, Z., Wang, H., and Yeung, D-Y. Convolutional lstm network: A machine learning approach for precipitation nowcasting. arXiv, 2015. URL https://arxiv.org/pdf/1506.04214v2.pdf.

Simonyan, Karen and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Stern., H. and Davidson, N. E. Trends in the skill of weather prediction at lead times of 1–14 days. Quarterly Journal of the Royal Meteorogical Society, 2015. URL https://rmets.onlinelibrary.wiley.com/doi/epdf/10.1002/qj.2559.

Weyn, J.A. Can machines learn to predict weather? using deep learning to predict gridded 500-hpa geopotential height from historical weather data. Journal of Advances in Modeling Earth Systems, 2019. URL https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2019MS001705.

Weyn, Jonathan A, Durran, Dale Richard, and Caruana, Rich. Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. *Earth and Space Science Open Archive*, pp. 30, 2020. doi: 10.1002/essoar.10502543.1. URL https://doi.org/10.1002/essoar.10502543.1.