Predicting Weather Temperature using Convolutional Neural Networks on Tabular Data: Final Report

G002 (s1813106, s1754795, s1865890)

Abstract

This paper evaluates how a Kaggle-inspired 1-dimensional Convolutional Neural Network (CNN) performs on heterogeneous tabular weather prediction, compared to a conventional model for such task and a stripped-down CNN and a fully-connected one network. We apply some pre-processing to normalise and decorrelate the feature data. Experiments show that the CNN is comparable to the mainstream models but does not perform as well as the other mainstream models. We conclude that it is a viable alternative and could be further explored, although other alternatives, such as TabNet, may be worth considering.

1. Introduction

Predicting weather has been of interest for a long time. For example, studies on predicting temperature have been done in Turkey (Ömer Altan Dombaycı & Gölcü, 2009) and in Georgia(Smith et al., 2006). These are aimed at forecasting temperature ahead of time in specific locations, using available historical weather data from there to feed into artificial neural networks. Other papers modelled data to predict heavy rain, hail, convective gusts, and thunderstorms (Zhou et al., 2019), showing that deep learning methods fit the task.

The focus so far has mainly been on treating weather forecasting as a spatio-temporal problem. Various deep learning architectures have been proposed, with most of them showing good performance in the short-term but not in the long-term (Ren et al., 2021). One reason for this is that the global average temperature on earth has been continuously increasing in the past years, with changes in climate properties taking a toll on vegetation, fauna and, ultimately, society and its routine functioning. In addition, forecasting temperature at locations where histories of weather data are hardly available has been challenging and not rigorously addressed due to data not being openly available previously.

This paper aims to exclude spatial and temporal references and, instead, use meteorological and climate properties like sun elevation and atmospheric pressure to predict air temperature. While this means that the task becomes more of a regression one, it also increases model robustness and capability to adapt to distributional shifts (Malinin et al., 2021).

Another aim, which this paper has, is to test deep-learning techniques on a heterogeneous tabular data set. Even though deep learning has been shown to work well on homogeneous data, using it to model heterogeneous data remains a challenge. Moreover, since heterogeneous data is the most common type of data at the moment, finding good models that work well with it is of utmost importance. Convolutional Neural Networks are a good candidate, given their recent successes in other fields such as computer vision or natural language processing. As such, this paper looks at the following research questions:

- How do deep learning techniques, particularly Convolutional Neural Networks, perform on a weather prediction task that uses heterogeneous tabular data?
- Do Convolutional Neural Networks respond well to time and spatial shifts for weather prediction?

It was found that CNN did not perform better than CatBoost overall. However, the difference between them is slight, and this seems promising. On the other hand, CNN was robust because it performed well on climate types not present in the training set-even coming close to the gradient-boosted DT model.

2. Data set and task

2.1. Heterogeneous tabular data

Our data (Dat) can be characterized as heterogeneous and tabular. Heterogeneous means that it comes from various sources (e.g. measurements of different physical quantities, output from weather prediction models and more). Heterogenous data contrasts with homogeneous data, which comes from a single source. Tabular data means it can be expressed as a table. One fundamental property of tabular data is that there is little to no correlation between consecutive rows, unlike, for example, time-series data or indexed data.

2.2. Data set

We will use the data set provided by Yandex. It features the weather data during the Yandex shifts challenge (Malinin et al., 2021), which took place in October-November 2021. It features tabular data, in which one row shows the measured temperature together with other climate and meteorological features at a particular time in a precise location, situated in a specific type of climate, as mapped by the Koppen climate classification (Chen & Chen, 2013).

The task at hand is to build a model that accurately finds the temperature at a specific location at a specific time, given a set of features characterizing that location at that time (e.g. pressure, humidity etc.). The model needs to work well regardless of time and location.

Figure 1 shows some of the 129 columns that are in the data set. The 123 columns represent helpful features for our task. 4 columns are metadata, giving information about the location (longitude, latitude), climate type and timestamp. These were disregarded for our computation. There were two tasks on this dataset at the contest where it first appeared: regression and classification tasks. Out of the remaining two columns, "fact_cwsm_class" was intended for the classification task, which we did not take on; hence it was also not used. The remaining column, "fact_temperature", was used as the target data.

The dataset was partitioned into training, development and evaluation sets. There are about 3 million data points in the training partition, 100 000 in development and about 1 million in the evaluation partition. The development and evaluation partitions were further split in half into in-domain, and out-of-domain data, that is, data from climates and regions that appear in the training set and data from climates and regions that do not. Figures 2 and 3 show more details about the different partitions of the data set.

The goal set by the organisers was for teams to construct a robust model that would respond well to distributional time and spatial shifts.

3. Methodology

3.1. Pre-Processing

Some pre-processing is applied to feature data:

- Imputing: The missing values are replaced in all input columns following a simple constant strategy (fill value is -1).
- Quantisation: Each input column is discretised into 100 quantile bins; then, the bin identifier can be considered a quantised numerical value of the original feature; The result is a normal distribution.
- Standardisation: Each quantised column is standardised by removing the mean and scaling to unit variance.
- PCA: Principal Component Analysis is applied and 95% of the variance is retained. These are concatenated to the original data to help with grouping features together.

Preprocessing was necessary at least in order to deal with missing data, but most of it was done based on a few assumptions and with inspiration from (Bondarenko, 2021), the paper written by the competition's winner. The first assumption is that, even if data come from entirely different sources, the number of features that result and are significantly different from one another will not be significant. Moreover, proof of this is that 22 components retained 95% of the variance of 123 features (see figure 4). The second assumption is that, despite being collected from all corners of the world, the distribution of most samples will be similar.

3.2. Model Description

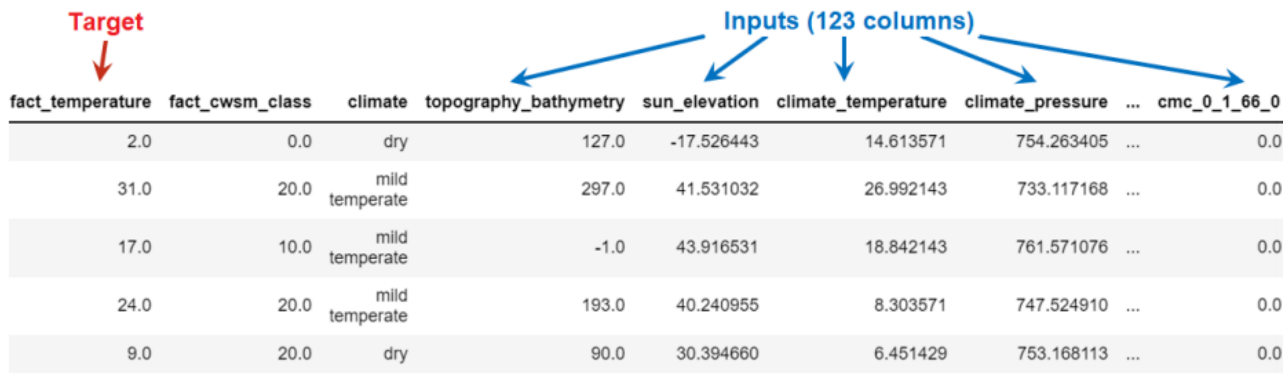
CNN is a deep neural network that mimics how the visual cortex of the brain processes and recognises images. It is well renowned for its prowess in image recognition. A CNN thrives on spatial correlations, which the neighbouring pixels in images have plenty of. That being said, tabular data is nothing like images. This section provides a brief introduction to the architecture of our CNN, inspired by a Kaggle solution (Medium) which won second place at a competition using tabular data. Figure 5 shows an overview of it.

Our model is a CNN that creates images from non-image data. The idea with tabular data is that we could create space for features in a fully connected layer since it is hard to detect spatial correlation. The fully connected layer is learning possible nonlinear functions in that space. Adding such a layer is a (usually) inexpensive way to learn nonlinear combinations of advanced features. After the first dense layer, we regroup the features in 16x1 feature maps, which we feed into the next layer, which is convolutional.

Convolutional layers contain filters that convert images, which are called convolutional. Putting the image through a convolutional layer will produce a feature map. The trained convolutional filter determines the features extracted in the convolutional layer. Therefore, the characteristics of convolutional layer extraction will vary depending on the convolutional filter used. An activation function processes the feature map created by the convolutional filter before the layer produces an output. We use four convolutional filters that adopt batch normalisation and residual connections to help with vanishing gradients. They use the ReLU activation function, which is of the form:

$$ReLU(x) = max(0, x)$$

Like convolutional layers, pooling layers are responsible for reducing the amount of space for convolutional features. This reduces the computing power required to process data by reducing dimensionality. In addition, it helps extract the main features of rotation and position invariance, thus maintaining the process of adequate training of the model. There are two types of Pooling: Max Pooling (Murray & Perronnin, 2014) and Average Pooling (Wang et al., 2017). Our model uses one Average-Pooling layer and one Max-Pooling.


Target
Inputs (123 columns)

fact_temperature	fact_cwsm_class	climate	topography_bathymetry	sun_elevation	climate_temperature	climate_pressure	...	cmc_0_1_66_0
2.0	0.0	dry	127.0	-17.526443	14.613571	754.263405	...	0.0
31.0	20.0	mild temperate	297.0	41.531032	26.992143	733.117168	...	0.0
17.0	10.0	mild temperate	-1.0	43.916531	18.842143	761.571076	...	0.0
24.0	20.0	mild temperate	193.0	40.240955	8.303571	747.524910	...	0.0
9.0	20.0	dry	90.0	30.394660	6.451429	753.168113	...	0.0

Figure 1. Table configuration. (Bondarenko, 2021)

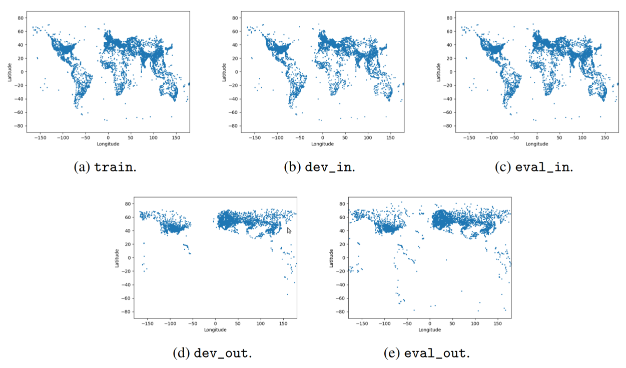
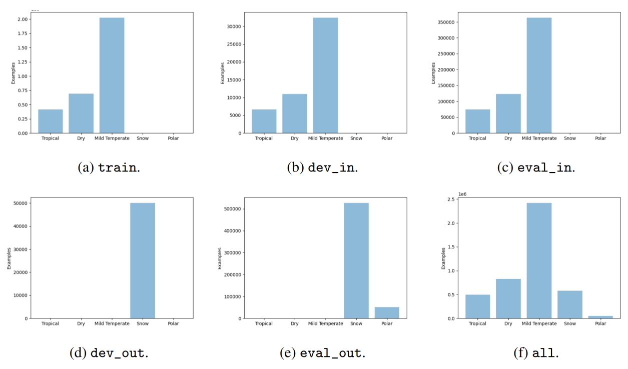

(a) train. (b) dev_in. (c) eval_in. (d) dev_out. (e) eval_out.

Figure 2. Geographical distribution of data in different partitions of the data set. (Malinin et al., 2021)



(a) train. (b) dev_in. (c) eval_in. (d) dev_out. (e) eval_out. (f) all.

Figure 3. Climate categories of data in different partitions of the data set. (Malinin et al., 2021)

3.3. Evaluation Metrics

Suppose there are n samples in a dataset, for which we have an array of n labels y_{true}, each label corresponding to a sample. After running the algorithm for prediction, we obtain n predicted values, in a vector y_{pred}. Let $e = y_{pred} - y_{true}$ be a vector of differences between predicted value and actual value for each sample. We have, according to (Botchkarev, 2019):

$$MSE = \frac{1}{n} * \sum_{i=1}^{n} e_i^2$$

$$RMSE = \sqrt{MSE}$$

$$MAE = \frac{1}{n} * \sum_{i=1}^{n} |e_i|$$

These are standard metrics for regression tasks, hence why we use them. We take the mean squared error (MSE), root mean squared error (RMSE) and mean absolute error (MAE) of all the data points in the evaluation dataset. The lower the value, the better, which implies that the predicted value is closer to the actual value.

4. Experiments

4.1. Setup

The experiments were designed to show how well our Kaggle-inspired convolutional neural networks predict weather temperature from tabular data compared to a stripped-down convolutional neural net, a fully-connected network and the CatBoost decision tree model provided by the competition's organisers. Note that we used a standard loss function for all deep learning models, the Mean Squared Error (MSE) one, which ensures that our trained model has no outlier predictions with huge errors since the squaring part of the function ensures that outliers are heavily taken into account. The different selected models help us answer our research questions in different ways:

- The CatBoost decision tree model helps us assess the CNN against a mainstream method of dealing with tabular data.

- The stripped-down CNN, composed of 4 convolutional layers and a dense layer to help output the results, with hidden size 512 and learning rate 1e-5, helps us assess whether the architecture of the

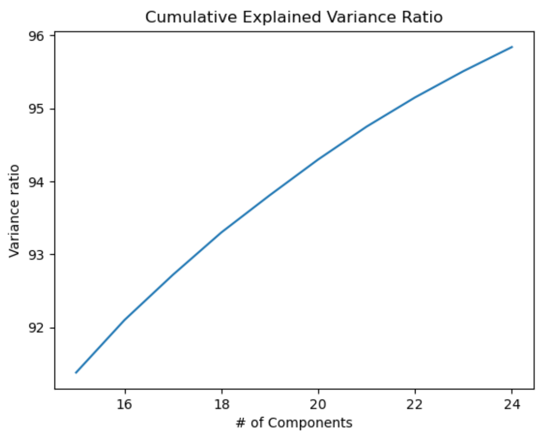


Figure 4. Cumulative explained variance ratio per number of components

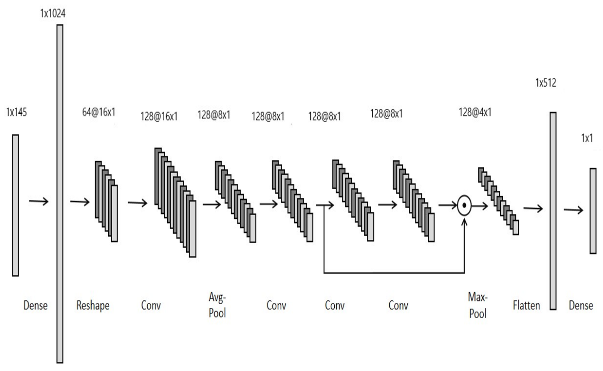


Figure 5. CNN Architecture for hidden layer size of 1024.

CNN significantly improves the results or it is the preprocessing doing.

- The FCN comes in as another deep learning solution. It provides us with the option to assess CNN against other reliable deep learning techniques. Note that this one has batch normalisation applied on all its five dense layers, with a hidden layer size of 512. Without batch normalisation, it suffered from gradient vanishing.

- The rest of the experiments are based on modifying hyper-parameters of the Kaggle-inspired model. We tried different configurations of the hidden layer size and the learning rate. Of course, doubling the hidden layer size meant scaling the dimensions of the other layers as well to keep the feature maps that are to be formed at dimension 16x1.

The starting hyperparameters are inspired by the Kaggle notebook. We firstly trained with a hidden layer size of

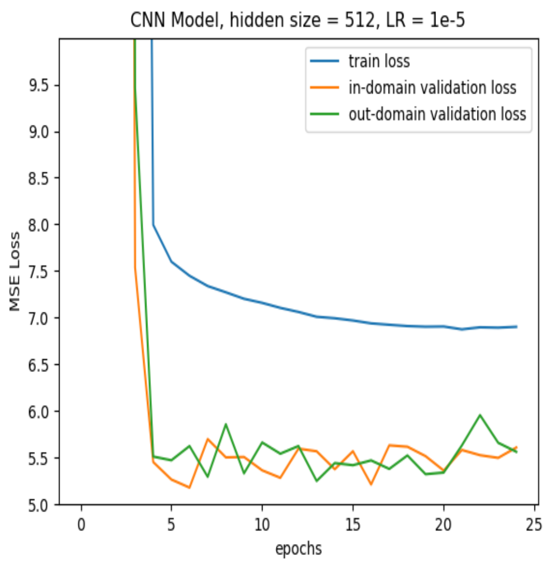


Figure 6. CNN training and validation loss for hidden layer size of value 512 and learning rate of 1e-5.

512 and a learning rate of 1e-3. This model suffered from gradient vanishing, so we lowered the learning rate. At 1e-5, we could notice that the model may underfit the data in that the training loss is bigger than validation loss, both in-domain and out-of-domain. Figure 6 illustrates that. So, we decided to increase once the hidden layer size, once the learning rate and once both, hoping that the results would get a bit tighter to one another. The results got tighter to each other after doing that. Training and validation losses met somewhere mid-way at around loss value 6. Our explanation for the increase in validation loss is that the model started learning the specific features of the training data better, and those did not fit the features in either validation set. At a more in-depth thought, this makes sense only for out-of-domain data because in-domain data is collected in the same climate and in the same time frame as the training data, so it should have similar features. We can generally notice that results for out-of-domain validation data are better than for in-domain validation data. Figures 6 and 7 show that.

4.2. Results & Analysis

Table 1 shows all the results of the experiments we performed. Noticeable is that CatBoost outputs the best results on in-domain data. The baseline CNN also did quite well, out-performing the Kaggle-inspired CNN on in-domain data and only slightly worse for out-of-domain data. This may suggest either that preprocessing contributes a lot more to achieving good results than the tweaks in the architecture or that there is more correlation between consecutive rows in this dataset than usual for tabular data. Either way, the dense layer at the beginning seems to work ineffectively for this particular dataset, not managing to find significant

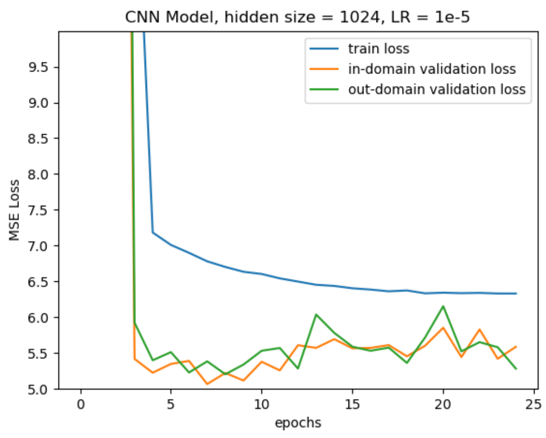


Figure 7. CNN training and validation loss for hidden layer size of value 1024 and learning rate of 1e-5.

relationships between features.

One more exciting thing to note is that the CNN models appear to have a more balanced performance for in-domain and out-of-domain data than CatBoost. Compare the Mean Squared Error, for example, between CatBoost and the CNN models. CatBoost seems to be doing considerably better for in-domain data, whereas, for out-of-domain data, the models perform similarly; some CNNs even did better.

The fully-connected network is the worst model and not even close to being comparable with the other ones. It seems to have learnt the training data a bit too well, and the predictions for evaluation data are off both for in-domain and out-of-domain data.

The results from Table 1 indicate that increasing the hidden layer size and keeping the learning rate at 1e-5 was the best choice, with results improving in all areas compared to other hyper-parameter settings combinations. Still, it is no match for the CatBoost decision trees on in-domain data, but it surpasses it on out-of-domain data.

5. Related work

In November 2021, Yandex held a contest named "Yandex Shifts Challenge", where among the tasks the participants had to tackle, one had to do with building a model for predicting the temperature at a specific location based on a set of other measurements; the classic regression task. The task specified that the model should be able to make accurate predictions even in climates that have not been seen before during training.

The winning solution was published in a paper (Bondarenko, 2021), where it is described how deep learning was used on tabular data to achieve great results even on data outside the domain of the training data set. The author also points out that deep learning is gaining popularity and has proven helpful in various Machine Learning tasks, such as natural language understanding or automatic speech recognition, but this has not been the case for tabular data, despite the potential they demonstrate.

How well do deep models work on such tasks compared to more conventional methods? Some papers claim that deep models can match and even outperform state-of-the-art decision trees. One such particular case is Google's TabNet (Arık & Pfister, 2021), a novel deep architecture for tabular data that utilises sequential attention. The authors claim that TabNet outperformed various gradient-boosted decision trees and other architectures in their tests.

However, not all of the literature is optimistic about the future of deep tabular learning. In (Shwartz-Ziv & Armon, 2022) various deep architectures are tested and compared against XGBoost (Chen & Guestrin, 2016), a tree-ensemble algorithm considered the state-of-the-art (Zhang et al., 2017) for structured data. (Shwartz-Ziv & Armon, 2022) reports that XGBoost outperformed all of the deep models that were put under test. However, an ensemble of deep models and XGBoost was also tested, and it seemed to outperform all the other models.

There is a variety of views on how valuable deep learning can be, especially on whether such models can perform considerably better than the current state of the art. This uncertain situation has inspired us to carry out our experiments and reach our conclusions.

6. Conclusions

In our experiments, the CNN deep learning models did not perform better than CatBoost overall; in fact, most of them performed slightly worse. This justifies both points of view discussed in section 5. On the one hand, these models did not outperform CatBoost, but on the other hand, they came quite close to doing so. It remains unclear whether deep models hold the key to achieving better performances on tabular data.

It is also noteworthy that the CNN models have shown signs of robustness on data from climate types that were not in the training dataset. In particular, they were much closer to the gradient-boosted DT model's performance for out-of-domain data than for in-domain; some models even outperformed it by some metrics. It could be the case that CNNs (or maybe deep-learning architectures in general) have some advantage over Decision trees when it comes to unfamiliar data.

Concerning the task, in particular, the minimum Mean Absolute Error that was achieved was about 2 degrees Celsius. This is not as good as CatBoost did, but it is still not a large error for weather forecasting.

Two questions that remain are whether CNNs can outperform state-of-the-art for tabular data and whether they possess any advantage compared to decision trees when it comes to unfamiliar data.

Model	LR	Hidden	RMSE		MSE		MAE	
			eval_in	eval_out	eval_in	eval_out	eval_in	eval_out
CatBoost Baseline	N/A	N/A	**1.600**	2.629	**2.560**	6.912	**1.185**	**1.925**
Fully-Connected Network	1e-5	512	17.721	17.704	314.037	313.446	15.422	16.344
Baseline CNN	1e-5	512	2.090	2.662	4.369	7.087	1.548	1.942
Kaggle-inspired CNN	1e-5	512	2.408	2.611	5.800	6.818	1.819	2.000
Kaggle-inspired CNN	1e-4	512	2.371	2.598	5.621	6.752	1.764	1.969
Kaggle-inspired CNN	1e-5	1024	2.298	**2.588**	5.279	**6.698**	1.707	1.955
Kaggle-inspired CNN	1e-4	1024	2.451	2.619	6.007	6.858	1.864	2.016

Table 1. Experiment results (RMSE, MSE, MAE) for different combinations of hidden layers size and learning rate.

References

Dataset resource. https://github.com/yandex-research/shifts.

Arık, Sercan O and Pfister, Tomas. Tabnet: Attentive interpretable tabular learning. In *AAAI*, volume 35, pp. 6679–6687, 2021.

Bondarenko, Ivan. More layers! end-to-end regression and uncertainty on tabular data with deep learning. *arXiv preprint arXiv:2112.03566*, 2021.

Botchkarev, Alexei. A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management*, 14:045–076, 2019. doi: 10.28945/4184. URL https://doi.org/10.28945%2F4184.

Chen, Deliang and Chen, Hans Weiteng. Using the köppen classification to quantify climate variation and change: An example for 1901–2010. *Environmental Development*, 6:69–79, 2013.

Chen, Tianqi and Guestrin, Carlos. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145%2F2939672.2939785.

Malinin, Andrey, Band, Neil, Chesnokov, German, Gal, Yarin, Gales, Mark JF, Noskov, Alexey, Ploskonosov, Andrey, Prokhorenkova, Liudmila, Provilkov, Ivan, Raina, Vatsal, et al. Shifts: A dataset of real distributional shift across multiple large-scale tasks. *arXiv preprint arXiv:2107.07455*, 2021.

Murray, Naila and Perronnin, Florent. Generalized max pooling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

Ren, Xiaoli, Li, Xiaoyong, Ren, Kaijun, Song, Junqiang, Xu, Zichen, Deng, Kefeng, and Wang, Xiang. Deep learning-based weather prediction: A survey. *Big Data Research*, 23:100178, 2021. ISSN 2214-5796. doi: https://doi.org/10.1016/j.bdr.2020.100178. URL https://www.sciencedirect.com/science/article/pii/S2214579620300460.

Shwartz-Ziv, Ravid and Armon, Amitai. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

Smith, Brian A, McClendon, Ronald W, and Hoogenboom, Gerrit. Improving air temperature prediction with artificial neural networks. *International Journal of Computational Intelligence*, 3(3):179–186, 2006.

Wang, Shuihua, Jiang, Yongyan, Hou, Xiaoxia, Cheng, Hong, and Du, Sidan. Cerebral micro-bleed detection based on the convolution neural network with rank based average pooling. *IEEE Access*, 5:16576–16583, 2017.

Zhang, Chongsheng, Liu, Changchang, Zhang, Xiangliang, and Almpanidis, George. An up-to-date comparison of state-of-the-art classification algorithms. *Expert Systems with Applications*, 82:128–150, 2017.

Zhou, Kanghui, Zheng, Yongguang, Li, Bo, Dong, Wansheng, and Zhang, Xiaoling. Forecasting different types of convective weather: A deep learning approach. *Journal of Meteorological Research*, 33(5):797–809, 2019.

Ömer Altan Dombaycı and Gölcü, Mustafa. Daily means ambient temperature prediction using artificial neural network method: A case study of turkey. *Renewable Energy*, 34(4):1158–1161, 2009. ISSN 0960-1481. doi: https://doi.org/10.1016/j.renene.2008.07.007. URL https://www.sciencedirect.com/science/article/pii/S0960148108002851.