Weather forecast using image classification

Időjárás előrejelzés képosztályozás segítségével

Nóra Stumphauser*, Bence Lestyan‡
Department of Mathematics, Budapest University of Technology and Economics, Hungary
*stumpy.nori@gmail.com ‡bence.lestyan@gmail.com

***Abstract*— The main idea of this research is to compare simple classifiers and neural network image classification for time series data with a well-known task transferred to the topic of weather forecasting. The task is usually the churn prediction, which means predicting what customers tend to leave the company. This can be important for banks or telecommunication companies. We used not common theme in this area, weather data and we would like to predict whether the average temperature on a given day was higher than a given value (the all day's median). This gave a binary classification problem, which was solved by simple classification methods for benchmark models (decision trees, SVM, KNN) as first step. Then an image was taken for each day with a size of 24x7 as 7 attributes were stored for the rows. Each pixel in the image was colored according to the value of that cell. 4004 images were created and sliced into train, validation and test sets. We built networks similar to those in the literature (DL-1, DL-2), also we tested the LSTM network, as well as transfer learning and hyperparameter optimization. The results were compared in terms of AUC and we found that the results obtained by neural networks were in many cases better than machine learning techniques for a binary classification problem (even 6% improvement). Thus, we mainly succeeded in proving the hypothesis of our theoretical and methodological research.**

***Kivonat*— A kutatás alapötlete az, hogy idősoros adatokra egyszerű osztályozók pontosságát hasonlítsuk össze neurális hálózatok képosztályozási képességeivel. Ez a feladat általában lemorzsolódás jóslását jelenti, azaz azt próbáljuk becsülni, hogy egy vásárló elhagyja-e a céget. Ez általában a bankszférában és telekommunikációs cégek esetében fontos. Egy nem szokványos témát felhasználva, kutatásunk során időjárás adatokat használtunk, melyben azt jósoltuk, hogy egy adott napra vett átlaghőmérséklet meghaladja-e az összes vizsgált nap átlaghőmérsékletének mediánját. Első lépésként, benchmark modelleket készítettünk el egyszerű osztályozó algoritmusok segítségével (döntési fák, SVM, KNN). Ezután minden napra készítettünk egy 27x7 pixelből álló képet, amelyen minden pixelt az adott cella értékei szerint színeztünk. 4004 képet generáltunk, melyet tanító, validációs és teszt halmazokra osztottunk. Képosztályozáshoz a szakirodalomban olvasott neurális hálózatokhoz hasonló hálókat készítettünk el (DL-1, DL-2), kipróbáltuk az LSTM neurális hálót, transfer learninget és hiperparaméter optimalizálást is. Az eredményeket AUC érték szerint hasonlítottuk össze és azt találtuk, hogy a neurális hálózatokkal elérhető pontosság sok esetben nagyobb, mint az egyszerű gépi tanulási módszerek eredménye (akár 6%-os javulás is elérhető). Tehát sikerült igazolni a főleg elméleti és módszertani kutatás hipotézisét, miszerint a neurális hálózatok jobban teljesítenek a feladaton.**

I. Introduction

In the project the goal was to compare the accuracy of simple classifiers and neural networks for a binary classification problem. We would like to show that the classifiers do not work as good on binary classification as neural networks on image classification. The problem in general is that there is a time-series dataset (for example user dataset) and binary classifier is built on this problem (usually for churn prediction). Intuitively, the classifiers like decision trees, usually not work very well on time-series data. On the other hand, neural networks can classify images easily with very high accuracy. So, we create images from data (for each person), and we use neural networks for classification.

The dataset we used is weather data from Kaggle datasets. The images were created for each day, and the goal of prediction was a binary variable which was created from the average temperature continuous attribute. This attribute was cut by median, and we predict weather the average temperature for a day is higher than 12.25 °C.

II. Literature review

About this method the literature usually contains research about churn. In 2006 there was a huge investigation in Taiwan [5]. They used several machine learning techniques for predicting churn, for example decision trees, k-nearest neighbor. Also, they used neural networks, but not for image classification, just for the rows of the data.

Similar article was published in 2012, where the researchers predict churn with a lot of machine learning techniques (Logistic Regressions, Linear Classifications, Naive Bayes, Decision Trees, Support Vector Machines and the Evolutionary Data Mining Algorithm) [6]. They were used in this research as methods for simple classifiers.

Hill et al. used random forest classifier for a specific weather forecasting problem [4]. From this, our expectation was that this method will be the best simple classifier on our dataset, too.

The main goal of this research is to differ the simple classification methods and the complex neural networks. This topic was examined in India in 2020 for weather forecasting.

They compare gradient boosting decision tree algorithm with ANN, and they found that in their case gradient boosting trees work better [3].

The idea of creating images from data came while reading an article from Asian researchers. [11] This paper shows that a simple neural network can do well on time series data. They got 0.74 AUC score on their data with 6 million customers.

III. Data

A Kaggle dataset were used from this website: https://www.kaggle.com/muthuj7/weather-dataset. The dataset contained 4004 days from 2006. January 1. to 2016 December 16. For 3 years there were only 363 days, for 5 years there were 364 days and for 3 years there were 365 days per year in the dataset. Descriptive statistical methods were used to examine the data.

For attribute Summary there were 27 types. The most common was Partly Cloudy with 31 616 occurrences (32.9%). There were three others with a high percentage of appearances: Mostly Cloudy (29.1%), Overcast (17.2%), Clear (11.3%). The types were coded into numbers from 1 to 27. Precip Type could be rain, snow or null. There was rain in outstandingly many cases, in 88.3%. The Humidity was between 0 and 1 with average of 0.74. Wind Speed ranged from 0 to 63.9 km/h, measuring an average of 10.8 km/h. The Wind Bearing was examined in degrees between 0 and 359, the average wind direction was 187.5 degrees. They also measured Visibility, the minimum was 0, the maximum was 16.1 km, and the average was 10.3 km. The last attribute was Pressure which was ranged between 0 and 1046.4 with an average of 1003.4 millibars.

IV. Data preparation

Several data preparation steps were made. The raw data contained 13 columns and 96 453 rows. The time column was divided into Date and Hour columns. Loud Cover column was dropped because it contained only zeros. Since 96 453 not divisible by 24, so in the first part of the program we deal with the problem of duplicates and missing data. 11 rows because of duplicates of hours (there were two 2:00 hours these days and we deleted the later one) were deleted which is negligible. Because of the missing data 14 days were dropped of all 4004 days, it is only 0.003% of the data so it will not mean a serious shortage. There was a date which appeared twice in the data, so it was deleted. After deletions the dataset contained 12 columns and 96 096 rows.

For next step target variable was created. Temperature-based target variable was planned to use, so we see the median of the mean temperatures, which was 12.25. Label 1 was created to those days where the mean is greater than 12.25. This case perfectly balanced data was generated with 2002-2002 days in each set. After this step, two columns were additionally dropped, because they had too much information about temperature, so the neural network could learn them easily. Labels and the data were saved into a csv file to use for the benchmark models.

V. Train, validation and test sets for image classification

In this research images were created which pixels are colored by the values of the cells in the dataset. 24 hours data were used for each day and 7 attributes, so the generated images were 7x24 pixels. For this method each column values were scaled into range 0 to 255 by min-max scaling. In the end of the procedure a 3D NumPy array was get with 4004 7x24 colored images.

Then train, validation and test sets were created with 70% (2802 images), 15% (601 images), 15% (601 images) of data, respectively.

VI. Results of benchmark models

For benchmark models simple binary classifiers were used on the data. Mainly decision tree classifiers were used, k-nearest neighbor and SVM algorithms were also tried for the prediction. Accuracy and AUC was measured on the validation and test sets.

First method was gradient boosting decision tree algorithm with several parameters which were optimized at some manually set ranges. The learning rate was seven values between 0.05 and 1, the max_depth parameter was changed between 1 and 5, and max_feature parameter was in the interval of 1 and 8. The algorithm was run with all possible permutations of the parameters. Next, we ran random forest classifier with parameter n_estimators from 1 to 1000 with steps 100. The optimized parameter was 600. For third algorithm we used logistic regression, but it seemed to be the worst method for this classification problem. Then we ran k-nearest neighbors algorithm for k equals 1 and 5, and we tried SVM also. The results are shown in Table 1 ordered by the AUC value of the test set.

TABLE 1
RESULTS OF THE SIMPLE CLASSIFIERS ORDERED BY AUC ON TEST SET

Method	VALID ACC	VALID AUC	TEST ACC	TEST AUC
Random Forest	67.09	67.06	67.53	67.54
Gradient Boosting	59.67	63.83	60.19	64.54
SVM	58.02	57.94	57.57	57.59
KNN-5	56.46	52.19	57.34	57.34
KNN-1	56.81	52.19	56.67	56.67
Logistic Regression	52.36	52.19	53.49	53.54

As we can see the best classifier was random forest algorithm with 67.54% AUC score on the test set. Our hypothesis is that the neural networks can perform better on our dataset.

VII. Neural networks

A. Callbacks

To begin with callbacks were defined. We set up early stopping with a 10 patience, which minimized the loss of the validation set (EarlyStopping). In addition, we introduced a callback to save good models (ModelCheckpoint). This was also set to monitor the minimization of validation error. Moreover, we created a learning rate modifier callback that observed that if the validation error does not improve by at least 0.1 for 7 epochs, it modifies the learning rate by a small value (ReduceLROnPlateau). Thus, further learning of the web became available due to the modification of the learning rate. The parameters were optimized manually in this part of the research, and the best models were saved in a folder for later use.

B. First neural networks

First, a simple neural network was created based on just intentions. The structure can be seen in Table 2. The number of the total trainable parameters was 203 457. We fitted each model for 40 epochs with 120 batch size, because it seemed to obtain the fastest training (the lower and higher batch sizes were not as good as this).

TABLE 2
Structure of the simple neural network

Layer	Output shape	Number of parameters
Conv2D	(5, 22, 32)	320
Conv2D	(3, 20, 64)	18 496
Conv2D	(1, 18, 64)	36 928
Flatten	(1 152)	0
Dense	(128)	147 584
Dense	(1)	129

To construct additional neural networks, we used the network structures found in [11]. These were DL-1 and DL-2, which use just a small number of filters for training. By trying these, learning could not be measured in any of our cases. So, the structure of the neural networks was changed manually but keeping the basic structure of the ones in the article. The structure of our DL-1 and DL-2 can be found in Table 3 and Table 4, respectively. The first convolution layer is designed to monitor the variations of the time of day by examining the behavior of each attribute in 10-hour periods. The next convolution layer examines the hourly values of all attributes (like a transpose of the first one).

TABLE 3
Structure of the DL-1 like neural network

Layer	Output shape	Number of parameters
Conv2D	(7, 20, 128)	768
Conv2D	(1, 20, 64)	57 408
MaxPooling2D	(1, 10, 64)	0
Flatten	(640)	0
Dense	(256)	164 096
Dense	(1)	257

TABLE 4
Structure of the DL-2 like neural network

Layer	Output shape	Number of parameters
Conv2D	(6, 15, 128)	2 688
Conv2D	(1, 15, 64)	49 216
MaxPooling2D	(1, 7, 64)	0
Dropout	(1, 7, 64)	0
Flatten	(448)	0
Dense	(128)	57 472
Dropout	(128)	0
Dense	(32)	4 128
Dense	(1)	33

Then parameters and layer styles were changed manually, and the best models were saved out. The AUC value on the test set were examined and the results are shown in Table 5 in descending order.

TABLE 5
Best manually optimized models ordered by the achieved AUC values

Model	AUC value
DL-2 v4	73.52
DL-2 v3	73.28
DL-2 v2	72.45
DL-2 v5	72.10
DL-2 v6	71.77
DL-1 v2	71.33
simple network v2	70.80
simple network v1	70.12
DL-1 v1	69.34
DL-2 v1	67.85

As we can see the best models are usually created from the DL-2 structure. The best achieved AUC score was 73.52%, which is better by nearly 6% than simple classifiers AUC values. From this fact we can conclude that our hypothesis seems to be justified.

C. Hyperparameter optimization

1) Keras tuner with RandomSearch

By loading the manually optimized best-performing model, we looked for hyperparameter optimization to see if improvement could be achieved with other network settings. Random Search can be used to optimize many parameters [2]. For the first two Convolution layers, the number of filters varied randomly between 128 and 512, and the percentage of dropout layers was also randomly generated by the program between 0.0 and 0.2 in 0.05 increments. The size of the additional Dense layers also varied randomly between 128 and 512, 64 and 256, 32 and 128, with step intervals of 128, 64, and 32, respectively. The permutations were made up to 50 times by the program, and then the teachings were run at a batch size of 120 for 75 epochs. The best model outcome obtained in this way was 70.26% AUC.

2) Keras tuner with Hyperband

The second hyperparameter optimization algorithm was the Hyperband, with which the algorithm decides whether it is worth teaching the network with the given parameters after just some epochs [7]. With this, hyperparameter optimization runs much faster. We ran similar settings as for Random Search, for a minimum of 3 and a maximum of 75 epochs. The best model obtained with this already produced an AUC value of 71.32%.

We ran the Hyperband Optimizer not only on the best model we got in the first phase, but also on a simpler model we created before. In this case, the neural network started with 3 convolution layers, with the number of filters in each layer ranging from 32 to 512 in 32-step increments. This was followed by a Flatten layer and after by a Dense layer with a unit value between 64 and 128 in 64 step increments. Finally, a layer of Dense closed the net. similar teaching was performed as for a DL-2-like neural network. The AUC of the resulting model was 70.75%.

D. LSTM

The neural network often used for time series data is the LSTM network [6]. This was also tested in this work, but due to the specific data set, the expectation was that the model would not produce good results. The neural network shown in Table 6 was taught at 120 batch size for 40 epochs. The result is an AUC of only 50.00% on both the validation and test sets. So, it can be said that this method is not suitable for this type of weather forecast.

TABLE 6
STRUCTURE OF THE DL-1 LIKE NEURAL NETWORK

Layer	Output shape	Number of parameters
LSTM	(7, 128)	78 336
Dropout	(7, 128)	0
LSTM	(128)	131 584
Dropout	(128)	0
Dense	(32)	4 128
Dropout	(32)	0
Dense	(1)	33

E. Transfer learning

Because of [1], we also used transfer learning. To do this, the images were first converted to appropriate .tiff files, and then the suitable teacher (2800 images), validation (600 images), and test (604 images) sets were created using Image Data Generator. InceptionResNetV2 was loaded and then taught the dataset with it. First, running Adam optimizer and default trainable layer number, the result was 55.13% AUC on the test set. Then, by enabling multiple teachable layers with the Adam optimizer, no spectacular improvement was detected. When testing the SGD optimizer with a momentum of 0.9 and a learning rate of 0.0001, we found that the AUC value reached only 44.87% on the test set. Only 5 epochs were taught, this may also be the cause of poor performance, but it is likely that transfer learning is not a suitable method to predict the weather with images either.

VIII. CONCLUSION

In summary, we have achieved the aim of our project, which is to predict from time series data by comparing machine learning methods. We found that simple classifiers for weather data work less well than neural network image classification methods. The best result was achieved by a neural network similar to the DL-2 that we prepared with an AUC of 73.52%, which is 5.98% better than the AUC of 67.54% achieved by the best classifier.

For future work better AUC values can be achieved by CNN-SVM neural networks or other hybrid neural networks, which are usually used for churn prediction or some special image classification problem. [8,9] Also, it can be a good idea to investigate this methodological difference on other datasets, too. Another version of this task could be to create a regression problem for temperature prediction. This task could be solved by LSTM recurrent neural networks, as Mohamed and Chaker solved the problem in 2016 [12].

ACKNOWLEDGEMENT

We would like to thank the instructors of the course Deep Learning in Practice with Python and LUA for creating and delivering this awesome course. We learnt a lot new things, which can be used in real life and work.

REFERENCES

[1] Ahmed, Uzair et al. "Transfer learning and meta classification based deep churn prediction system for telecom industry". arXiv preprint arXiv:1901.06091. (2019).

[2] Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." The Journal of Machine Learning Research 13.1 (2012): 281-305.

[3] Datta, Anisha, Shukrity Si, and Sanket Biswas. "Complete Statistical Analysis to Weather Forecasting." Computational Intelligence in Pattern Recognition. Springer, Singapore, 2020. 751-763.

[4] Hill, Aaron J., Gregory R. Herman, and Russ S. Schumacher. "Forecasting Severe Weather with Random Forests." Monthly Weather Review 148.5 (2020): 2135-2161.

[5] Hung, Shin-Yuan et al. "Applying data mining to telecom churn management". Expert Systems with Applications 31. 3(2006): 515–524.

[6] Huang, Bingquan et al. "Customer churn prediction in telecommunications". Expert Systems with Applications 39. 1(2012): 1414–1425.

[7] Li, Lisha, et al. "Hyperband: A novel bandit-based approach to hyperparameter optimization." The Journal of Machine Learning Research 18.1 (2017): 6765-6816.

[8] Mena, C. Gary, et al. "Churn Prediction with Sequential Data and Deep Neural Networks. A Comparative Analysis." arXiv preprint arXiv:1909.11114 (2019).

[9] Niu, Xiao-Xiao, and Ching Y. Suen. "A novel hybrid CNN–SVM classifier for recognizing handwritten digits." Pattern Recognition 45.4 (2012): 1318-1325.

[10] Tsai, Chih-Fong, and Yu-Hsin Lu. "Customer churn prediction by hybrid neural networks." Expert Systems with Applications 36.10 (2009): 12547-12553.

[11] Wangperawong, Artit et al. "Churn analysis using deep convolutional neural networks and autoencoders". arXiv preprint arXiv:1604.05377. (2016).

[12] Zaytar, Mohamed Akram, and Chaker El Amrani. "Sequence to sequence weather forecasting with long short-term memory recurrent neural networks." International Journal of Computer Applications 143.11 (2016): 7-11.