Second Milestone (first milestone supplemented)

Team: HáLab

Team members: Stumphauser Nóra, Lestyan Bence

Motivation

In the project our goal is to compare the accuracy of simple classifiers and neural networks for a binary classification problem. We would like to show that the classifiers don't 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, or like in our project weather data) and we would like to build a binary classifier (usually for churn, but we use a label invented by ourselves). 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, or in our case for each day) and we use neural networks for classification.

Literature review

About this method the literature usually contains research about churn. In 2006 there was a huge investigation in Taiwan. [2] 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 on 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) that we can use as methods for simple classifiers. [3] The idea of creating images from data came while reading an article from Asian researchers. [6] It is a short paper but shows that a simple neural network can do well on this type of data. They got 0.74 AUC score on their data with 6 million customers. (we will have more cites in the last document)

Data preparation

We use a Kaggle dataset from this website: https://www.kaggle.com/muthuj7/weather-dataset

First, we load the files and change every value to a number. We divide the time column into day and hour. The raw data contained 13 columns and 96 453 rows. We realize that 96 453 not divisible by 24, so in the first part of the notebook we deal with the problem of duplicates and missing data. We deleted 11 rows which is negligible. And because of the missing data we should drop 14 days of the 4018 days, it is only 0,003% of the data so it will not mean a serious shortage. We need to delete a day that was twice in the data. We also drop a column, because it contains only 0 numbers. After deletions we got a dataset with 12 columns and 96 096 rows.

In the next step we create our target variable. We would like to choose a temperature-based target variable, so we see the median of the mean temperatures and we got that it is 12,25. So we create label 1 to those days where the mean is greater that 12,25. This case we got perfectly balanced data with 2002-2002 days in each set. After this step we drop the two columns which contained too much information about temperature (it can be learnt easily), and saved the labels and the data into a csv file to use it in the other step of research (for simple classification).

Creating images

In this research we would like to create images, which pixels are colored by the values of the cells. We have 24 hours data for each day and 7 attributes, so we create 7x24 pixel images. For this we transform each column values into 0-255 by min-max scaling. In the end we got a 3D numpy array with 4004 7x24 colored images.

Train, validation, test sets

For Milestone 1 we create the train, valid and test datasets. We decided to make 70% for training, 15% for validation and 15% for test, but if we need to change these values during our work, we will. So, we got 2802 train images and 601-601 for validation and test.

NEW PART

Simple Classifiers

For a start we ran some simple classifiers on our data. We decided to try out some decision tree classifiers, but we also used k-nearest neighbor and SVM algorithms for the binary classification problem. We created train, validation, and test sets with 70%-15%-15% data in each, like in the neural network part. We measure accuracy and AUC score on validation and test sets.

First, we used gradient boosting decision tree algorithm with several parameters which were somewhat optimized. 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. We ran the algorithm 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 simple classifiers ordered by the accuracy on the 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 than this on our dataset.

Neural networks

To begin with, we defined some callbacks. 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.

First, we create a simple neural network based on our intentions. The structure can be seen in Table 2. The number of the total trainable parameters was 203 457. We fitted this model for 40 epochs with 120 batch size.

Table 2: Structure of the simple neural network.

LAYER	OUTPUT SHAPE	NUMBER OF PARAMS
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 [6]. 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 we performed the teaching phase by changing them manually but keeping the basic structure of the networks. The structure of our DL-1 and DL-2 can be found in Table 3 and Table 4, respectively.

Table 3: Structure of our DL-1 like neural network.

LAYER	OUTPUT SHAPE	NUMBER OF PARAMS
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 our DL-2 like neural network.

LAYER	OUTPUT SHAPE	NUMBER OF PARAMS
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

In evaluating part, we load the best models of the manually optimized ones, and we examined the AUC value on the test set. The results are shown in Table 5 in descending order.

Table 5: AUC scores on test set of the best manually optimized models.

MODEL	AUC SCORE
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 model structure is in Table 6. For this time the best achieved AUC score was 73.52, which is better by nearly 6% than simple classifiers AUC score. From this fact we can conclude that our hypothesis seems to be justified.

Table 6: Best performance model based on DL-2 structure.

LAYER	OUTPUT SHAPE	NUMBER OF PARAMS
Conv2D	(7, 20, 256)	1536
MaxPooling2D	(7, 10, 256)	0
Conv2D	(1, 10, 128)	229 504
Dropout	(1, 10, 128)	0
MaxPooling2D	(1, 5, 128)	0
Flatten	(640)	0
Dense	(128)	82 048
Dense	(64)	8 256
Dense	(32)	2 080
Dense	(1)	33

Future work

For next steps we would like to create a hyperparameters optimization in Keras and try to make better neural networks than our manually optimized ones. Also, we plan to try LSTM network, because we read in [4] that it can be also a good method for churn prediction. Of course, our data is different, so we think that this method won't be as good as the tried ones. We would like to use some transfer learning technics, too. In image classification this is a well-known method. If we have enough time, we will make hybrid classifiers based on [1]. In this article researcher created a CNN with SVM and random forest in the end. Because of the characteristic of our problem, it can be classified with a simple CNN or a simple random forest well, so it is a good second hypothesis that the ensemble method would be even better. Also we found that for handwritten digit recognition the CNN-SVM is a widely used method, like in a work of two Chinese researchers. [5]

Bibliography

[1] Chouiekh, A. (2020). Deep Convolutional Neural Networks for Customer Churn Prediction Analysis. International Journal of Cognitive Informatics and Natural Intelligence (IJCINI), 14(1), 1-16.

[2] Hung, S. Y., Yen, D. C., & Wang, H. Y. (2006). Applying data mining to telecom churn management. *Expert Systems with Applications*, *31*(3), 515-524.

[3] Huang, B., Kechadi, M. T., & Buckley, B. (2012). Customer churn prediction in telecommunications. *Expert Systems with Applications*, *39*(1), 1414-1425.

[4] Jie, Z. H. O. U., YAN, J. F., Lu, Y. A. N. G., Meng, W. A. N. G., & Peng, X. I. A. (2019). Customer Churn Prediction Model Based on LSTM and CNN in Music Streaming. DEStech Transactions on Engineering and Technology Research, (aemce).

[5] Niu, X. X., & Suen, C. Y. (2012). A novel hybrid CNN–SVM classifier for recognizing handwritten digits. Pattern Recognition, 45(4), 1318-1325.

[6] Wangperawong, A., Brun, C., Laudy, O., & Pavasuthipaisit, R. (2016). Churn analysis using deep convolutional neural networks and autoencoders. *arXiv preprint arXiv:1604.05377*.