COS711 Assignment 3

Luveshan Marimuthu
u17087709
https://github.com/Luveshanm/cos711Assignment5
Dept. of Computer Science
University of Pretoria

***Abstract*—The purpose of this report is to compare the performance of two deep learning techniques on air quality prediction. The report shows the results of a Convolutional Neural Network (CNN) and a Long Short-Term Memory (LSTM) Network which are implemented in Python using the TensorFlow/Keras API. The data is preprocessed to fill in missing values and scale the input values for each feature (column).**

Results Summary

***Index Terms*—Convolutional Neural Network, Long Short-Term Memory, Deep Learning, Air Quality.**

I. Introduction

In this report I apply two deep learning techniques, Convolutional Neural Network(CNN) and Long Short-Term Memory(LSTM) Network, to create an air quality prediction model. The data pre-processing as well as the prediction models are implemented in python.

The data set used to train and test the models is provided by https://zindi.africa which is an African data science and machine learning competition platform. Each pattern in the data set is a 5-day series of hourly weather data readings from different sensors across Uganda. Therefore the air quality prediction task is of a temporal nature. The target value is the air quality level at exactly 24 hours after the last sensor readings. The air quality is measured in PM2.5 (particulate matter smaller than 2.5 micrometers in diameter or around 1/30th the thickness of a human hair) and is measured in micro-grams per cubic meter (μ/m3). [1]
The pre-processing of the data set involves converting the column values into numerical arrays, filling in the missing values for some columns in each row in the data and scaling all of the values to a common range in preparation for machine learning.

Concolutional Neural Networks(CNNs) are a category of deep neural networks that is typically used for image classification. CNNS have however also been applied to time series prediction tasks. In this assignment a convolutional neural network is implemented to train on the data from the weather sensors as 2D arrays of readings for different 'features' (columns). Given the structure of the data, the 2D inputs are matrices of *features x readings*.
Long Short-Term Memory(LSTM) Networks are suited for making predictions on time series data and is able to deal with the vanishing gradient problem of traditional Recurrent Neural Networks (RNN). The LSTM model is trained on the same data as the CNN with input patterns being matrices of *features x readings*.

mention the outcomes

II. Background

The weather data needs to be analyzed and pre-processed to make it possible for the deep learning algorithms to more effectively learn the hidden relationships between the input features and target values. Input scaling is a crucial part of pre-processing as having features on a similar scale can help the models converge more quickly.
In the given data set, each column consisted of a string of values. These values were converted to numerical arrays to create a 2D array representation on the data. As with most real world machine learning problems, the data had missing entries scattered throughout the set. Usually the rows with missing values are removed however there were too many rows that had missing values and so removing these would mean losing somewhat valuable data patterns. Instead the data was filled in by replacing them with an average of the present values. The average was calculated across the array of values in that particular column for that particular row. This was done because the array of values for one row in one specific column are weather readings of 5 consecutive days which in the literal sense by nature are more likely to have similar readings because the weather will be somewhat, but not necessarily, similar.
Lastly the values in each column were scaled using standardization (z-score normalization), this time taking into account all of the values across all the rows in the data set.

In the case of CNN, the model will seek to find any spacial relationships between the weather readings. Although the data is temporal in nature, weather readings of 5 consecutive days have a good chance of having similar properties. Furthermore, the CNNs pooling technique for feature extraction could possibly find relationships that generalize across the different sensor locations.

In the case of LSTM, these models excel at time series prediction and I expect it to outperform the CNN, together with the fact that CNNs were designed more for images analysis. LSTMs are able to 'remember' past data patterns which should prove useful in this task. By remembering more information the model could simulate having a longer series per pattern during training, for example hypothetically having 7 days worth of readings instead of only the 5 days provided in each pattern.

The various hyper-parameters and number of layers (depth) of each model was decided through trail and error. Initially, the models began more simple (smaller) and increased in complexity (number of layers or units per layer) until they reached diminishing return or the training time became too long. The activation functions and optimizers play an important role in training and were chosen through their known successes in practice.
The metric used by Zindi to judge the competition entries was the Root Mean Squared Error function.[1] For this reason, I used the Mean Squared Error loss functions and included the Root Mean Squared Error metric in the evaluation to determine the relative performance/'accuracy' of my models. *Note*: I did not enter the competition.

III. Experimental Set-Up

The deep learning models discussed below were implemented in Python using the following libraries: TensorFlow, Keras, pandas, numpy and scikit-learn.

A. Pre-processing

The data was loaded from the .csv file into a pandas dataframe. As mentioned earlier the column values were strings of values so these had to be converted into 1D arrays of floating point values. Afterwards, each column was separated into individual 2D arrays to make the following pre-processing steps easier to compute.

1) Missing Values : Missing values were scattered all throughout the data set. Some were singular missing entries and others were long strings of missing entries, probably due to events such as power cuts or maintenance time. In light of this, two strategies were used to fill in the missing entries:

First, the singular missing values were found and replaced with the average of the values immediately before and after the missing value. In terms of weather readings this was ideal because even if the before and after readings are very different, the change from one to the other is gradual thus using the average as the inbetween value. The corner cases are either the very first or the very last value is missing, in which case it is simply filled in with the value immediately after or before respectively.

Second, some rows/columns still contain strings of two or more consecutive missing values. In this case the missing entries are replaced with an average of all the values of that particular column for that particular row. As explained in the background, this was done because the array of values for one row in one specific column are weather readings of 5 consecutive days which in the literal sense by nature are more likely to have similar readings because the weather will be somewhat, but not necessarily, similar. Furthermore, since no date and time information was given with the data, the 5-day readings across the rows could possibly be months apart and have no correlation in terms of weather which makes calculating the average across the rows of a specific column inappropriate for this scenario.

2) Input Scaling :

B. CNN

C. LSTM

IV. Research Results

V. Conclusions

expand by making specific to locations

References

[1] *AirQo Ugandan Air Quality Forecast Challenge*. 2020. URL: https://zindi.africa/competitions/airqo-ugandan-air-quality-forecast-challenge/data (visited on 06/10/2020).