Deep Learning
CS-6863

Final Homework

Instructor:
Mahdi Khodayar

Author:
Selim Karaoglu

May 5, 2022

This project is the final homework of Deep Learning class. In this project, we present answers to questions about two Deep Learning areas. First we experiment with PCA and CNN on CIFAR10 dataset, further we focus on time series data analysis with LSTM and GRU. After each experiment, we provided the experiment results and compare different Deep Learning methods.

1 Introduction

In this assignment, we provide answers to the questions presented in the final homework assignment. There are two different questions with subproblems. First part of the assignment focuses on the experiments with the CIFAR10 dataset[1]. We run PCA dimensionality reduction and train different Convolutional Neural Networks (CNNs). Further we present the experiment results and compare different models with their accuracy scores. In the second part of this project, we shift our focus to time series data experiment with Recurrent Neural Networks (RNNs) by training several LSTM and GRU models. We illuminate their performance by comparing their mean squared error (MSE) losses.

The source file provided with this homework assignment is structured accordingly to what assignment suggested:

For each question, type a report in word that shows your answer and your results. Please save your word file as PDF and submit your PDF + all your source codes in separate folders named for example “Q1-a”, “Q1-b”, . . .

You can use any toolbox/package in any programming language to answer the questions.

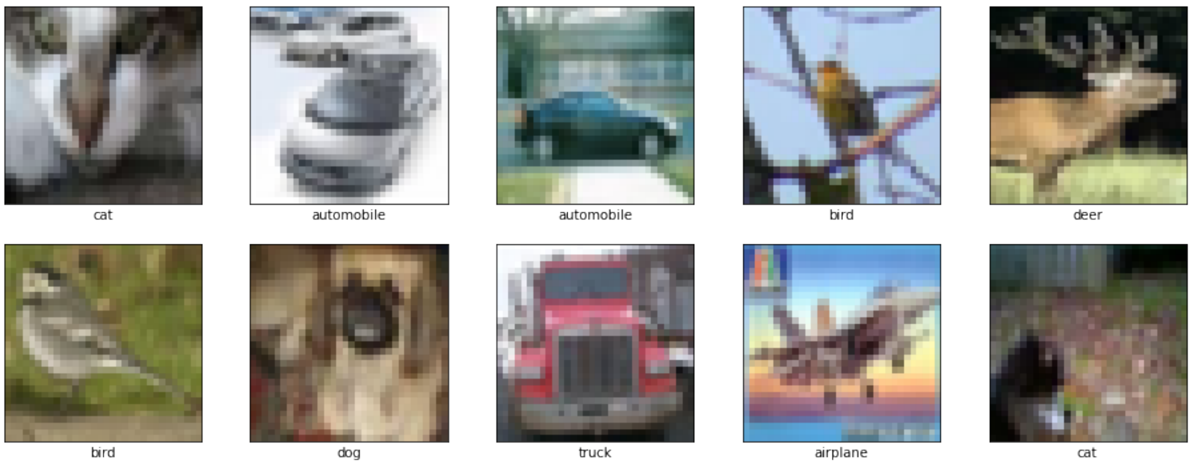
2 Question 1 - Convolutional Neural Networks

The CIFAR-10 dataset is a famous computer vision dataset. It consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. Download the dataset using this link: https://www.cs.toronto.edu/∼kriz/cifar.html

a) Plot 5 random images inside the training set and 5 random images inside the testing set.

Answer: CIFAR10 is an image dataset that contains images from 10 categories with sizes 28x28. These images are categorized according to what they represent; airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. As stated in the dataset's website[1]: "The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class."

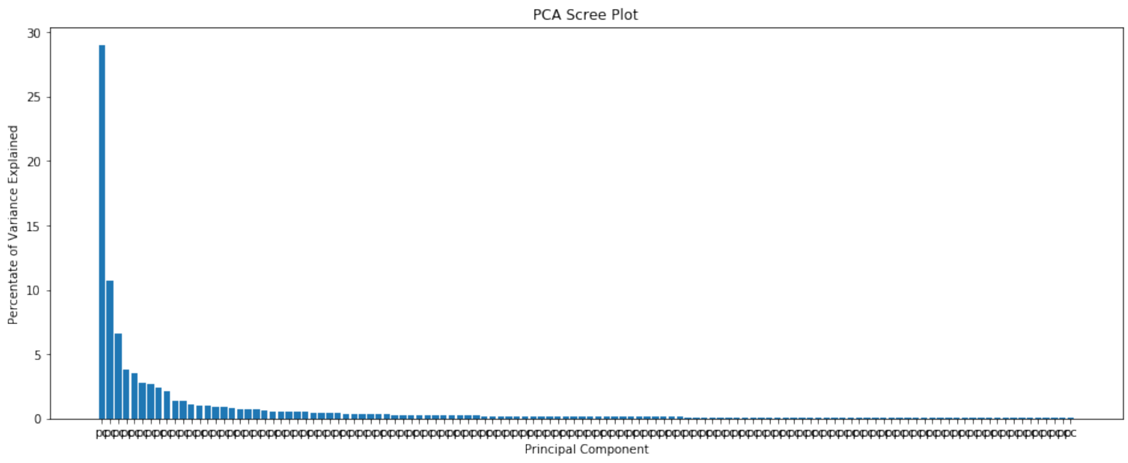
Here is the resulting output of our code that picks 5 random training images and 5 random test images and plot them. The top row represents to the 5 randomly selected training images and the bottom row represents the 5 randomly selected test images.



[1]https://www.cs.toronto.edu/∼kriz/cifar.html

b) Randomly choose 1000 images from the training set and put them in a 1000*3072-dimensional data matrix A. Note that 3072 is 32*32*3 where 32 is the height and weight of the images and 3 is the number of colors (channels). Run PCA dimensionality reduction method on data matrix A and obtain the top 120 principal components (directions/components which show highest variation in the data). Report those components and show how much of data variation is represented by those 120 directions.

Answer: To implement principal component analysis (PCA) in our work, we utilized the PCA method provided by sklearn library[2]. After we randomly picked 1000 examples from training set, we implemented PCA dimensionality recudtion. PCA's cross validation score resulted with 3350.65 on 120 components. The component with the highest variance was resulted with %29 and the 2nd highest component resulted with %11 variance. Remaining 118 components resulted with variances lower than %7. We plotted the variance captured by each component:



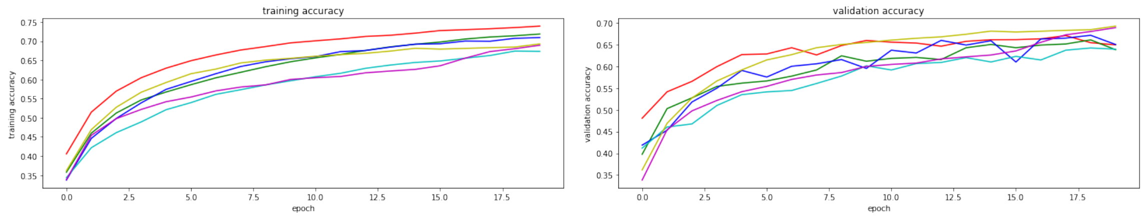
c) Train a convolution neural network in Keras with L number of layers. Each "layer" is a convolution+ReLU+Pooling. Set L to be 2,3, and 4. The Pooling can be Max Pooling or Average Pooling. Plot the training and testing loss and accuracy as a function of number of training epochs. Report the results of all 6 settings: L=2 with Max pooling, L=2 with Avg pooling, L=3 with Max pooling, L=3 with Avg pooling, L=4 with Max pooling, L=4 with Avg pooling. What setting is the best setting? Why?

Answer: We developed 6 different CNNs to provide answers to this question. All of these CNNs are constructed with Convolution + Activation (ReLU) + Pooling structure for 1 convolutional layer. The output layer of these network is a dense layer with output size of 10. We constructed CNNs with max pooling and average pooling and convolutional layer numbers of 2, 3 and 4. We trained and tested the CNNs on the same training and test sets with 20 epochs:

- First CNN with 2 convolutional layers and max pooling achieved 0.7392 training accuracy and 0.6495 test accuracy scores.
- Second CNN with 2 convolutional layers and average pooling achieved 0.7185 training accuracy and 0.6384 test accuracy scores.
- Third CNN with 3 convolutional layers and max pooling achieved 0.7092 training accuracy and 0.6504 test accuracy scores.
- Fourth CNN with 3 convolutional layers and average pooling achieved 0.6733 training accuracy and 0.6395 test accuracy scores.
- Fifth CNN with 4 convolutional layers and max pooling achieved 0.6892 training accuracy, and 0.6579 test accuracy.
- Sixth CNN with 4 convolutional layers and average pooling achieved 0.6927 training and 0.6402 test accuracy.

[2]https://scikit-learn.org/stable/modules/decomposition.html#pca.

Our experiment results can also be seen from the training and validation comparison plots presented below. The colors represent the models respectively with; red, green, blue, cyan, magenta and yellow. Validation accuracy scores shows that the sixth CNN model represented with yellow performed the best validation score after 20 epochs.



3 Question 2 - Recurrent Neural Networks

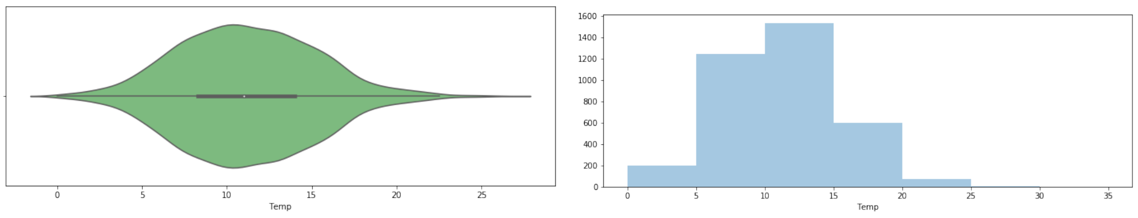
The excel file “daily-min-temperatures.csv” describes the minimum daily temperatures over 10 years (1981-1990) in the city of Melbourne, Australia. The units are in degrees Celsius and there are 3650 observations. The source of the data is credited as the Australian Bureau of Meteorology. At each time t, we want to observe the measurements of:

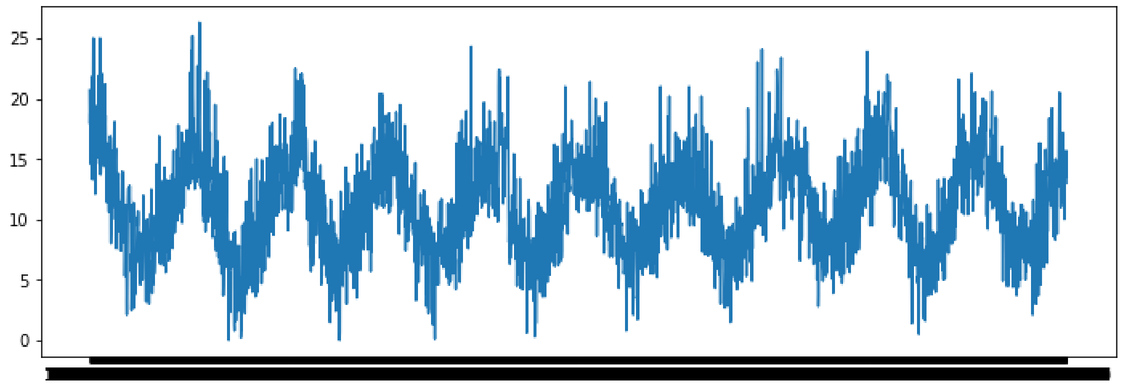
$t-k, t-k+1, t-k+2, \ldots, t-1, t$

To predict the measurement at time $t+1$. This problem is similar to the prediction problems we solved in the class.

a) Create a training set and a testing set for this problem. In your code, k should be an input variable. You can assume 80% of the data is used for training and 20% for testing.

Answer: For this question, we implemented the temperature time series dataset provided with the homework assignment. We wrote our own functions to create timeseries with given time point t and number of previous data to train k. This functions takes the given t and k values and returns the training set with size k, validation set with size $train\_split - t + k$ and test set with size $dataset\_length - train\_split$. The code implementation should be examined for clarity. This dataset contains a date string for each day of observations and minimum daily temperature value for that day. The temperature distributions can be seen from the plots:



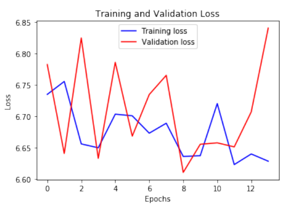


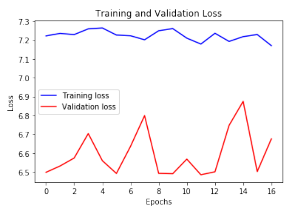
b) Train an LSTM to predict the measurement at $t+1$, and report the mean squared error and mean absolute percentage error of the training and testing as a function of the number of epochs used. Do this for four different values of k.

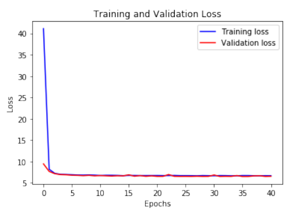
Answer: With the function provided for previous question, we assigned four different t and k values and experimented with them. Since as the k increases, the number of the training examples also increase. More training examples can yield better accuracy scores for LSTM models. The experiment results showed that there are not a big difference on the training losses for different k sizes. This can be the caused by the simplicity of the LSTM models we employed in this project with only 1 LSTM layer with 32 neurons and a Dense output layer. The effects of different k values might yield more distinctive results with more complicated RNN models. For small k values our model stopped improving on 6.5 validation loss on training, for larger k values the results improved to 6.48. The change in the validation loss is not very meaningful considering our experiment results. Similarly on the test results with Mean Squared Error (MSE); the highest k value resulted with the lowest MSE value (lower is better on MSE). However, the difference between the smallest and largest k value experiments is 0.19 MSE. This difference does not appear too important, but the experiment conditions can be improved to achieve better observations with more complex networks. We experimented with four different k values; 500, 1000, 1200 and 2000. Results are shown respectively:

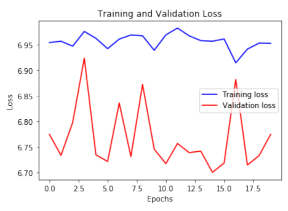
- First experiment with $k = 500$: LSTM model achieved Train Score: 6.59 MSE (2.57 RMSE) and Test Score: 5.84 MSE (2.42 RMSE) with early stopping on 14 epochs.
- Second experiment with $k = 1000$: LSTM model achieved Train Score: 7.22 MSE (2.69 RMSE) and Test Score: 6.11 MSE (2.47 RMSE) with early stopping on 17 epochs.
- Third experiment with $k = 1200$: LSTM model achieved Train Score: 6.72 MSE (2.59 RMSE) and Test Score: 5.95 MSE (2.44 RMSE) with early stopping on 41 epochs.
- Fourth experiment with $k = 2000$: LSTM model achieved Train Score: 7.02 MSE (2.65 RMSE) and Test Score: 5.76 MSE (2.40 RMSE) with early stopping on 20 epochs.

Resulting training and validation losses are presented respectively:





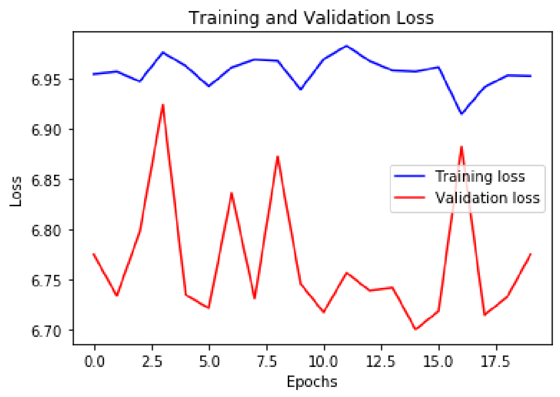


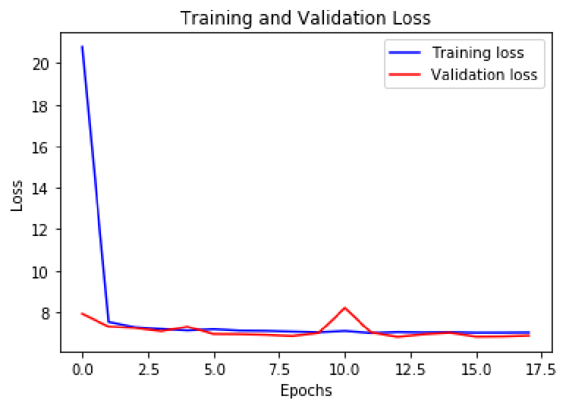


Our experiment results showed that the best test score is achieved with the k value being 2000. It makes sense since the larger k values provide a larger training set size, therefore yield more accurate models.

c) Do your research and find what a Gated Recurrent Unit (GRU) is. What is the difference between LSTM and GRU? Implement a GRU for this question and compare its results with your LSTM in both the training and testing datasets.

Answer: For this question, we implemented a GRU model with the same structure with LSTM layer. Our simple model is build with a GRU layer with 32 neurons. We trained the model with the same size training and test sets with previous question and our GRU model achieved Train Score 6.90 MSE (2.63 RMSE) and Test Score 5.91 MSE (2.43 RMSE).





Performance plots show the training and validation scores achieved by both models when they are trained with k value being 2000. The first graph shows the performance of the LSTM model and the second graphs shows the performance of the GRU model. The LSTM model achieved 5.76 MSE score and the GRU achieved 5.91 MSE. Even though the MSE loss scores are very close for our models, the LSTM achieved slighlty better results. This is a result of a more complicated structure of LSTM models. The GRU models utilize 3 gates inside their structures while LSTM model employs and additional gate and designed with 4 gates. This structure helps LSTM to learn better with limited training sizes or lower numbers of epochs. Our experiment results suggested the more complex structure of LSTM resulted with better accuracy scores.