MSc - Media and Textmining

Homework results

Daniel Mark Kiss

2023

Contents

Chapter 1

Task description

The data set contains small black and white images labelled with character identifier. The task is to recognize the character (0-9, a-z, A-Z) of unknown images in the test set by deep learning classifier without any human activity (human's help is equivalent to cheating). Work out a deep learning classifier model, and train this model by the training dataset! Use cross-validation in order to measure the goodness indicators (accuracy, AUC) of your model! Please pay attention to the parameter optimization! Please investigate the model, the result and describe the details of your solution!

Chapter 2

Ellaborated work

In the chapter I will present the ellaborated work that I have done. For my work I have created separate classes and files for the different subtasks of the Homework. In every section I will present and explain what the different classses are used for.

2.1 program.py

Program.py is the core of the program. It is the main file that is used to run the program. It is responsible for the following tasks:

1. Initializing the FileReader class
2. Running the FileReader class functions
   - read_training_files()
   - create_train_set()
3. Initializing the ConvolutionalNeuralNetwork class
4. Running the ConvolutionalNeuralNetwork class functions
   - split()
   - compile()
   - train()
   - evaluate()
5. Reading the test files
6. Creating the test set

7. Exporting results to output.txt file

When we would like to run the program we should also import the FileReader and ConvolutionalNeuralNetwork classes. For running we should type python program.py to the terminal and the program should start running.

2.2 cnn.py

This file contains the fundamental code for the neural network. I used many other libraries for my homework like *numpy*, *tensorflow*, and *sklearn* also.

2.2.1 init

In the __init__ function I initialize the arrays which I used for traing and spliting. These are the different variables that I used in my class:

- self.images: contains the raw images
- self.label: contains the labels for the images
- self.img_train: contains the train set of images
- self.img_test: contains the test set of images.
- self.label_train: contains the labels of the train set
- self.label_test: contains the labels of the test set
- self.model: is a model used from tensorflow.keras.It has five layers, three 2D convolutional layers and two 2D MaxPooling layers. The first convolutional layer has 32 filters/kernels. Each filter is 3x3 matrix. I choose ReLU as the activation function. The input shape is 128x128, 1 which is set because the given images are this size and they are grey scaled. The MaxPooling layers are used for down-sampling and reducing the spatial dimensions of the input. It uses a 2x2 pooling window/filter. It also helps reducing the chnace of overfitting.It only retains only the most important informaton from it.

After the Initialization of the model I add a Flatten layer to it.It is used for converting and outputing a 1D array. It is necessary before passing to a dense, fully connected layer. The next line adds a layer with 64 units (neurons) and a ReLU activation function. A dense layer means that each neuron in this layer is connected to every neuron in the previous layer. The last line in the init function adds another dense layer with 62 units, representing the output layer. The activation function used in the output layer is "softmax." Softmax is often used for multi-class classification problems. It

converts the raw output scores into probabilities, where each value represents the probability of the input belonging to a particular class. The sum of all probabilities for a given input is 1, making it suitable for classification tasks like this task also.

2.2.2 split

The split function is used for spliting the images and the labels into traing and testing set, with the test set is 33% of the input. I also add a $\text{random}_{s}tateseedsoitwillresultalawysthesameoutput.$

2.3 labeler.py

2.4 filereader.py

Chapter 3

Results and parameter optimization