MACHINE LEARNING

SHEET 5

27.01.2022

10:15 A.M. UNTIL 11:45 A.M. VIA ZOOM

Please prepare the exercises in order to present them in the meeting. Fill-in the exercises you solved in the questionnaire in StudIP until 26.01.2022 4:00 p.m.
Join the meeting via this Zoom link:
https://uni-trier.zoom.us/j/81323251864?pwd=aGdtZ2RHbnJiTW5Ed0xHL1NsYU9Hdz09

TASK 1: APPLYING IMAGE FILTERS

In this exercise, we will apply filters on images to see their effect. This is important to see how Convolutional Neural Networks (CNNs) learn.

a) Load the MNIST fashion dataset (https://www.tensorflow.org/datasets/catalog/fashion_mnist) and scale the data to the range of [0,1].
b) Select five images randomly from the training set and apply the following convolutions (using the library *scipy.ndimage*) on the original image:
    1. [[-1, -1, -1], [2, 2, 2], [-1, -1, -1]]
    2. [[-1 ,2, -1], [-1, 2, -1], [-1, 2, -1]]
    3. [[-1, -1, 2], [-1, 2, -1], [2, -1, -1]]
c) Visualize side-by-side comparisons of the transformed images and the original images. What are the convolutions highlighting?

TASK 2: CONVOLUTIONAL NEURAL NETWORKS WITH KERAS

This exercise works with custom CNN architectures that are suited for image classification.

a) Load the MNIST fashion dataset (https://www.tensorflow.org/datasets/catalog/fashion_mnist) and scale the data to the range of [0,1].
b) Build a simple CNN with the following layers, print a summary of the structure, train it with appropriate parameters for five epochs, and evaluate it on the training set:
    1. A reshape layer that adds a third dimension to the images.
    2. Two convolutional layers (128 and 64 filters with a 3x3 kernel size). Explain the output shapes printed in the summary.
    3. A flatten layer followed by two dense layers with 64 neurons and 10 neurons, respectively. The last dense layer should have a softmax activation.
c) Add a max-pooling layer after every convolution (pool size of 2x2) and compare the performance of the new model with the old one. Is it performing better? Why is pooling a good chance for performance improvements?

DEPARTMENT OF BUSINESS INFORMATION SYSTEMS II
PROF. DR. RALPH BERGMANN
UNIVERSITY OF TRIER

Optional Task 3: Using Pre-Trained Convolutional Neural Networks

In this exercise, we will look at CNNs from a different point of view. We use a pre-trained TensorFlow model for the CIFAR-10 dataset (https://www.cs.toronto.edu/~kriz/cifar.html).

a) Fetch the CIFAR-10 dataset from Keras datasets and scale the images to the range of [0,1].
b) Load the pre-trained model from Tensorflow Hub as a Keras layer (https://tfhub.dev/deepmind/ganeval-cifar10-convnet/1). Inspect the loaded model. Can you see its structure?
c) Predict the classes for three exemplary images and visualize the predicted probabilities for each class.
d) Evaluate the accuracy of the model on the test images. You should split the data into smaller pieces to avoid OOM errors.
e) How could this model be used for transfer learning?

Task 4: Manual Convolutions and Poolings

In this exercise, we will manually perform convolutions and pooling operations. Implement two methods (one for convolutions and one for pooling) and test your methods with some data.

a) Convolutions:
   Implement a method that takes an input matrix (assume a 2D array which has the same number of rows and columns) and a kernel matrix (assume a 2D array that has the same number of rows and columns). Perform the convolution operation by applying the kernel matrix to the input matrix. Assume that overlapping is desired, i.e., a stride of (1,1).
b) Max-Pooling:
   Implement a method that takes an input matrix (assume a 2D array which has the same number of rows and columns) and an integer number that specifies the edge length of the pool matrix (assume a 2D array which has the same number of rows and columns). Perform the max-pooling operation to the input matrix. Assume that overlapping is desired, i.e., a stride of (1,1).
c) Test both methods with a few examples.

Task 5: Recurrent Neural Networks With Keras

This exercise will work with Recurrent Neural Networks (RNNs) to predict time-series data. We will use a dataset of a continuous EEG measurement (https://www.openml.org/d/1471).

a) Fetch the data and scale its values to [0,1]. Split the data into training and test set (test ratio of 0.2).
b) Create “sliding windows” for training and test data. Use the appropriate Keras preprocessing method with a sequence length of 10. Explain the structure of the resulting data of this method.
c) Create an RNN like the one shown in the picture below. Train the network for 15 epochs in predicting the correct binary output using cross-entropy loss. Use 10% of the training data as validation data during training. What purpose do the individual layers of the network have?

```
model = keras.Sequential()
model.add(keras.layers.InputLayer(input_shape=(None, 14)))
model.add(keras.layers.TimeDistributed(keras.layers.Dense(128, activation='relu')))
model.add(keras.layers.TimeDistributed(keras.layers.Dense(256, activation='relu')))
model.add(keras.layers.SimpleRNN(128, return_sequences=True))
model.add(keras.layers.SimpleRNN(128))
model.add(keras.layers.Dense(128, activation='relu'))
model.add(keras.layers.Dropout(0.5))
model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dropout(0.5))
model.add(keras.layers.Dense(2, activation='softmax'))
```

d) Evaluate the performance of the model on the test set in terms of accuracy.
e) Replace the *SimpleRNN* layers with LSTM layers. What is the accuracy of the new model? What is the main advantage of LSTMs compared to SimpleRNN layers?

Optional Task 6: Recurrent Neural Networks For Sequences With Variable-Length

In the last exercise we used sequence data for training and testing that was split-up in sequences of equal length. In many real world scenarios, this is not the case.

a) Create univariate synthetic sequence data that contains 1000 examples with a random sequence length between 5 and 10 and a feature value range between 0 and 1. Also create 1000 targets that are integers between 1 and 5.
b) Usually, we feed a single numpy array to the *model.fit* method to train our model. You will notice that this is not possible with sequences of variable length. To overcome this limitation, we can use a tf.data generator that yields individual training batches of examples one by one. Implement a training loop with a generator and feed batches of examples to the model. Does it work?
c) One way to handle sequences of variable length is padding and masking. Explain how these two techniques work. Demonstrate these techniques by padding some of the synthetic sequences to a common length of 11 and masking the added values afterwards. What problems do you face?

d) Tensorflow's native solution to variable-length sequences are ragged tensors. Explain what they do. Mask the padded values to a ragged tensor so that the result is equal to the values before padding.