Pretraining_Example

December 14, 2021

1 Pretraining Neural Network

This is the main program structure that calls the sub functions and explains what happens in the code. The purpose of this program is to create the neural network model and train the model by fitting it to the given data. The trained model predicts based on the data given as an input and the target parameter used during the training.

To be more specific, the example case in our project predicts the plywood sheet shrinkage. We used the tensorflow keras regression model for the task.

1.0.1 Importing the Neural network program

This line imports the Neural_Network.py file that contains the following functions: - `get_database_data` - `create_and_fit` - `model_stat`

[1]:
```
from Pretraining.Neural_Network import *
```

With `help`-funtion you can view how to use the functions and which gives more detailed information on the parameters you can change:

```
# Example code:
help(get_database_data)
```

Getting data with `get_database_data` function The next line calls the function get_database_data which fetches the data from the database and creates the dataframe named df_features. This function also calls couple of additional programs: ssh_connector.py and chuncker.py.

The database connection requires correct connection configurations. These can be changed in the `variables.env` file.

For more information about these programs and the `get_database_data` parameters use the `help`-function.

[2]:
```
df_features = get_database_data(env_path="./variables.env",␣
↪table_name="Preprocessed3", ssh=True,␣
↪columns_to_drop=['peelFile','m_dThickness','traindevtest','dryFile'])
```

```
# Content of the dataframe:
df_features
```

1.0.2 Parameter setting selection

Use the parameters in the next section to adjust:

- **Data related parameters:** Select the target parameter, define the size of the test-set and select the normalization type.
- **Model creation parameters:** Define the number of input nodes, hidden layer nodes and output nodes. Set the activation function (typically relu is used) and define the size of the dropout (e.g. None, 0.2, 0.5) to prevent potential overfitting.
- **Model fitting parameters:** Define the amount of training epochs, loss function, optimizer, learning rate (lr), metrics to be reported in the epoch results and decide if you want the predicted results and the true values to be shown after each epoch.
- **Model saving parameters:** Define if you want the model to be saved and where it will be saved and in which format (SavedModel folder or .h5 file). This will be the model that will be used later to future training with larger dataset. So after saving you need to move this to the actual server which will do the future training using the web user interface.
- **Model stats saving parameters:** Define if you want the training model statistics to be saved and where to save the .csv file.

[5]:
```
# Data related parameters:
df_target='dryShrinkage'
test_size=0.2
normalization='standard' # 'standard' or 'minmax'

# Model creation parameters:
inputs = 84 # df_features.shape[1]
hidden_sizes = [256,128,64,32,16]
outputs = 1
activation = 'relu'
dropout = 0.2 # None or float

# Model fitting parameters:
EPOCHS = 5
loss = 'mse'
optimizer = tf.keras.optimizers.Adam
lr = 0.001
metrics = ['mae', 'mse', 'mape'] # Optional custom functions:␣
 ↪distance_from_true, precentage_from_true
with_predictions = False

# Model saving parameters:
save_model = False
path='./models/SavedModel'

# Model stats saving parameters:
save_to_csv=True
csv_path='./logs/'
```

```
# Do Not Touch This One: This variable stores the previous parameter settings.␣
↪The order of the parameters is critical.
parameters=(df_features, df_target, test_size, normalization, inputs,␣
↪hidden_sizes, outputs, activation, EPOCHS, loss, optimizer, lr, metrics,␣
↪dropout, with_predictions, save_model, path, csv_path)
```

1.0.3 Creating the model and training the neural network

The next line calls the function `create_and_fit` which creates the model and fits the model to the data during the training of the neural network. It returns the values for `model`, `history`, `train`, `test` and `model_name`. For more information use the `help`-function.

`%%time` tracks the duration of the processing time of the create_and_fit function.

```
%%time
model, history, train, test, model_name, y_train, y_test, train_norm, test_norm \
= create_and_fit(*parameters)
```

1.0.4 Saving and creating model_stats csv

The next line calls the function `model_stat` which creates the dataframe that summarizes the essential information of the model structure and the training parameters used during the training of the neural network. It also saves the model statistics dataframe into a .csv file if so configured in the parameter settings.

```
[8]: stats = model_stat(model, history, df_features, df_target, test_size,␣
     ↪normalization, inputs, hidden_sizes, outputs, activation, EPOCHS, loss,␣
     ↪optimizer, lr, metrics, dropout, train, test, csv_path, save_to_csv,␣
     ↪model_name)
```

```
# Content of the model_stat:
stats
```

1.0.5 Log file of the epoch results

The next line reads the .csv file created as a result of the `create_and_fit` function.

```
[11]: logs = pd.read_csv('./logs/2021-12-09 11:40:25.497703-log.csv')
```

```
# Content of the training epoch results log file
logs
```