Nguyen Ngoc Gia Thinh

Student ID: 103809954

COS30018

Task B4: Weekly Report

Deep Learning Model

For this version of the Task 4 I'm writing a function that takes as input the number of layers, the size of each layer, the layer name of LSTM, RNN and GRU. Also, I must write a hyperparameter configurations for each of the DL networks.

First of all, I have to research for libraries that contain all the layers that I needed and I came up with tensorflow and keras.

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, SimpleRNN, GRU, Dense, Dropout
```

Then, I come up with this function.

```
def create_dl_model(layer_type='LSTM', num_layers=3, layer_sizes=[50, 50, 50],
input_shape=(60, 1), dropout_rate=0.2):
```

In this function, the parameter will be

- Layer_type: Specifies the type of recurrent layer (LSTM, RNN or GRU) ; string
- Num_layers: Number of layes in the model ; int
- Layer_sizes: A list defining the number of neurons in each layer ; list
- Input_shape: Shape of the input data, (60,1 means 60 time steps and 1 feature per step) ; tuple
- Dropout_rate: Percentage of neurons randomly deactivated during training to prevent overfitting; float.

Then I created an empty sequential model where layers will be added one by one.

After that, I define a dictionary mapping the layer type to the corresponding Keras layer and ensure the user selects a valid layer type.

```
layer_dict = {'LSTM': LSTM, 'RNN': SimpleRNN, 'GRU': GRU}
    if layer_type not in layer_dict:
        raise ValueError("Invalid layer type. Choose from 'LSTM', 'RNN', or
'GRU'.")
    Layer = layer_dict[layer_type]
```

Then I add the first layer by writing this function

```
model.add(Layer(units=layer_sizes[0], return_sequences=(num_layers > 1),
input_shape=input_shape))
    model.add(Dropout(dropout_rate))
```

In this function, I added a dropout_rate to randomly disables neurons to reduce overfitting.

Then I added a hidden layers to the function because it can give a more accurate and better learning for the machine learning process.

```
for i in range(1, num_layers - 1):
        model.add(Layer(units=layer_sizes[i], return_sequences=True))
        model.add(Dropout(dropout_rate))
```

Then I added the final recurrent layer

```
if num_layers > 1:
        model.add(Layer(units=layer_sizes[-1]))
        model.add(Dropout(dropout_rate))
```

To fully connected layer with 1 neuron as I am predicting the stock price I added an output layer to the function.

```
model.add(Dense(units=1))
```

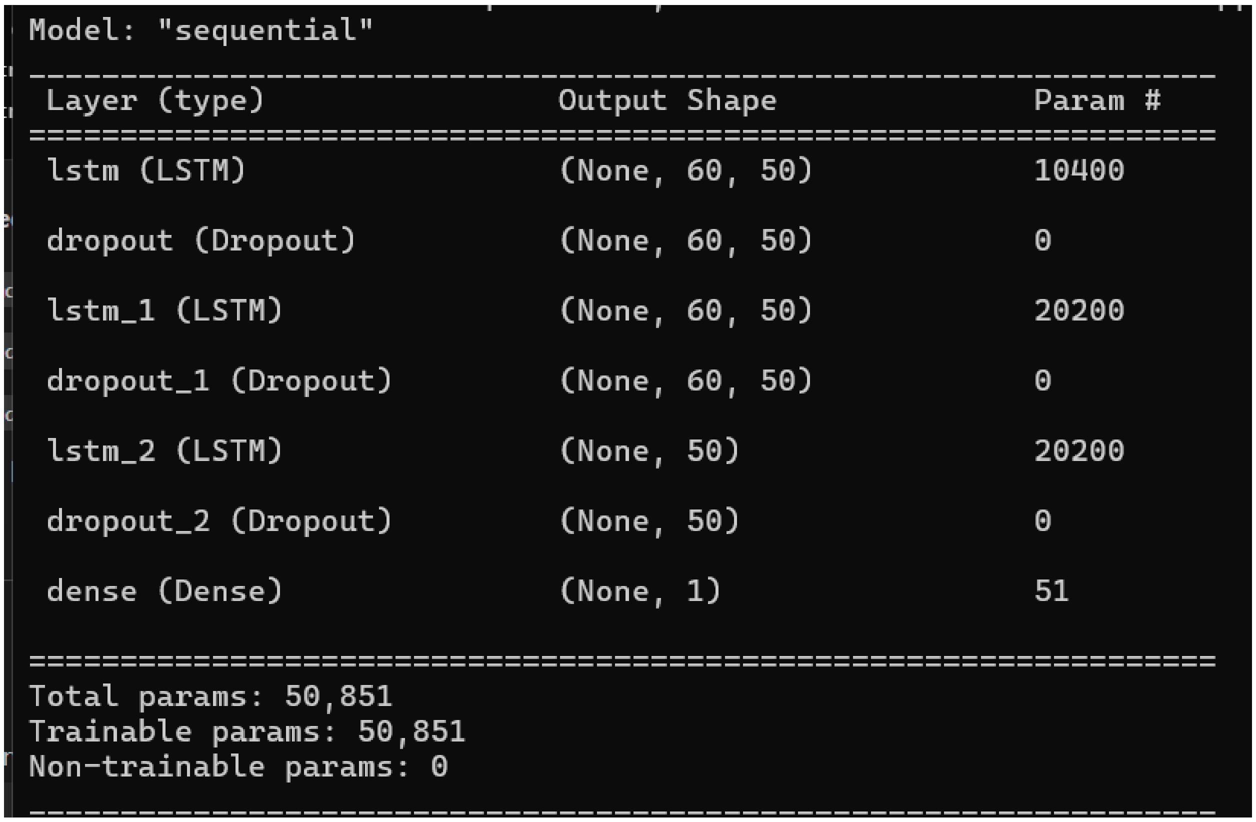
Then I compiling the model using adam optimizer which adapts learning rates dynamically. Also mean squared error are commonly used for regression tasks.

```
model.compile(optimizer='adam', loss='mean_squared_error')
    return model
```

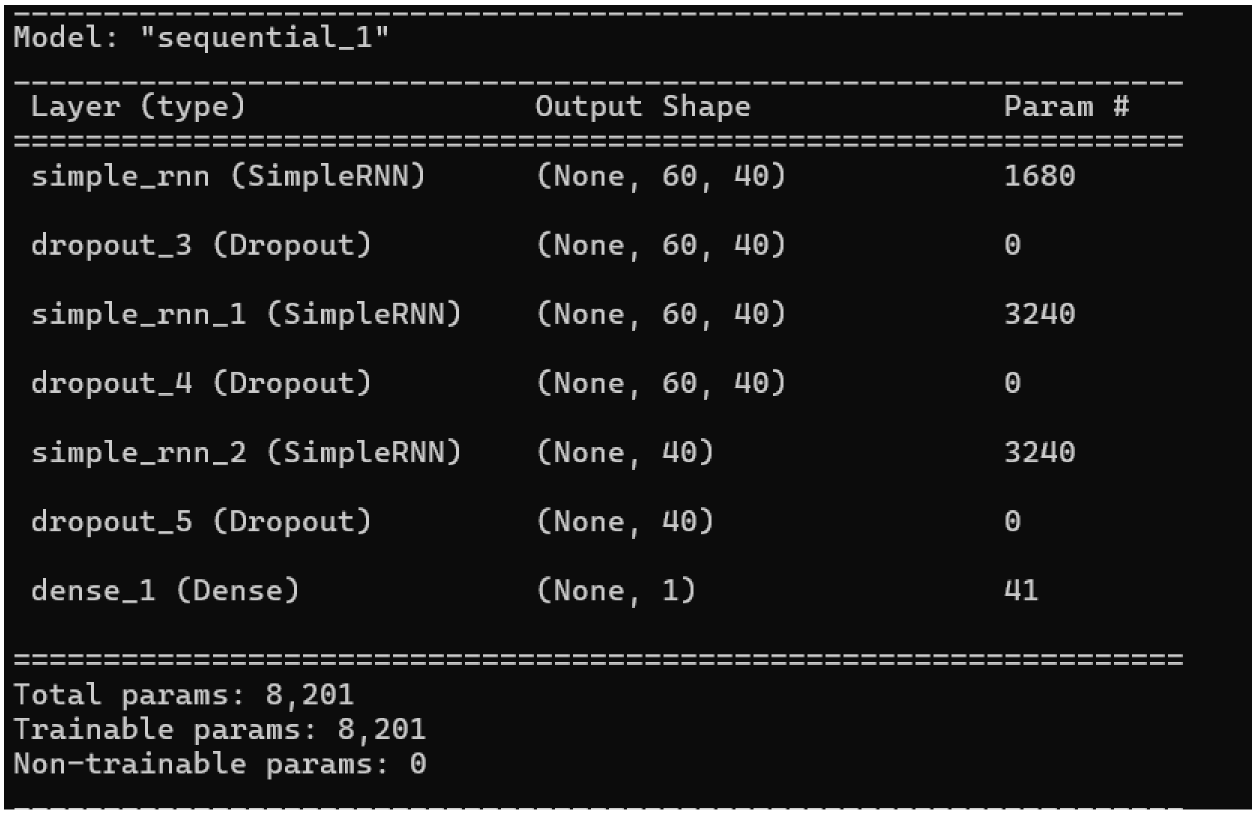
Then I run all three different architectures.

```
lstm_model = create_dl_model(layer_type='LSTM', num_layers=3, layer_sizes=[50,
50, 50])
rnn_model = create_dl_model(layer_type='RNN', num_layers=3, layer_sizes=[40,
40, 40])
gru_model = create_dl_model(layer_type='GRU', num_layers=3, layer_sizes=[60,
60, 60])
```

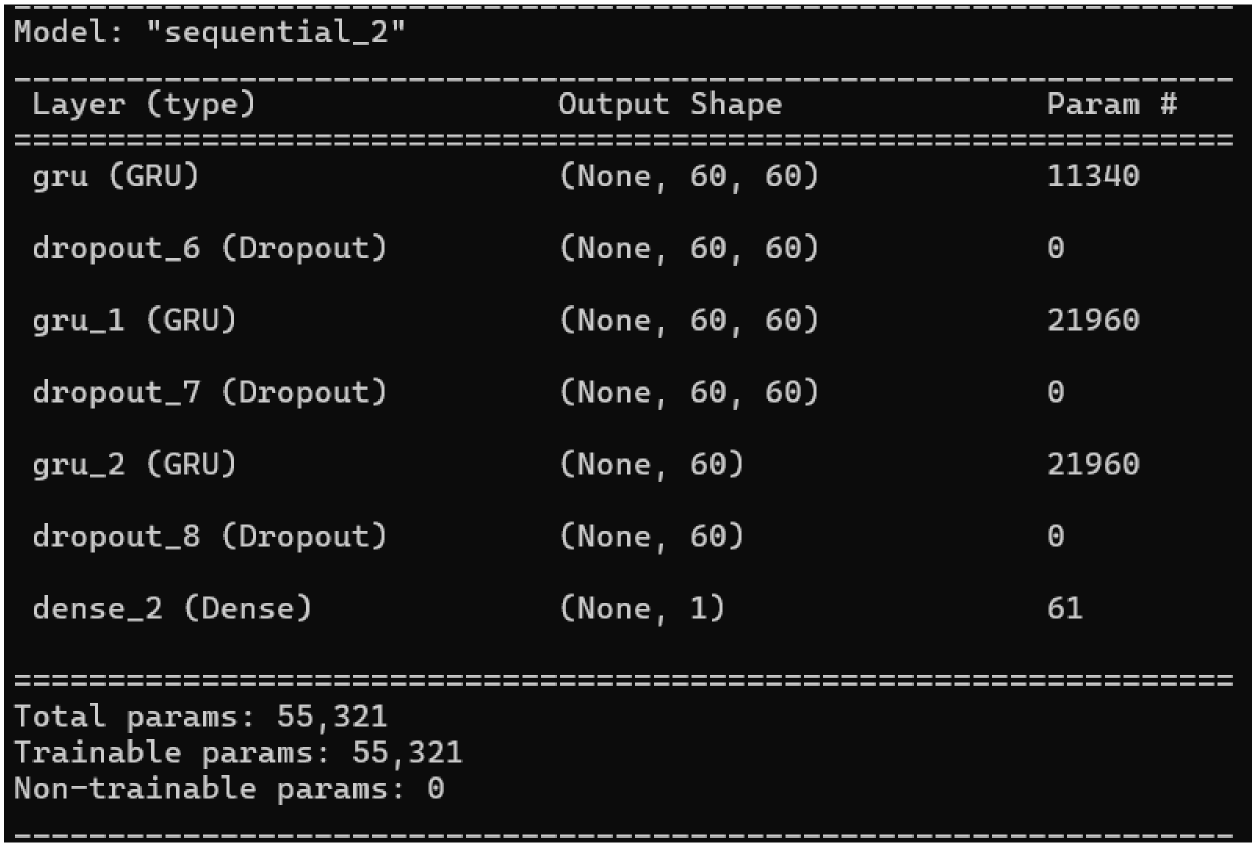
```
lstm_model.summary()
rnn_model.summary()
gru_model.summary()
```



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 60, 50)            10400

 dropout (Dropout)           (None, 60, 50)            0

 lstm_1 (LSTM)               (None, 60, 50)            20200

 dropout_1 (Dropout)         (None, 60, 50)            0

 lstm_2 (LSTM)               (None, 50)                20200

 dropout_2 (Dropout)         (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```



```
_________________________________________________________________
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 60, 40)            1680

 dropout_3 (Dropout)         (None, 60, 40)            0

 simple_rnn_1 (SimpleRNN)    (None, 60, 40)            3240

 dropout_4 (Dropout)         (None, 60, 40)            0

 simple_rnn_2 (SimpleRNN)    (None, 40)                3240

 dropout_5 (Dropout)         (None, 40)                0

 dense_1 (Dense)             (None, 1)                 41

=================================================================
Total params: 8,201
Trainable params: 8,201
Non-trainable params: 0
_________________________________________________________________
```



```
_________________________________________________________________
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 60, 60)            11340

 dropout_6 (Dropout)         (None, 60, 60)            0

 gru_1 (GRU)                 (None, 60, 60)            21960

 dropout_7 (Dropout)         (None, 60, 60)            0

 gru_2 (GRU)                 (None, 60)                21960

 dropout_8 (Dropout)         (None, 60)                0

 dense_2 (Dense)             (None, 1)                 61

=================================================================
Total params: 55,321
Trainable params: 55,321
Non-trainable params: 0
_________________________________________________________________
```