Model over-view and parameter calculation:

Ripunjoy Gohain

In [1]:
```
from tensorflow.keras.models import model_from_json
```

In [2]:
```
# Model 1
# load json and create model
with open("./hourly_models/model_1/model.json", "r") as json_file:
    model_json = json_file.read()
model = model_from_json(model_json)
```

```
WARNING:tensorflow:From c:\users\ripunjoy gohain\appdata\local\programs\python\python37\lib\site-packages\tensorflow\python\ops\resource_variable_ops.py:435: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

In [3]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100)               40800
_________________________________________________________________
dense (Dense)                (None, 48)                4848
=================================================================
Total params: 45,648
Trainable params: 45,648
Non-trainable params: 0
_________________________________________________________________
```

This is simle vanilla LSTM model. (LSTM has 4 gates, forget, update, memory cell/input, output gates).

Input Layer:

1. 48 timestemps, and 1 feature each time stemps

Layer 1: LSTM

1. Input timestemps is 48 & output is also 48. There is only 1 feature each time step.
2. Input shape is [batch_size=None, n_timestemps=48,n_features=1]
3. The first hidden layer have 100 LSTM units (cells). We have only one layer in this Network.
4. So at t-th timestemp, it will get t-1 th time-stemps memory cells (which is 100, as defined in units).
5. It will also get 1 features at that time stemp.
6. So, for 1 gate. The number of parameters to learn will be (100 memory cells-previous + 1 feature + 1 bias) * 100 memory cells-to next. Which is 10200.
7. As it has 4 gates, total parameters to learn will be 10200 * 4 = 40800.
8. These 40800 parameters will be shared across all the 48 timestemps (common parameters they will try to learn).

Output Layer:

1. In Output, we need 48 outputs. LSTM layer will provide 100 memory cells for each timestamps (the parameters are shared, but the value will be different as every time-stemp have different values).
2. We have dense-ly connected LSTM cells to outputs (48). 1 for each timestemp.
3. So one timestemp need to learn (100 memory cells + 1 bias) * (1_output) = 101 weight parameters.
4. For 48 timestemps. 101 * 48 = 4848.

In [4]:
```
with open("./hourly_models/model_2/model.json", "r") as json_file:
    model_json = json_file.read()
model2 = model_from_json(model_json)
```

In [5]:
```
model2.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 100)               40800
_________________________________________________________________
repeat_vector (RepeatVector) (None, 48, 100)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 48, 100)           80400
_________________________________________________________________
time_distributed (TimeDistri (None, 48, 50)            5050
_________________________________________________________________
time_distributed_1 (TimeDist (None, 48, 1)             51
=================================================================
Total params: 126,301
Trainable params: 126,301
Non-trainable params: 0
_________________________________________________________________
```

Model 2: LSTM encoder decoder univariate input

When return sequence is true, in output we can see the sequence also (batch, sequence, memory cells)

Input Layer:

1. Univariate (1 feature) for 48 time-stemps.

LSTM Encoder:

1. 100 memory cells.
2. So parameter to learn in LSTM1 is (100 + 1 + 1) * 100 * 4 = 40800.
3. For encoder-decoder, input is squashed into a single feature vector (40800 parameters to learn), if we want the output to regenerate the same dimension as the original input, we can "artificially" convert this feature tensor from 1D into 2D by replicating it using RepeatVector().

LSTM Decoder:

1. 100 memory cells (units).
2. From encoder, the number of features at every time-stemp become 100. (unlike 1 in encoder input).
3. So, 100 decoder memory cells from previous layer + 100 features from encoder + 1 bias connected to 100 decoder memory cells.
4. (100 + 100 + 1) * 100 * 4 (gates) = 80400

Time Distributed Layer1:

1. output 50 dense units.
2. From previous decoder we have 100 memory cells, wich will be connected to 50 dense time distributed unit.
3. Unlike simple dense for each layer of output (model 1). These parameters will be shared across 48 output timestemps.
4. (100 cells + 1 bias) * 50 = 5050

Time Distributed Layer2:

1. output 1 dense units.
2. 50 units from above layer.
3. (50 + 1) * 1 = 51. (shared across all 48 outputs)

In [6]:
```
with open("./hourly_models/model_4/model.json", "r") as json_file:
    model_json = json_file.read()
model4 = model_from_json(model_json)
```

In [7]:
```
model4.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm (LSTM)                  (None, 200)               167200
_________________________________________________________________
repeat_vector (RepeatVector) (None, 48, 200)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 48, 200)           320800
_________________________________________________________________
time_distributed (TimeDistri (None, 48, 100)           20100
_________________________________________________________________
time_distributed_1 (TimeDist (None, 48, 1)             101
=================================================================
Total params: 508,201
Trainable params: 508,201
Non-trainable params: 0
_________________________________________________________________
```

Model 4: LSTM encoder decoder multivariate output.

Input:

1. 72 time-stemps (3 days). Each time stemps having 100 features.

Encoder:

1. 200 memory cells.
2. (200 prev memory cells + 8 features + 1 bias) * 200 this momory cells * 4 gates = 167200
3. RepeatVector 48 times to get 48 outputs.

Decoder:

1. 200 memory cells.
2. 200 features from encoder.
3. (200 prev + 200 feat encoder + 1 bias) * 200 this * 4 gates = 320800

Time Distributed Layer 1:

1. 100 dense output shared across.
2. (200 from decoder + 1) * 100 = 20100

Time Distributed Layer 2:

1. 1 output shared across 48 sequence (timestemps).
2. (100 from prev dense + 1) * 1 = 101

In [8]:
```
with open("./hourly_models/model_5/model.json", "r") as json_file:
    model_json = json_file.read()
model5 = model_from_json(model_json)
```

In [9]:
```
model5.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 70, 64)            1600
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 35, 64)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 33, 64)            12352
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 16, 64)            0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
```

```
_________________________________________________________________
repeat_vector (RepeatVector) (None, 48, 1024)          0
_________________________________________________________________
lstm (LSTM)                  (None, 48, 200)           980000
_________________________________________________________________
time_distributed (TimeDistri (None, 48, 100)           20100
_________________________________________________________________
time_distributed_1 (TimeDist (None, 48, 1)             101
=================================================================
Total params: 1,014,153
Trainable params: 1,014,153
Non-trainable params: 0
_________________________________________________________________
```

CNN-LSTM model with Multivariate input

Input:

1. 72 timestemps each having 8 features. (72,8) or in 3d (1,72,8)

Conv1D Layer 1:

1. kernel size 3, basicall 1*3.
2. 64 filters.
3. each filter needs, 3 * 8 features + 1 bias to learn = 25 parameters.
4. Total 64 * 25 = 1600 parameters.
5. Output will be, there is no zero padding. stride is 1. So, ((72 input+2*0) - 3 filter size)/ 1 stride + 1 = 70 columns and as have 64 filters. So (70, 64).

MaxPool1D:

1. 2*2 filter.
2. Channels will be same. (64)
3. Stride is default to pool_size, that is 2.
4. So, output will be (70/2, 64) = (35,64).
5. No parameters to learn.

Conv1D Layer 2:

1. Input is out put of previous (35, 64).
2. kernel size 3.
3. 64 filters.
4. 1 filter needs, (3 kernel * 64 input channels + 1 bias) = 68 param.
5. Total, (3 * 64 + 1) * 64 = 12352.
6. output, (33, 64)

MaxPool1D:

1. Input from previous.
2. Output (16, 64)

Flatten:

1. 16 * 64 = 1024

Repeat Vecotor to 48 output times

LSTM Layer 1:

1. 200 memory cells.
2. Input is 1024.
3. (200 memory in + 1024 feature in + 1 bias) * 200 memory out * 4 gates = 980000 shared across 48 output repeat vectors

Time distributed dense:

1. Input 200 memory cells.
2. 100 dense units.
3. (200 + 1) * 100 = 20100 shared across.

Time distributed dense:

1. Input 100 dense
2. Output 1
3. (100 + 1) * 1 = 101

In [10]:
```
with open("./hourly_models/model_6/model.json", "r") as json_file:
    model_json = json_file.read()
model6 = model_from_json(model_json)
```

In [11]:
```
model6.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv_lst_m2d_3 (ConvLSTM2D)  (None, 1, 24, 64)         55552
_________________________________________________________________
flatten_3 (Flatten)          (None, 1536)              0
_________________________________________________________________
repeat_vector_3 (RepeatVecto (None, 48, 1536)          0
_________________________________________________________________
lstm_3 (LSTM)                (None, 48, 200)           1389600
_________________________________________________________________
time_distributed_6 (TimeDist (None, 48, 100)           20100
_________________________________________________________________
time_distributed_7 (TimeDist (None, 48, 1)             101
=================================================================
Total params: 1,465,353
Trainable params: 1,465,353
Non-trainable params: 0
_________________________________________________________________
```

ConvLSTM model with Multivariate Input

Input:

1. Converted data to have 3 timestemps, 1 row, 24 columns, and 8 features. (72 data points are divided into 3 days, each day 1 time stemps, each day have 24 hours, ie columns, 8 features ie. channels)

ConvLSTM2D: Encoder

1. Key point is if filter is 1, ConvLSTM2D is just LSTM.
2. Kernel filter size 1 * 3 & input is (1 row is there, 24 columns, 8 features)
3. same padding, 1*24 size wil remain
4. 1 kernel parameters, (3 filter size * 8 features + 1 bias) = 25
5. number of parameters = 4 * output channels * (filter-row * filter-column * (input channels + output channels) + 1)
6. [1 * 3 filter * (8 input + 64 output) + 1] * 64 output channels * 4 gates = 55552. _For general LSTM (input units + input feature + 1 bias) * output units
7. All 55552 parameters will be shared across 3 time-stemps.
8. Ouput shape: (1, 24, 64)

Flatten:

1. Flat shape = 24 * 64 = 1536

Repeat num_output times:48

Decoder LSTM:

1. input 1536.
2. 200 cells.
3. (200 cells + 1536 input + 1 bias) * 200 output * 4 gates = 1389600

Time distributed dense:

1. 100 dense units.
2. Input 200 LSTM cells.
3. (200 + 1) * 100 = 20100 shared across 48 time-stemps

Time distributed dense:

1. 1 dense unit.
2. Input 100 dense unit.
3. (100 + 1) * 1 = 101