Animation (Predicting Animation Sequences with LSTM) Part 2

Author: Benjamin Kenwright

Continuation on the tutorial that focuses on solving animation sequence problems using LSTMs.

Animation prediction is a classic example of one-to-many sequence problems where you have a single animation as input and you have to predict the following animation output in the form (i.e., prediction for the next X seconds, where input is the movement of the previous Y seconds - a classic example of many-to-many sequence problems).

You'll learn about the basic concepts around one-to-many and many-to-many problems. However, the concepts learned in this tutorial are for laying the foundation for solving advanced animation sequence problems, such as animation prediction and automated motion classification.

One-to-Many Sequence Problems

One-to-many animation sequence problems are the type of sequence problems where input data has one time-step and the output contains a vector of multiple values or multiple time-steps.

You'll see how to solve one-to-many sequence problems where the input has a single feature. You will then move on to see how to work with multiple features input to solve one-to-many sequence problems.

One-to-Many Sequence Problems with a Single Feature

Create a simple test dataset

```
X = []
Y = []
X = [x+3 for x in range(-2, 43, 3)]

for i in X:
    output_vector = []
    output_vector.append(i+1)
    output_vector.append(i+2)
    Y.append(output_vector)

print(X)
print(Y)
```

```
[1, 4, 7, 10, 13, 16, 19, 22, 25, 28, 31, 34, 37, 40, 43]
[[2, 3], [5, 6], [8, 9], [11, 12], [14, 15], [17, 18], [20, 21], [23, 24], [26, 27], [29, 30],
```

Your input will contains 15 samples with one time-step and one feature. For each value in the input sample, the corresponding output vector contains the next two integers.

For instance, if the input is 4, the output vector will contain values 5 and 6. Hence, the problem is a simple

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.layers import Bidirectional
import tensorflow as tf
import numpy as np
import math

X = []
Y = []
X = [x+3 for x in range(-2, 43, 3)]

for i in X:
    output_vector = []
    output_vector.append(i+1)
    output_vector.append(i+2)
    Y.append(output_vector)

#print(X)
#print(Y)

X = np.array(X).reshape(15, 1, 1)
Y = np.array(Y)

model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(1, 1)))
model.add(Dense(2))
model.compile(optimizer='adam', loss='mse')

model.fit(X, Y, epochs=20, validation_split=0.2, batch_size=3, verbose=0)

test_input = np.array([5])
test_input = test_input.reshape((1, 1, 1))
test_output = model.predict(test_input, verbose=0)
print('output is:', test_output)
```

```
     output is: [[0.89800316 0.53350645]]
```

Stacked LSTM

Taking the example further....

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.layers import Bidirectional
import tensorflow as tf
import numpy as np
import math

X = []
Y = []
X = [x+3 for x in range(-2, 43, 3)]

for i in X:
    output_vector = []
    output_vector.append(i+1)
    output_vector.append(i+2)
    Y.append(output_vector)

#print(X)
#print(Y)

X = np.array(X).reshape(15, 1, 1)
Y = np.array(Y)

model = Sequential()
model.add(LSTM(50, activation='relu', return_sequences=True, input_shape=(1, 1)))
model.add(LSTM(50, activation='relu'))
model.add(Dense(2))
model.compile(optimizer='adam', loss='mse')

model.fit(X, Y, epochs=20, validation_split=0.2, batch_size=3, verbose=0)

test_input = np.array([5])
test_input = test_input.reshape((1, 1, 1))
test_output = model.predict(test_input, verbose=0)
print('output is:', test_output)
```

```
output is: [[1.2432528 1.3007957]]
```

Bidirectional LSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.layers import Bidirectional
import tensorflow as tf
import numpy as np
import math

X = []
Y = []
X = [x+3 for x in range(-2, 43, 3)]

for i in X:
    output_vector = []
    output_vector.append(i+1)
    output_vector.append(i+2)
    Y.append(output_vector)

#print(X)
#print(Y)

X = np.array(X).reshape(15, 1, 1)
Y = np.array(Y)

model = Sequential()
model.add(Bidirectional(LSTM(50, activation='relu'), input_shape=(1, 1)))
model.add(Dense(2))
model.compile(optimizer='adam', loss='mse')
model.fit(X, Y, epochs=20, validation_split=0.2, batch_size=3, verbose=0)

test_input = np.array([5])
test_input = test_input.reshape((1, 1, 1))
test_output = model.predict(test_input, verbose=0)
print('output is:', test_output)
```

```
output is: [[1.1110531 1.2886944]]
```

One-to-Many Sequence Problems with Multiple Features

You'll now see how to develop a one-to-many sequence problems where input samples will have one time-step, but two features. The output will be a vector of two elements.

```
# Dataset
nums = 25

X1 = list()
X2 = list()
X = list()
Y = list()

X1 = [(x+1)*2 for x in range(25)]
X2 = [(x+1)*3 for x in range(25)]

for x1, x2 in zip(X1, X2):
    output_vector = list()
    output_vector.append(x1+1)
    output_vector.append(x2+1)
    Y.append(output_vector)

X = np.column_stack((X1, X2))
print(X)
```

```
[[ 2  3]
 [ 4  6]
 [ 6  9]
 [ 8 12]
 [10 15]
 [12 18]
 [14 21]
 [16 24]
 [18 27]
 [20 30]
 [22 33]
 [24 36]
 [26 39]
 [28 42]
 [30 45]
 [32 48]
 [34 51]
 [36 54]
 [38 57]
 [40 60]
 [42 63]
 [44 66]
 [46 69]
 [48 72]
 [50 75]]
```

You can see each input time-step consists of two features. The output will be a vector which contains the next two elements that correspond to the two features in the time-step of the input sample. For instance, for the input sample `[2, 3]`, the output will be `[3, 4]`, and so on.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.layers import Bidirectional
import tensorflow as tf
import numpy as np
import math

nums = 25

X1 = list()
X2 = list()
X = list()
Y = list()

X1 = [(x+1)*2 for x in range(25)]
X2 = [(x+1)*3 for x in range(25)]

for x1, x2 in zip(X1, X2):
    output_vector = list()
    output_vector.append(x1+1)
    output_vector.append(x2+1)
    Y.append(output_vector)

X = np.column_stack((X1, X2))
#print(X)

X = np.array(X).reshape(25, 1, 2)
Y = np.array(Y)

model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(1, 2)))
model.add(Dense(2))
model.compile(optimizer='adam', loss='mse')

model.fit(X, Y, epochs=20, validation_split=0.2, batch_size=3, verbose=0)

test_input = np.array([30,50])
test_input = test_input.reshape((1, 1, 2))
test_output = model.predict(test_input, verbose=0)
print('output is:', test_output)
```

```
output is: [[33.326694 44.129814]]
```

Stacked LSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.layers import Bidirectional
import tensorflow as tf
import numpy as np
import math

nums = 25

X1 = list()
X2 = list()
X = list()
Y = list()

X1 = [(x+1)*2 for x in range(25)]
X2 = [(x+1)*3 for x in range(25)]

for x1, x2 in zip(X1, X2):
    output_vector = list()
    output_vector.append(x1+1)
    output_vector.append(x2+1)
    Y.append(output_vector)

X = np.column_stack((X1, X2))
#print(X)

X = np.array(X).reshape(25, 1, 2)
Y = np.array(Y)

model = Sequential()
model.add(LSTM(50, activation='relu', return_sequences=True, input_shape=(1, 2)))
model.add(LSTM(50, activation='relu'))
model.add(Dense(2))
model.compile(optimizer='adam', loss='mse')

model.fit(X, Y, epochs=20, validation_split=0.2, batch_size=3, verbose=0)

test_input = np.array([30,50])
test_input = test_input.reshape((1, 1, 2))
test_output = model.predict(test_input, verbose=0)
print('output is:', test_output)
```

```
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>
output is: [[32.31493 49.09465]]
```

Bidirectional LSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.layers import Bidirectional
import tensorflow as tf
import numpy as np
import math
```

```
nums = 25

X1 = list()
X2 = list()
X = list()
Y = list()

X1 = [(x+1)*2 for x in range(25)]
X2 = [(x+1)*3 for x in range(25)]

for x1, x2 in zip(X1, X2):
    output_vector = list()
    output_vector.append(x1+1)
    output_vector.append(x2+1)
    Y.append(output_vector)

X = np.column_stack((X1, X2))
#print(X)

X = np.array(X).reshape(25, 1, 2)
Y = np.array(Y)

model = Sequential()
model.add(Bidirectional(LSTM(50, activation='relu'), input_shape=(1, 2)))
model.add(Dense(2))
model.compile(optimizer='adam', loss='mse')

model.fit(X, Y, epochs=20, validation_split=0.2, batch_size=3, verbose=0)

test_input = np.array([30,50])
test_input = test_input.reshape((1, 1, 2))
test_output = model.predict(test_input, verbose=0)
print('output is:', test_output)
```

```
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<locals>
output is: [[32.770287 49.239826]]
```

Many-to-Many Sequence Problems

In one-to-many and many-to-one sequence problems, you saw that the output vector can contain multiple values. Depending upon the problem, an output vector containing multiple values can be considered as having single (since the output contains one time-step data in strict terms) or multiple (since one vector contains multiple values) outputs.

However, in some sequence problems, you want multiple outputs divided over time-steps. In other words, for each time-step in the input, you want a corresponding time-step in the output. Such models can be used to solve many-to-many sequence problems with variable lengths.

Many-to-Many Sequence Problems with Single Feature

The input X contains 20 samples where each sample contains 3 time-steps with one feature.

```
# Dataset
X = list()
Y = list()
X = [x for x in range(5, 301, 5)]
Y = [y for y in range(20, 316, 5)]

X = np.array(X).reshape(20, 3, 1)
Y = np.array(Y).reshape(20, 3, 1)
```

You can see that the input sample contain 3 values that are basically 3 consecutive multiples of 5.

The output contains the next three consecutive multiples of 5. You can see the output in this case is different than what we have seen in the previous sections. The output should also be converted into a 3D format containing the number of samples, time-steps, and features.

You have created our dataset; the next step is to train our models. As usual you'll train stacked LSTM and bidirectional LSTM models.

Stacked LSTM

```
from tensorflow.keras.layers import RepeatVector
from tensorflow.keras.layers import TimeDistributed
import tensorflow as tf
import numpy as np
import math

model = Sequential()

# encoder layer
model.add(LSTM(100, activation='relu', input_shape=(3, 1)))

# repeat vector
model.add(RepeatVector(3))

# decoder layer
model.add(LSTM(100, activation='relu', return_sequences=True))

model.add(TimeDistributed(Dense(1)))
model.compile(optimizer='adam', loss='mse')

print(model.summary())
```

```
Model: "sequential_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_12 (LSTM)               (None, 100)               40800
_________________________________________________________________
repeat_vector (RepeatVector) (None, 3, 100)            0
_________________________________________________________________
lstm_13 (LSTM)               (None, 3, 100)            80400
_________________________________________________________________
time_distributed (TimeDistri (None, 3, 1)              101
=================================================================
Total params: 121,301
```

```
Trainable params: 121,301
Non-trainable params: 0
_________________________________________________________________
None
```

```
from tensorflow.keras.layers import RepeatVector
from tensorflow.keras.layers import TimeDistributed
import tensorflow as tf
import numpy as np
import math

X = list()
Y = list()
X = [x for x in range(5, 301, 5)]
Y = [y for y in range(20, 316, 5)]

X = np.array(X).reshape(20, 3, 1)
Y = np.array(Y).reshape(20, 3, 1)


model = Sequential()
model.add(LSTM(100, activation='relu', input_shape=(3, 1)))
model.add(RepeatVector(3))
model.add(LSTM(100, activation='relu', return_sequences=True))
model.add(TimeDistributed(Dense(1)))
model.compile(optimizer='adam', loss='mse')
print(model.summary())

model.fit(X, Y, epochs=20, validation_split=0.2, batch_size=3, verbose=0)

test_input = np.array([300, 305, 310])
test_input = test_input.reshape((1, 3, 1))
test_output = model.predict(test_input, verbose=0)
print(test_output)
```

```
Model: "sequential_12"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_20 (LSTM)               (None, 100)               40800
_________________________________________________________________
repeat_vector_4 (RepeatVecto (None, 3, 100)            0
_________________________________________________________________
lstm_21 (LSTM)               (None, 3, 100)            80400
_________________________________________________________________
time_distributed_4 (TimeDist (None, 3, 1)              101
=================================================================
Total params: 121,301
Trainable params: 121,301
Non-trainable params: 0
_________________________________________________________________
None
[[[312.0389]
  [321.6118]
  [328.943 ]]]
```

Bidirectional LSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.layers import RepeatVector
from tensorflow.keras.layers import TimeDistributed
import tensorflow as tf
import numpy as np
import math

X = list()
Y = list()
X = [x for x in range(5, 301, 5)]
Y = [y for y in range(20, 316, 5)]

_X = np.array(X).reshape(20, 3, 1)
_Y = np.array(Y).reshape(20, 3, 1)

X = tf.cast(_X, dtype='float64')
Y = tf.cast(_Y, dtype='float64')

model = Sequential()
model.add(Bidirectional(LSTM(100, activation='relu', input_shape=(3, 1))))
model.add(RepeatVector(3))
model.add(Bidirectional(LSTM(100, activation='relu', return_sequences=True) ) )
model.add(TimeDistributed(Dense(1)))
model.compile(optimizer='adam', loss='mse')
# print(model.summary())

model.fit(X, Y, epochs=20, validation_split=0.2, batch_size=3, verbose=0)

test_input = np.array([300, 305, 310])
test_input = test_input.reshape((1, 3, 1))
test_output = model.predict(test_input, verbose=0)
print(test_output)
```

```
[[[315.71298]
  [319.43057]
  [325.48364]]]
```

Many-to-Many Sequence Problems with Multiple Features

In many-to-many sequence problems, each time-step in the input sample contains multiple features.

```
# Dataset
X = list()
Y = list()
X1 = [x1 for x1 in range(5, 301, 5)]
X2 = [x2 for x2 in range(20, 316, 5)]
Y = [y for y in range(35, 331, 5)]

X = np.column_stack((X1, X2))
```

Stacked LSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.layers import RepeatVector
from tensorflow.keras.layers import TimeDistributed
import tensorflow as tf
import numpy as np
import math

X = []
Y = []
X1 = [x1 for x1 in range(5, 301, 5)]
X2 = [x2 for x2 in range(20, 316, 5)]
Y  = [y for y in range(35, 331, 5)]

X = np.column_stack((X1, X2))

X = np.array(X).reshape(20, 3, 2)
Y = np.array(Y).reshape(20, 3, 1)

model = Sequential()
model.add(LSTM(100, activation='relu', input_shape=(3, 2)))
model.add(RepeatVector(3))
model.add(LSTM(100, activation='relu', return_sequences=True))
model.add(TimeDistributed(Dense(1)))
model.compile(optimizer='adam', loss='mse')
# print(model.summary())

model.fit(X, Y, epochs=20, validation_split=0.2, verbose=0, batch_size=3)

X1 = [300, 305, 310]
X2 = [315, 320, 325]
test_input = np.column_stack((X1, X2))
test_input = test_input.reshape((1, 3, 2))
print('test output:', test_input)
```

```
test output: [[[300 315]
  [305 320]
  [310 325]]]
```

Bidirectional LSTM

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.layers import RepeatVector
from tensorflow.keras.layers import TimeDistributed
import tensorflow as tf
import numpy as np
import math

X = []
Y = []
```

```
X1 = [x1 for x1 in range(5, 301, 5)]
X2 = [x2 for x2 in range(20, 316, 5)]
Y  = [y for y in range(35, 331, 5)]

X = np.column_stack((X1, X2))

_X = np.array(X).reshape(20, 3, 2)
_Y = np.array(Y).reshape(20, 3, 1)

X = tf.cast(_X, dtype='float64')
Y = tf.cast(_Y, dtype='float64')

model = Sequential()
model.add(Bidirectional(LSTM(100, activation='relu', input_shape=(3, 2))))
model.add(RepeatVector(3))
model.add(Bidirectional(LSTM(100, activation='relu', return_sequences=True)))
model.add(TimeDistributed(Dense(1)))
model.compile(optimizer='adam', loss='mse')
# print(model.summary())

model.fit(X, Y, epochs=20, validation_split=0.2, verbose=0, batch_size=3)

X1 = [300, 305, 310]
X2 = [315, 320, 325]
test_input = np.column_stack((X1, X2))
test_input = test_input.reshape((1, 3, 2))
print('test output:', test_input)
```

```
test output: [[[300 315]
  [305 320]
  [310 325]]]
```

Conclusion

In this tutorial, you saw how to solve one-to-many and many-to-many animation sequence problems in LSTM. You also saw how to predict multi-step outputs.