

model.py

```
1  from tensorflow.keras.models import Model
2  from tensorflow.keras.layers import LSTM, Bidirectional, TimeDistributed, Dense, RepeatVector, ReLU
3  import tensorflow as tf
4
5
6  class FCEncoder(Model):
7      def __init__(self, n_features):
8          super().__init__()
9          self.FC = []
10         for level, n_feature in enumerate(n_features):
11             if level == 0:
12                 pass
13             elif level == len(n_features) - 1:
14                 dense = Dense(n_feature, name=f"encoder{level}")
15                 self.FC.append(dense)
16                 relu = ReLU(name=f"activation{level}")
17                 self.FC.append(relu)
18
19             else:
20                 dense = Dense(n_feature, name=f"encoder{level}")
21                 self.FC.append(dense)
22                 relu = ReLU(name=f"activation{level}")
23                 self.FC.append(relu)
24
25         def call(self, x):
26             out = x
27             for layer in self.FC:
28                 out = layer(out)
29                 # print(layer.name, out.shape)
30             return out
31
32
33  class FCDecoder(Model):
34      def __init__(self, n_features):
35          super().__init__()
36          self.FC = []
37         for level, n_feature in enumerate(n_features):
38             if level == 0:
39                 pass
40             elif level == len(n_features) - 1:
41                 dense = Dense(n_feature, name=f"decoder{level}")
42                 self.FC.append(dense)
43             else:
44                 dense = Dense(n_feature, name=f"decoder{level}")
45                 self.FC.append(dense)
46                 relu = ReLU(name=f"activation{level}")
47                 self.FC.append(relu)
48
49         def call(self, x):
50             out = x
51             for layer in self.FC:
52                 out = layer(out)
53                 # print(layer.name, out.shape)
54             return out
55
56
57  class AE(Model):
58      def __init__(self, n_features):
59          super().__init__()
60          self.encoder = FCEncoder(n_features)
61          n_features.reverse()
62          self.decoder = FCDecoder(n_features)
63          self.decoder2 = FCDecoder(n_features)
64
65         def call(self, x):
66             z = self.encoder(x)
67             w1 = self.decoder(z)
68             w2 = self.decoder2(z)
```

```
69         w3 = self.decoder2(self.encoder(w1))
70
71     return w1, w2, w3
72
```