Week 9 Meeting note

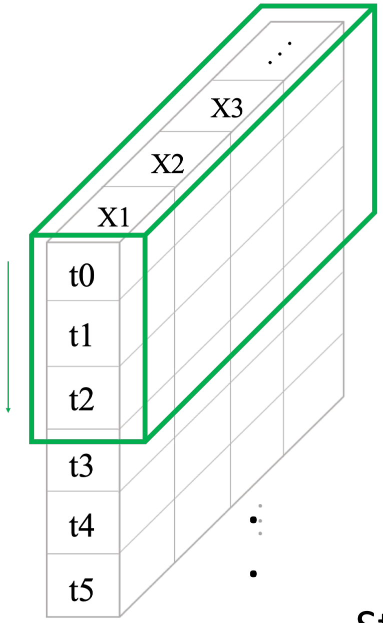
Stateful and Stateless Time series forecasting

Stateful connections (state) between sequences(samples) would be retained. It can enable the model to learn the timing characteristics between samples. Which sample is in the first place and which sample is in the second place will have an impact on the model.

Stateless during training, the state is reset after each sequence, state isn't retained between the sequences(samples). Which can be said to be independent of each sample, with no contextual relationship between them.

Stateful forecasting

Original Data

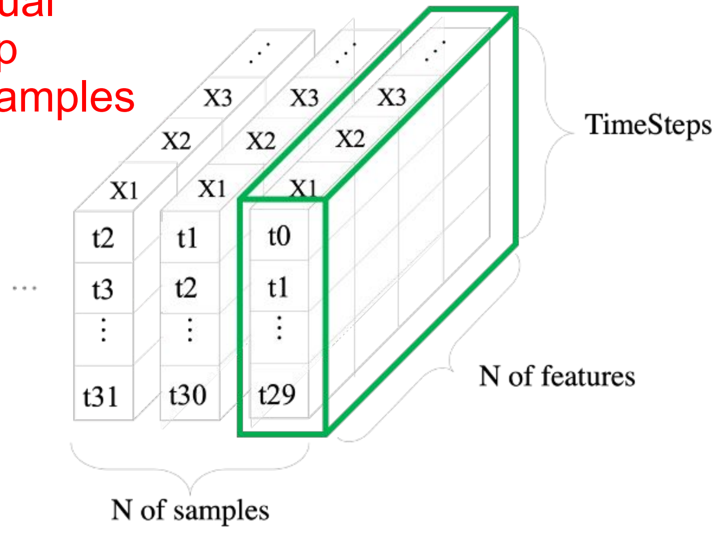


Create the input data by moving the window one step down each time

Stateless

Rolling window = 1

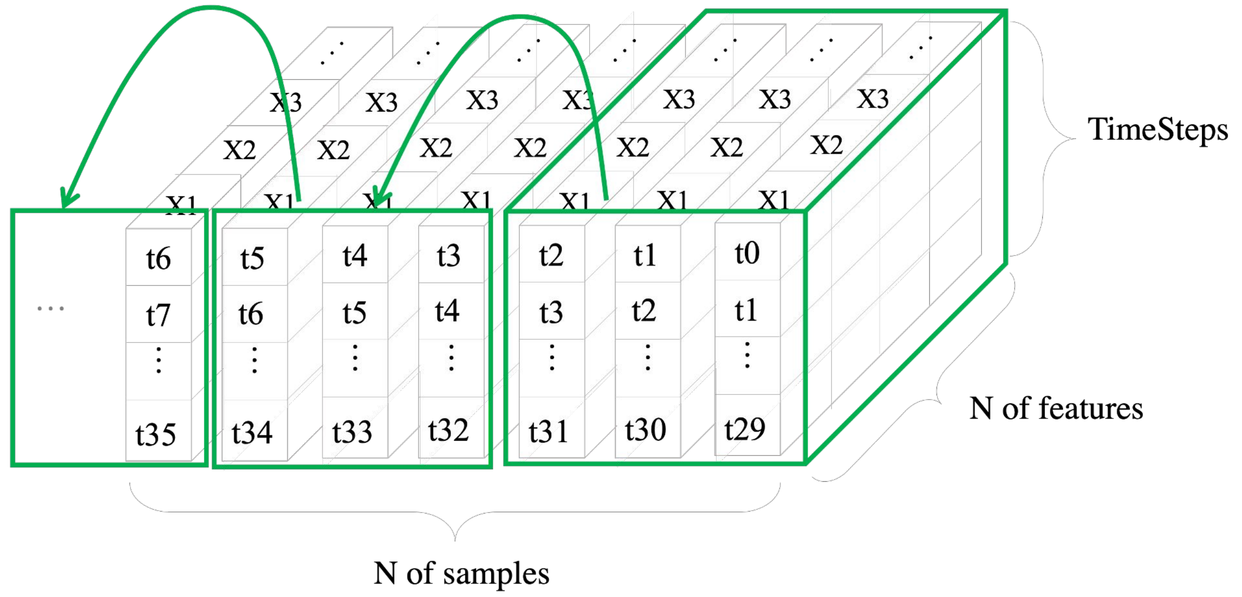
no contextual relationship between samples



Stateful

Rolling window = 1

batch_size = 3

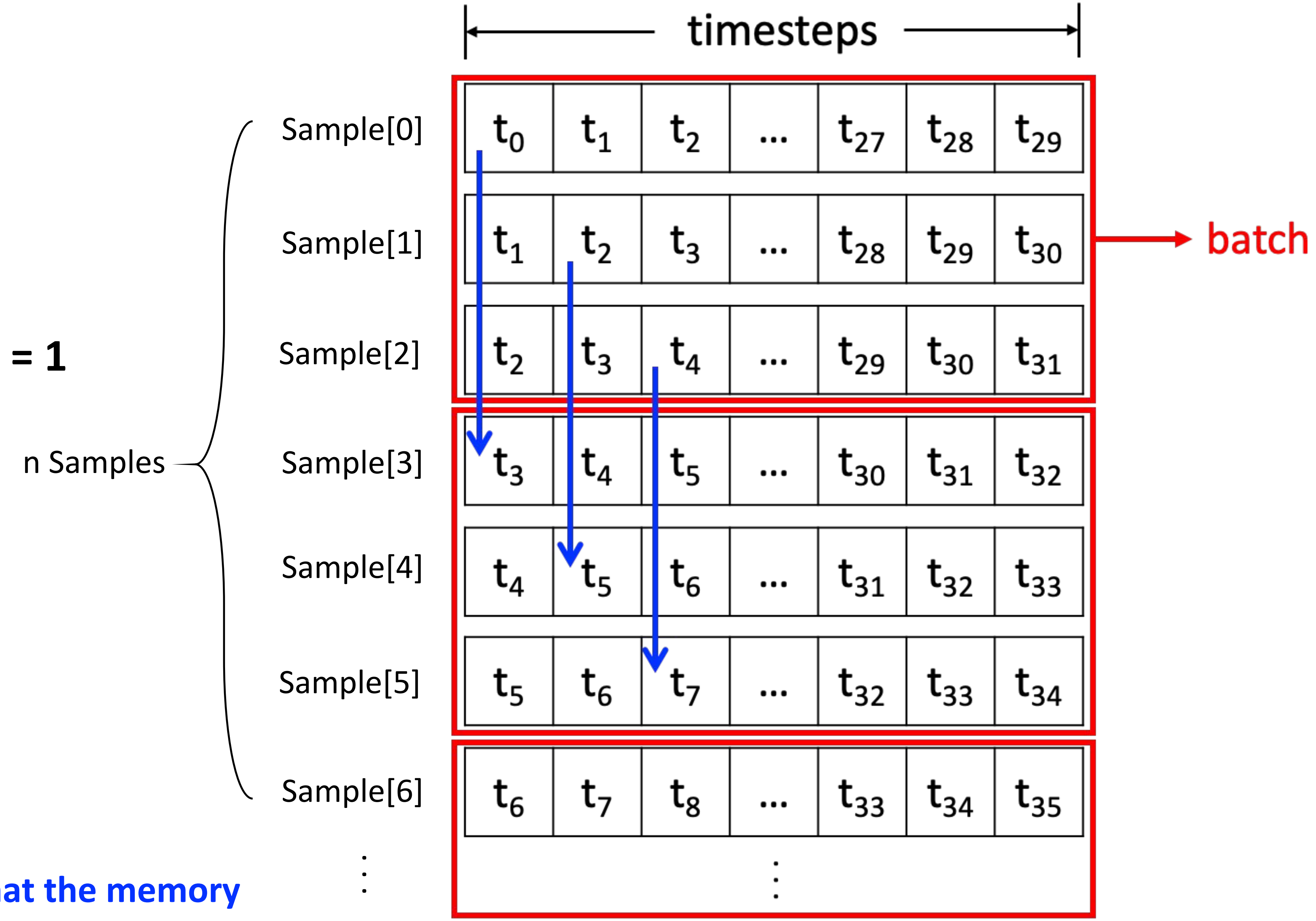


When
Stateful = True

Batch_size = 3

Timesteps = 30

Rolling window = 1



timesteps
batch
n Samples
Sample[0]
Sample[1]
Sample[2]
Sample[3]
Sample[4]
Sample[5]
Sample[6]
t_0 t_1 t_2 ... t_{27} t_{28} t_{29}
t_1 t_2 t_3 ... t_{28} t_{29} t_{30}
t_2 t_3 t_4 ... t_{29} t_{30} t_{31}
t_3 t_4 t_5 ... t_{30} t_{31} t_{32}
t_4 t_5 t_6 ... t_{31} t_{32} t_{33}
t_5 t_6 t_7 ... t_{32} t_{33} t_{34}
t_6 t_7 t_8 ... t_{33} t_{34} t_{35}

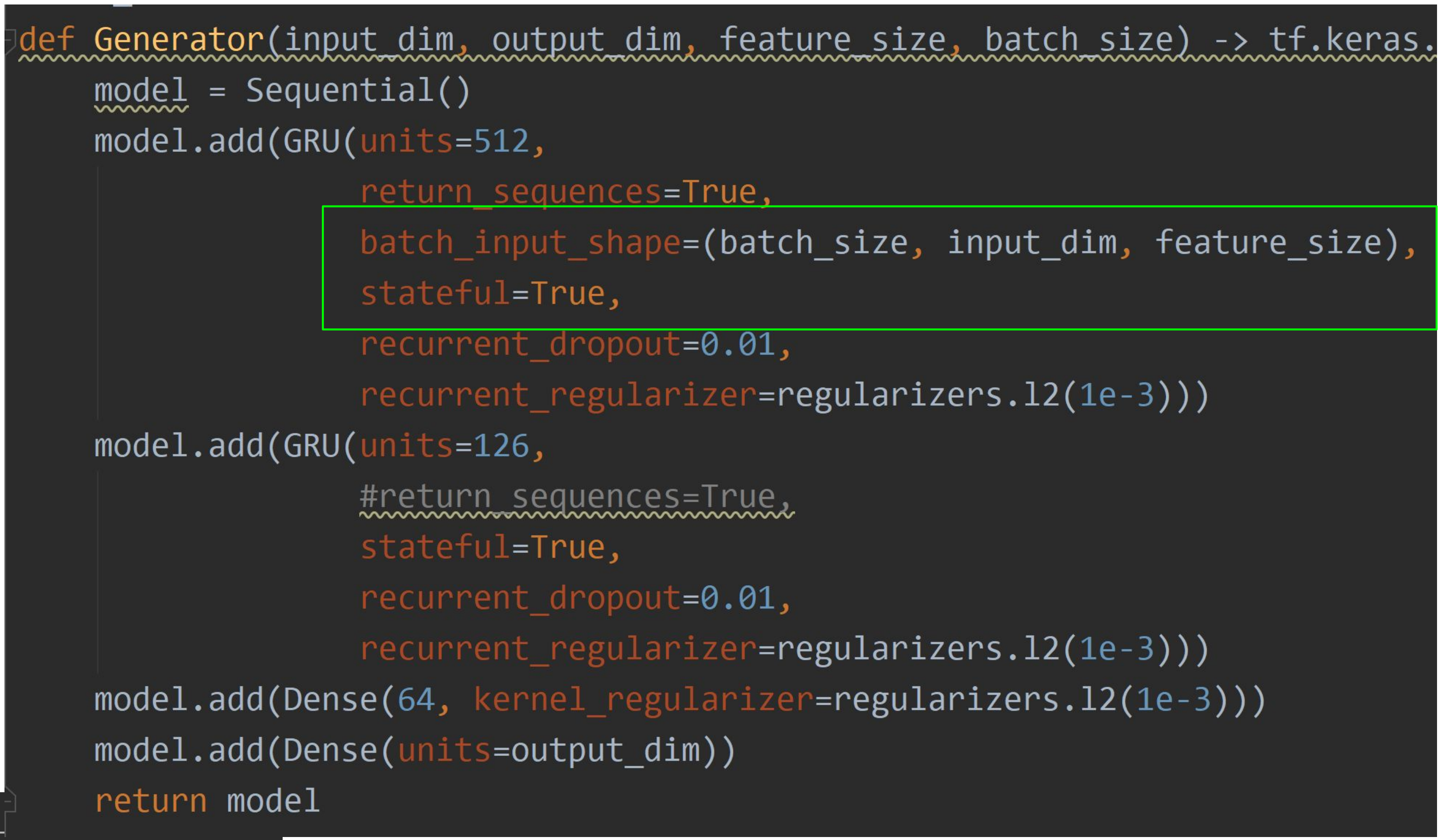
Blue arrow means that the memory state transfer between batches

Generator

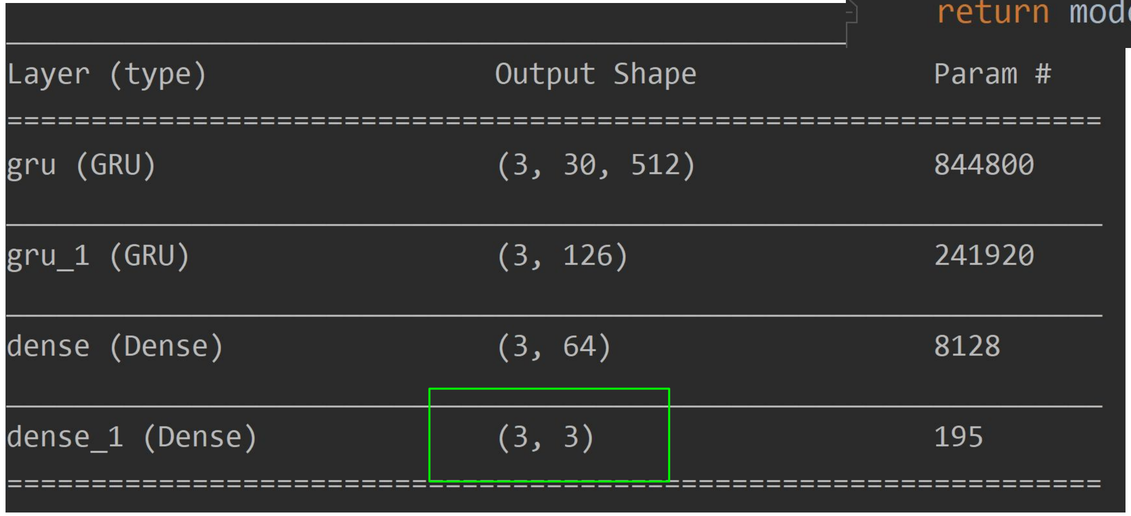
When
Stateful = True

Batch_size = 3

Timesteps = 30

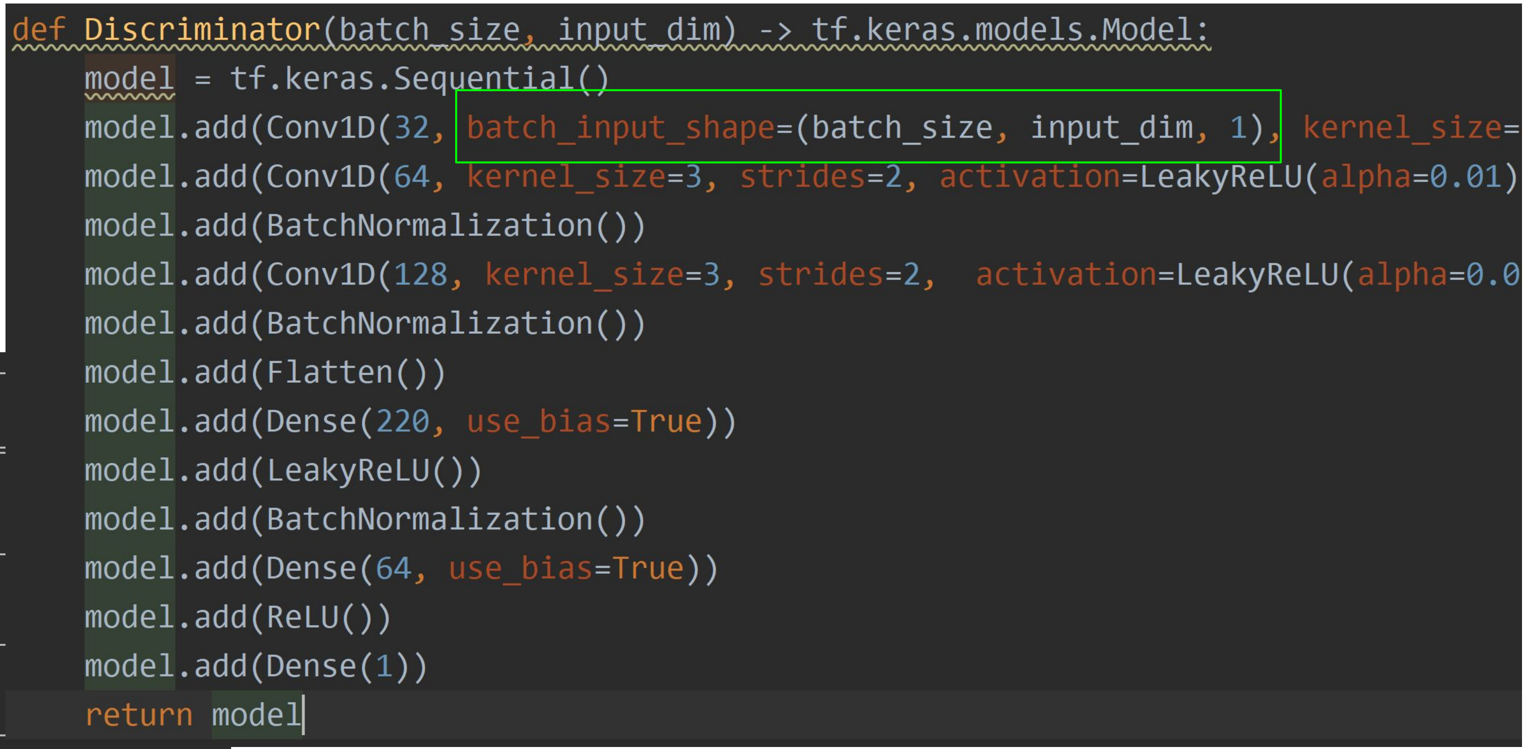


```
def Generator(input_dim, output_dim, feature_size, batch_size) -> tf.keras.
    model = Sequential()
    model.add(GRU(units=512,
                  return_sequences=True,
                  batch_input_shape=(batch_size, input_dim, feature_size),
                  stateful=True,
                  recurrent_dropout=0.01,
                  recurrent_regularizer=regularizers.l2(1e-3)))
    model.add(GRU(units=126,
                  #return_sequences=True,
                  stateful=True,
                  recurrent_dropout=0.01,
                  recurrent_regularizer=regularizers.l2(1e-3)))
    model.add(Dense(64, kernel_regularizer=regularizers.l2(1e-3)))
    model.add(Dense(units=output_dim))
    return model
```

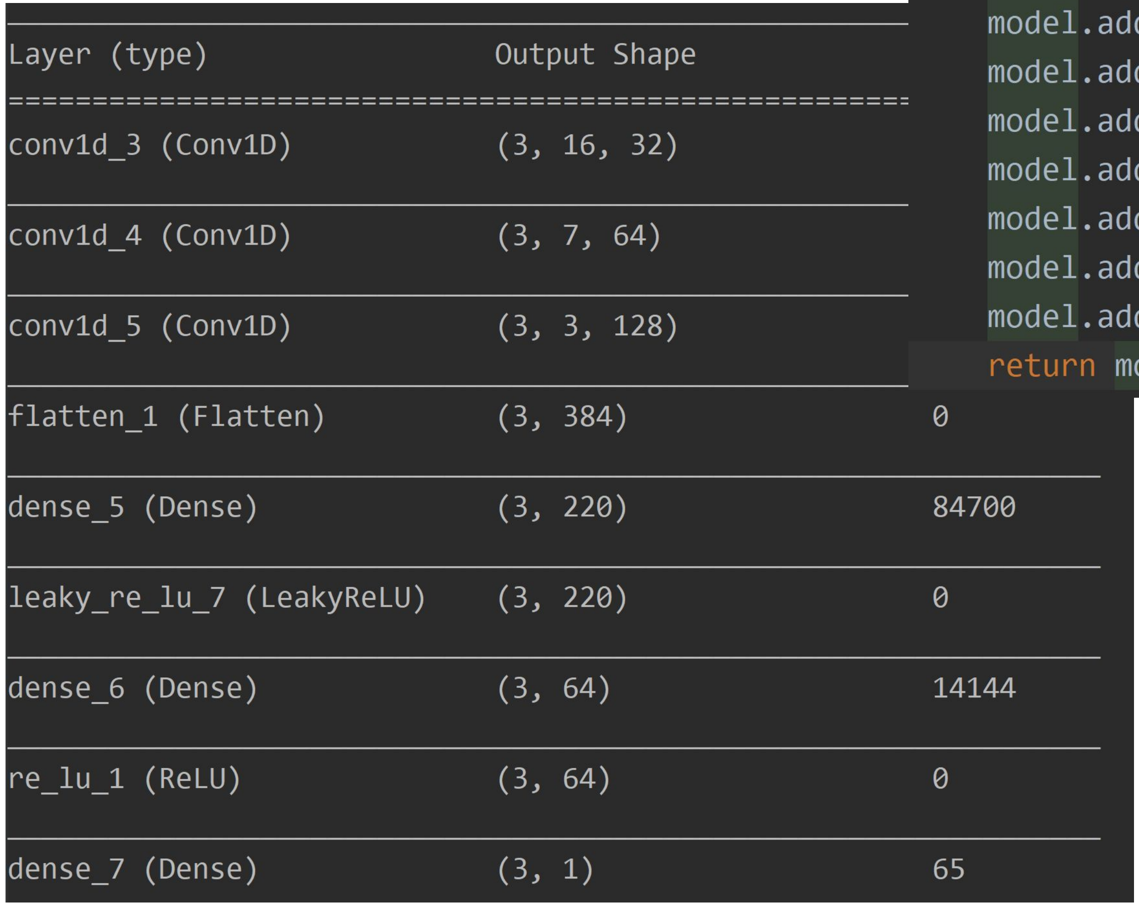


```
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (3, 30, 512)              844800

gru_1 (GRU)                  (3, 126)                  241920

dense (Dense)                (3, 64)                   8128

dense_1 (Dense)              (3, 3)                    195
=================================================================
```

Discriminator



```
def Discriminator(batch_size, input_dim) -> tf.keras.models.Model:
    model = tf.keras.Sequential()
    model.add(Conv1D(32, batch_input_shape=(batch_size, input_dim, 1), kernel_size=
    model.add(Conv1D(64, kernel_size=3, strides=2, activation=LeakyReLU(alpha=0.01)
    model.add(BatchNormalization())
    model.add(Conv1D(128, kernel_size=3, strides=2,  activation=LeakyReLU(alpha=0.0
    model.add(BatchNormalization())
    model.add(Flatten())
    model.add(Dense(220, use_bias=True))
    model.add(LeakyReLU())
    model.add(BatchNormalization())
    model.add(Dense(64, use_bias=True))
    model.add(ReLU())
    model.add(Dense(1))
    return model
```



```
Layer (type)                Output Shape
============================================================
conv1d_3 (Conv1D)           (3, 16, 32)

conv1d_4 (Conv1D)           (3, 7, 64)

conv1d_5 (Conv1D)           (3, 3, 128)

flatten_1 (Flatten)         (3, 384)                  0

dense_5 (Dense)             (3, 220)                  84700

leaky_re_lu_7 (LeakyReLU)   (3, 220)                  0

dense_6 (Dense)             (3, 64)                   14144

re_lu_1 (ReLU)              (3, 64)                   0

dense_7 (Dense)             (3, 1)                    65
```

Future work

- Working on the code with stateful model
- Pretrain the model for Generator and Discriminator

Thank you