# Model 1

class SOPTD(tf.keras.Model):  
 def \_\_init\_\_(self):  
 super(SOPTD, self).\_\_init\_\_()  
 self.dense1 = keras.layers.Dense(4, input\_shape=(1, 3), activation='linear', name='dense\_1')  
 self.dense2 = keras.layers.Dense(16, activation='tanh', name='dense\_2')  
 self.dense3 = keras.layers.Dense(16, activation='tanh', name='dense\_3')  
 self.dense4 = keras.layers.Dense(2, activation='linear', name='dense\_4')  
 self.dense5 = keras.layers.Dense(16, input\_shape=(1, 5), activation='linear', name='dense\_5')  
 self.dense6 = keras.layers.Dense(16, activation='tanh', name='dense\_6')  
 self.dense7 = keras.layers.Dense(16, activation='tanh', name='dense\_7')  
 self.dense8 = keras.layers.Dense(4, activation='linear', name='dense\_8')  
 self.Kp1 = tf.Variable(tf.cast(1.0, dtype=tf.float32))  
 self.Kp2 = tf.Variable(tf.cast(-1.0, dtype=tf.float32))  
 self.Tau1 = tf.Variable(tf.cast(1.0, dtype=tf.float32))  
 self.Tau2 = tf.Variable(tf.cast(1.0, dtype=tf.float32))  
 self.z1 = tf.Variable(tf.cast(0.5, dtype=tf.float32))  
 self.z2 = tf.Variable(tf.cast(0.5, dtype=tf.float32))  
  
 def call(self, inputs, \*\*kwargs):  
 t, states = inputs  
 u = states[0]  
 x1 = states[1]  
 x2 = states[2]  
 x1\_p = states[3]  
 x2\_p = states[4]  
 u1 = tf.reshape([u, x1, x2], [1, 3])  
 u2 = self.dense1(u1)  
 u3 = self.dense2(u2)  
 u4 = self.dense3(u3)  
 u5 = self.dense4(u4)  
 u6 = tf.reshape(u5, [2, ])  
 x1\_dot = x1\_p  
 x1\_p\_dot = ((self.Kp1/self.Tau1)\*u6[0] - 2.0\*(self.z1\*x1) - x1\_p/self.Tau1)/self.Tau1  
 x2\_dot = x2\_p  
 x2\_p\_dot = ((self.Kp2/self.Tau2)\*u6[1] - 2.0\*(self.z2\*x2) - x2\_p/self.Tau2)/self.Tau2  
 x\_dot = tf.concat([[x1\_dot], [x2\_dot], [x1\_p\_dot], [x2\_p\_dot], [u]], axis=0)  
 g0 = tf.reshape(x\_dot, [1, 5])  
 g1 = self.dense5(g0)  
 g2 = self.dense6(g1)  
 g3 = self.dense7(g2)  
 g4 = self.dense8(g3)  
 g5 = tf.reshape(g4, [4, ])  
  
 u\_dot = tf.reshape(tf.cast(0, dtype=tf.float32), [1, ])  
 h = tf.concat([u\_dot, g5], axis=0)  
 return h

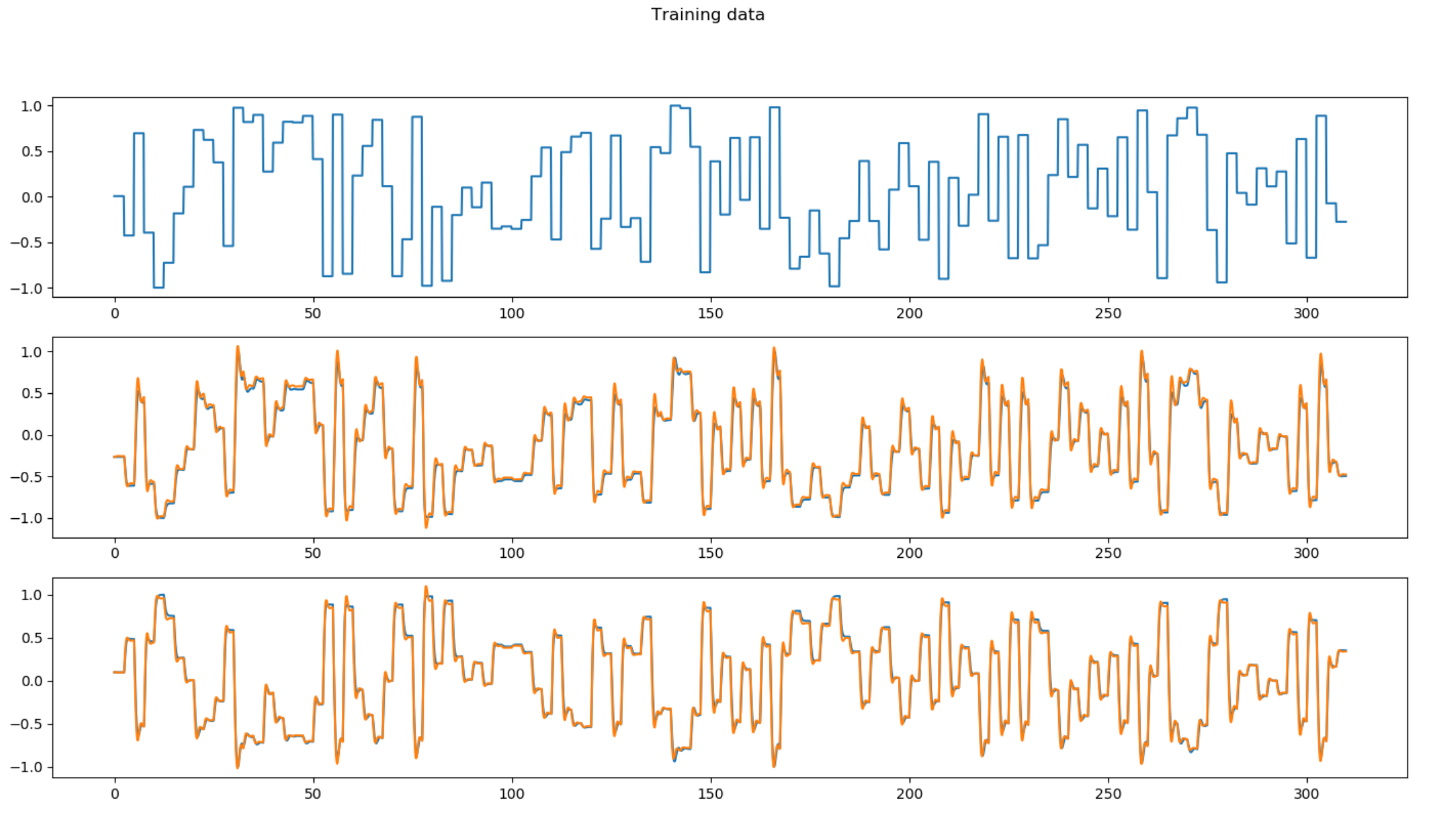
Batch size: 50

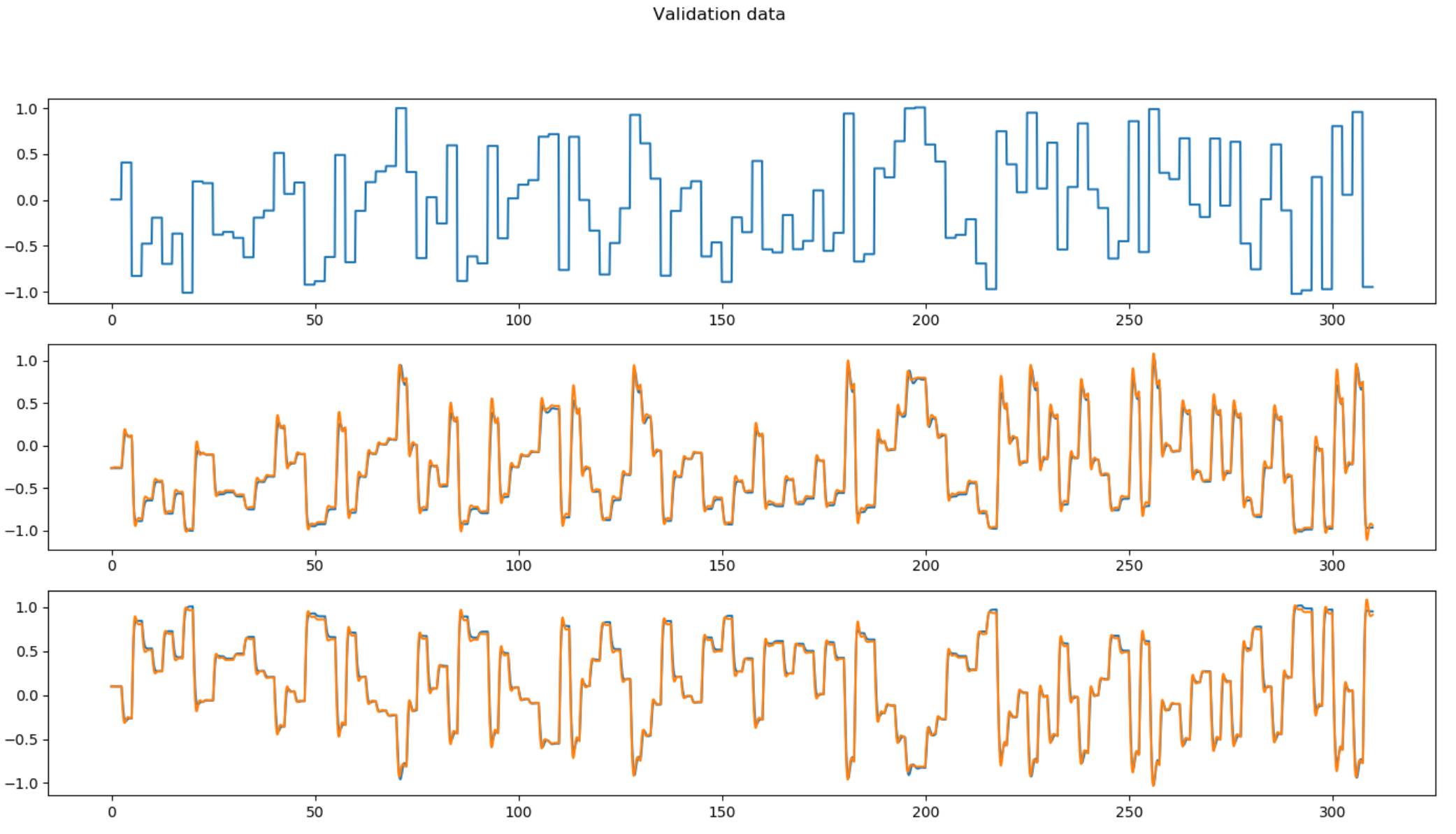
Learning rate: 0.01

Training time: 3728.4154884815216

Training loss: 17.336041173931605

validation\_loss: 15.50629774173765





# Model 2

class SOPTD(tf.keras.Model):  
 def \_\_init\_\_(self):  
 super(SOPTD, self).\_\_init\_\_()  
 self.dense1 = keras.layers.Dense(4, input\_shape=(1, 3), activation='linear', name='dense\_1')  
 self.dense2 = keras.layers.Dense(32, activation='tanh', name='dense\_2')  
 self.dense3 = keras.layers.Dense(32, activation='tanh', name='dense\_3')  
 self.dense4 = keras.layers.Dense(2, activation='linear', name='dense\_4')  
 self.dense5 = keras.layers.Dense(32, input\_shape=(1, 5), activation='linear', name='dense\_5')  
 self.dense6 = keras.layers.Dense(32, activation='tanh', name='dense\_6')  
 self.dense7 = keras.layers.Dense(32, activation='tanh', name='dense\_7')  
 self.dense8 = keras.layers.Dense(4, activation='linear', name='dense\_8')  
 self.Kp1 = tf.Variable(tf.cast(1.0, dtype=tf.float32))  
 self.Kp2 = tf.Variable(tf.cast(-1.0, dtype=tf.float32))  
 self.Tau1 = tf.Variable(tf.cast(1.0, dtype=tf.float32))  
 self.Tau2 = tf.Variable(tf.cast(1.0, dtype=tf.float32))  
 self.z1 = tf.Variable(tf.cast(0.5, dtype=tf.float32))  
 self.z2 = tf.Variable(tf.cast(0.5, dtype=tf.float32))  
  
 def call(self, inputs, \*\*kwargs):  
 t, states = inputs  
 u = states[0]  
 x1 = states[1]  
 x2 = states[2]  
 x1\_p = states[3]  
 x2\_p = states[4]  
 u1 = tf.reshape([u, x1, x2], [1, 3])  
 u2 = self.dense1(u1)  
 u3 = self.dense2(u2)  
 u4 = self.dense3(u3)  
 u5 = self.dense4(u4)  
 u6 = tf.reshape(u5, [2, ])  
 x1\_dot = x1\_p  
 x1\_p\_dot = ((self.Kp1/self.Tau1)\*u6[0] - 2.0\*(self.z1\*x1) - x1\_p/self.Tau1)/self.Tau1  
 x2\_dot = x2\_p  
 x2\_p\_dot = ((self.Kp2/self.Tau2)\*u6[1] - 2.0\*(self.z2\*x2) - x2\_p/self.Tau2)/self.Tau2  
 x\_dot = tf.concat([[x1\_dot], [x2\_dot], [x1\_p\_dot], [x2\_p\_dot], [u]], axis=0)  
 g0 = tf.reshape(x\_dot, [1, 5])  
 g1 = self.dense5(g0)  
 g2 = self.dense6(g1)  
 g3 = self.dense7(g2)  
 g4 = self.dense8(g3)  
 g5 = tf.reshape(g4, [4, ])  
  
 u\_dot = tf.reshape(tf.cast(0, dtype=tf.float32), [1, ])  
 h = tf.concat([u\_dot, g5], axis=0)  
 return h

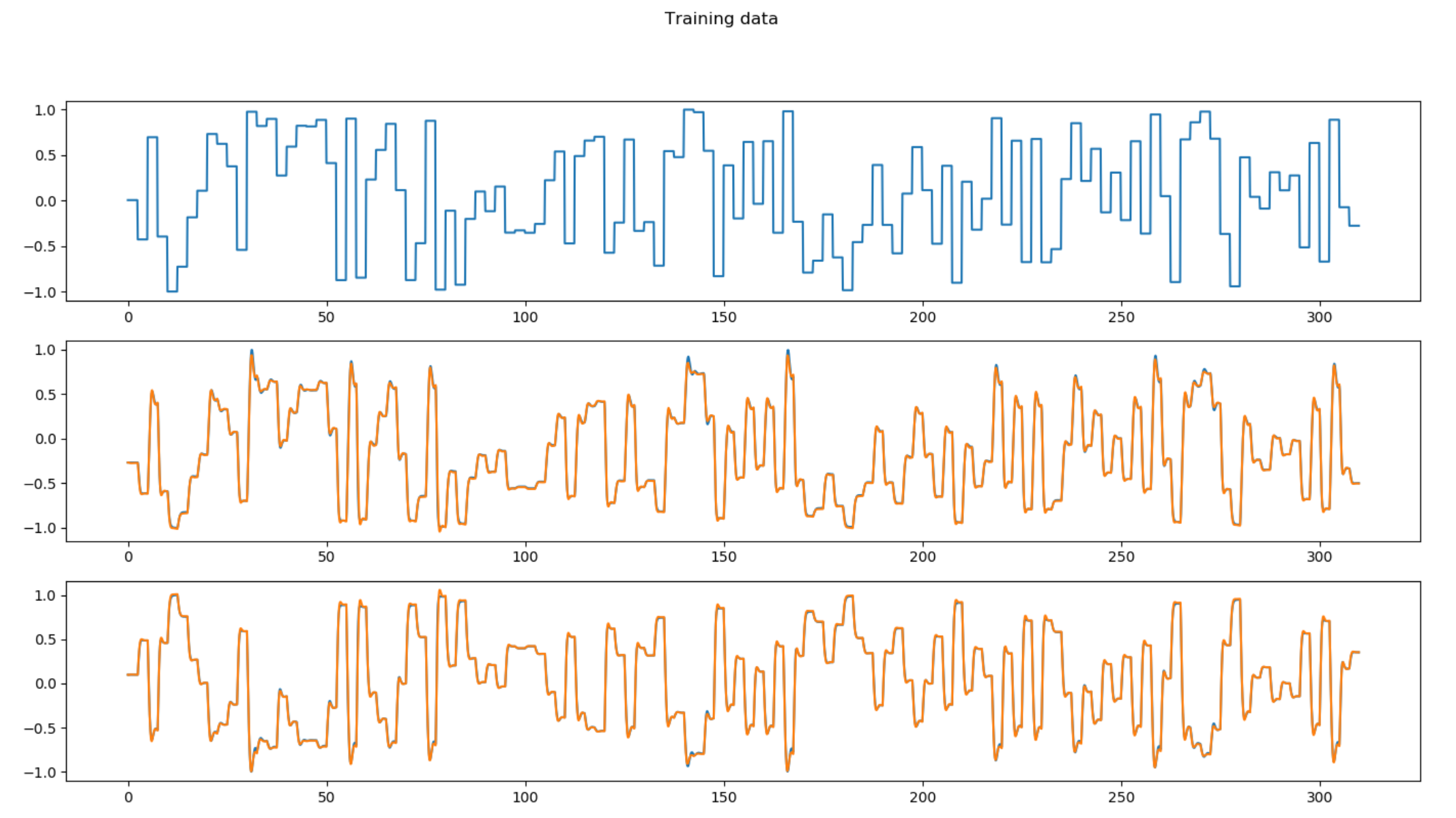
Batch size: 50

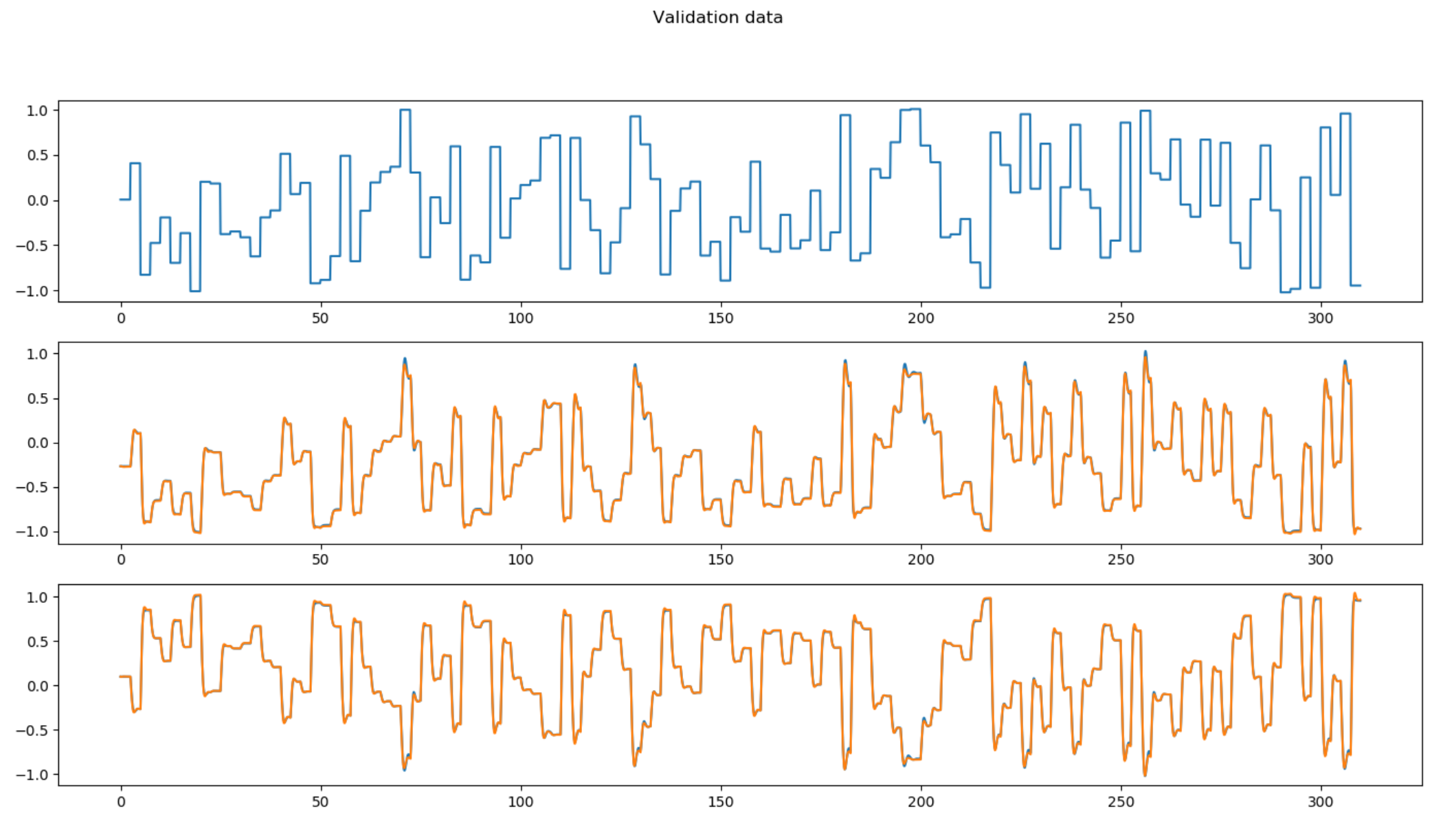
Learning rate: 0.01

Training time: 3791.777435541153

Training loss: 2.4237738983609365

validation\_loss: 2.415115003802585





# Model 3

class SOPTD(tf.keras.Model):  
 def \_\_init\_\_(self):  
 super(SOPTD, self).\_\_init\_\_()  
 self.dense1 = keras.layers.Dense(4, input\_shape=(1, 3), activation='linear', name='dense\_1')  
 self.dense2 = keras.layers.Dense(32, activation='tanh', name='dense\_2')  
 self.dense3 = keras.layers.Dense(32, activation='tanh', name='dense\_3')  
 self.dense4 = keras.layers.Dense(32, activation='tanh', name='dense\_4')  
 self.dense5 = keras.layers.Dense(2, activation='linear', name='dense\_5')  
 self.dense6 = keras.layers.Dense(32, input\_shape=(1, 5), activation='linear', name='dense\_6')  
 self.dense7 = keras.layers.Dense(32, activation='tanh', name='dense\_7')  
 self.dense8 = keras.layers.Dense(32, activation='tanh', name='dense\_8')  
 self.dense9 = keras.layers.Dense(32, activation='tanh', name='dense\_9')  
 self.dense10 = keras.layers.Dense(4, activation='linear', name='dense\_10')  
 self.Kp1 = tf.Variable(tf.cast(1.0, dtype=tf.float32))  
 self.Kp2 = tf.Variable(tf.cast(-1.0, dtype=tf.float32))  
 self.Tau1 = tf.Variable(tf.cast(1.0, dtype=tf.float32))  
 self.Tau2 = tf.Variable(tf.cast(1.0, dtype=tf.float32))  
 self.z1 = tf.Variable(tf.cast(0.5, dtype=tf.float32))  
 self.z2 = tf.Variable(tf.cast(0.5, dtype=tf.float32))  
  
 def call(self, inputs, \*\*kwargs):  
 t, states = inputs  
 u = states[0]  
 x1 = states[1]  
 x2 = states[2]  
 x1\_p = states[3]  
 x2\_p = states[4]  
 u1 = tf.reshape([u, x1, x2], [1, 3])  
 u2 = self.dense1(u1)  
 u3 = self.dense2(u2)  
 u4 = self.dense3(u3)  
 u5 = self.dense4(u4)  
 u6 = tf.reshape(self.dense5(u5), [2, ])  
 x1\_dot = x1\_p  
 x1\_p\_dot = ((self.Kp1/self.Tau1)\*u6[0] - 2.0\*(self.z1\*x1) - x1\_p/self.Tau1)/self.Tau1  
 x2\_dot = x2\_p  
 x2\_p\_dot = ((self.Kp2/self.Tau2)\*u6[1] - 2.0\*(self.z2\*x2) - x2\_p/self.Tau2)/self.Tau2  
 x\_dot = tf.concat([[x1\_dot], [x2\_dot], [x1\_p\_dot], [x2\_p\_dot], [u]], axis=0)  
 g0 = tf.reshape(x\_dot, [1, 5])  
 g1 = self.dense6(g0)  
 g2 = self.dense7(g1)  
 g3 = self.dense8(g2)  
 g4 = self.dense9(g3)  
 g5 = tf.reshape(self.dense10(g4), [4, ])  
  
 u\_dot = tf.reshape(tf.cast(0, dtype=tf.float32), [1, ])  
 h = tf.concat([u\_dot, g5], axis=0)  
 return h

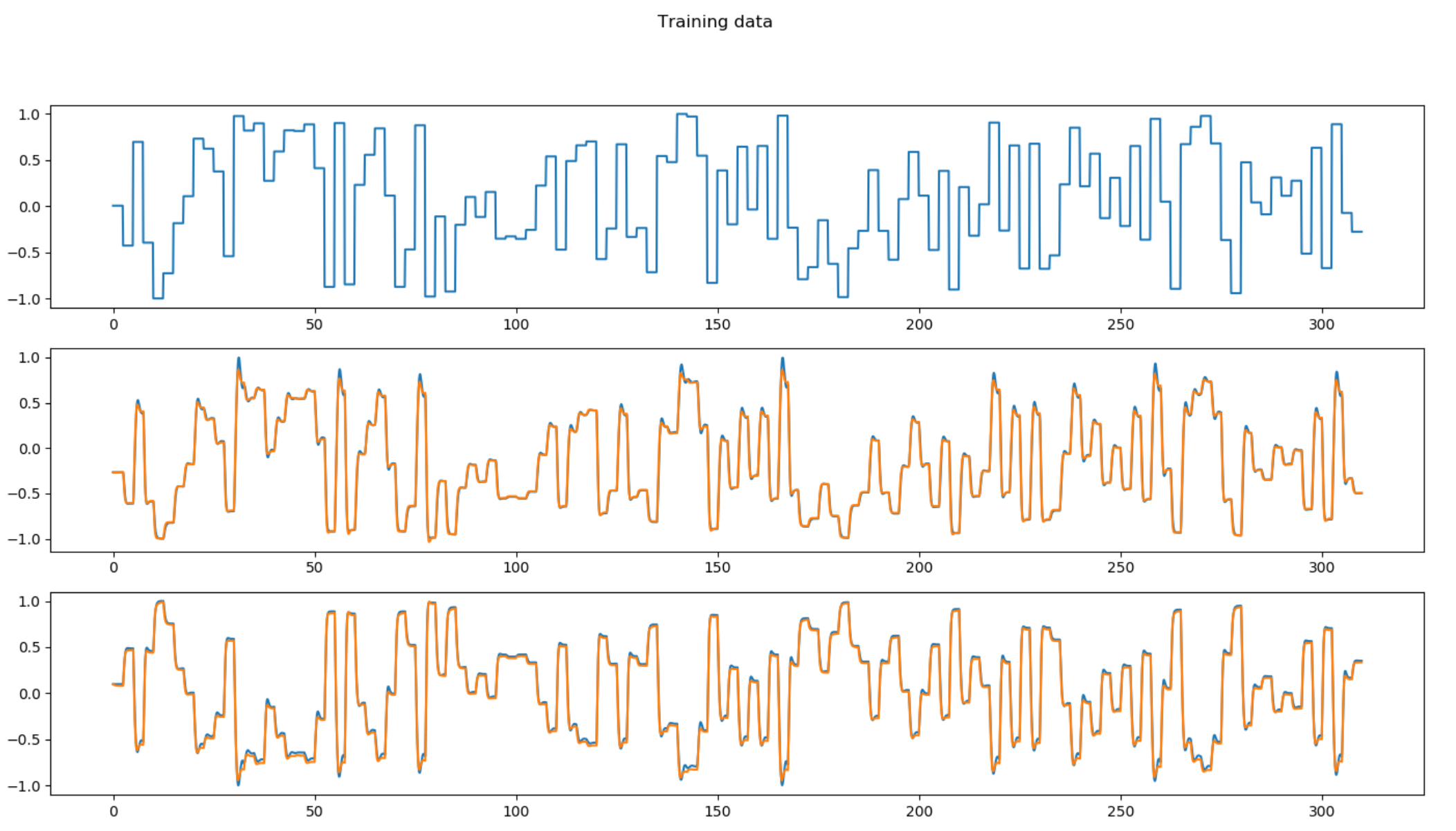
Batch size: 50

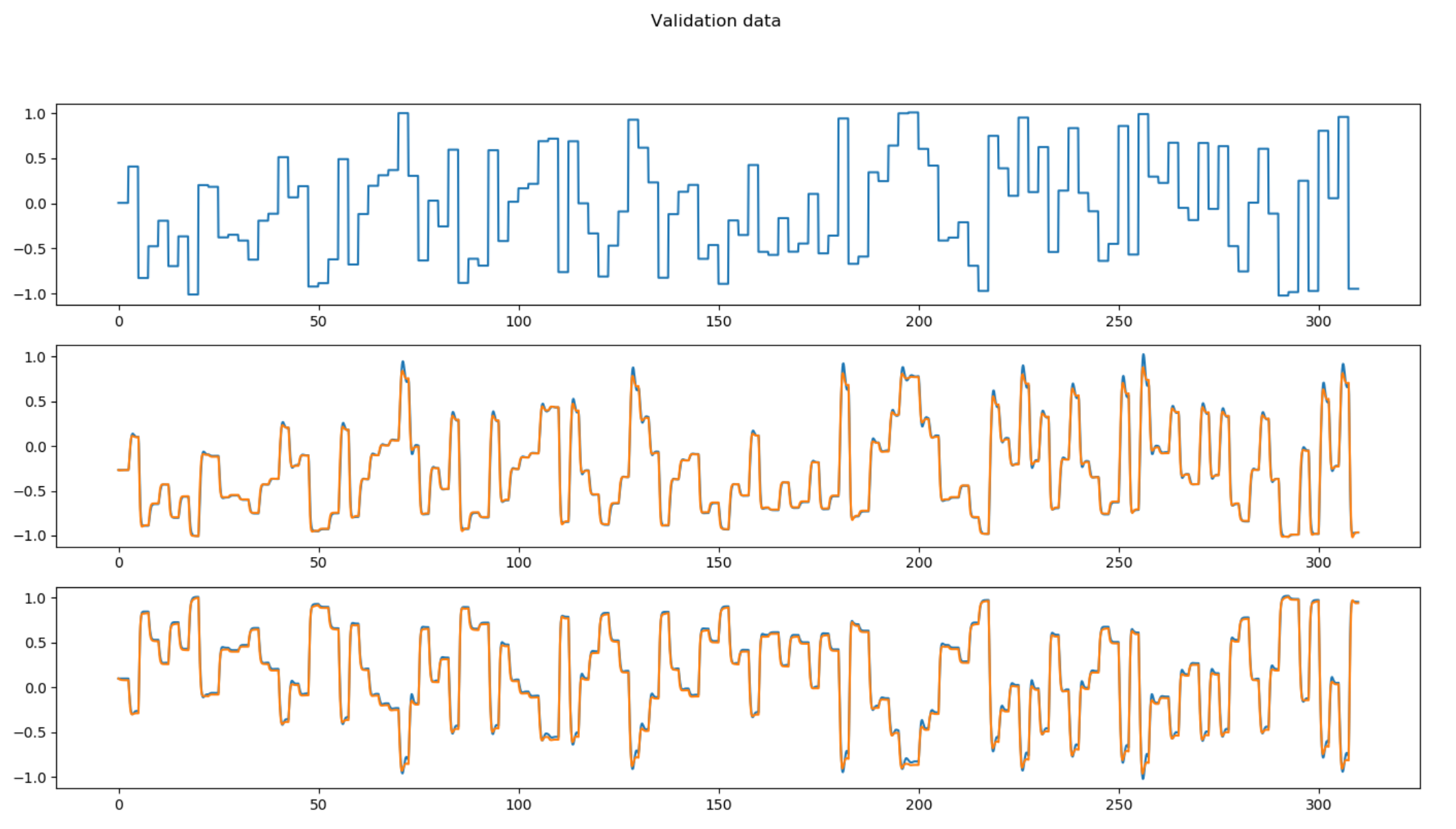
Learning rate: 0.01

Training time: 4334.174948453903

Training loss: 7.269271510500609

validation\_loss: 6.3183347562049





# Model 4

class SOPTD(tf.keras.Model):  
 def \_\_init\_\_(self):  
 super(SOPTD, self).\_\_init\_\_()  
 self.dense1 = keras.layers.Dense(4, input\_shape=(1, 3), activation='linear', name='dense\_1')  
 self.dense2 = keras.layers.Dense(32, activation='tanh', name='dense\_2')  
 self.dense3 = keras.layers.Dense(32, activation='tanh', name='dense\_3')  
 self.dense4 = keras.layers.Dense(2, activation='linear', name='dense\_4')  
 self.Kp1 = tf.Variable(tf.cast(1.0, dtype=tf.float32))  
 self.Kp2 = tf.Variable(tf.cast(-1.0, dtype=tf.float32))  
 self.Tau1 = tf.Variable(tf.cast(1.0, dtype=tf.float32))  
 self.Tau2 = tf.Variable(tf.cast(1.0, dtype=tf.float32))  
 self.z1 = tf.Variable(tf.cast(0.5, dtype=tf.float32))  
 self.z2 = tf.Variable(tf.cast(0.5, dtype=tf.float32))  
  
 def call(self, inputs, \*\*kwargs):  
 t, states = inputs  
 u = states[0]  
 x1 = states[1]  
 x2 = states[2]  
 x1\_p = states[3]  
 x2\_p = states[4]  
 u1 = tf.reshape([u, x1, x2], [1, 3])  
 u2 = self.dense1(u1)  
 u3 = self.dense2(u2)  
 u4 = self.dense3(u3)  
 u5 = self.dense4(u4)  
 u6 = tf.reshape(u5, [2, ])  
 x1\_dot = x1\_p  
 x1\_p\_dot = ((self.Kp1/self.Tau1)\*u6[0] - 2.0\*(self.z1\*x1) - x1\_p/self.Tau1)/self.Tau1  
 x2\_dot = x2\_p  
 x2\_p\_dot = ((self.Kp2/self.Tau2)\*u6[1] - 2.0\*(self.z2\*x2) - x2\_p/self.Tau2)/self.Tau2  
 x\_dot = tf.concat([[x1\_dot], [x2\_dot], [x1\_p\_dot], [x2\_p\_dot]], axis=0)  
 g5 = tf.reshape(x\_dot, [4, ])  
  
 u\_dot = tf.reshape(tf.cast(0, dtype=tf.float32), [1, ])  
 h = tf.concat([u\_dot, g5], axis=0)  
 return h

Batch size: 50

Learning rate: 0.01

Training time: 3190.381765127182

Training loss: 7.225109343192752

validation\_loss: 6.759640057697522

