Homework 5 (100 points)

This homework will focus on Neural Networks and visualization.

a) Write a function that takes a keras network and outputs an image (png format) of the network. (10points)

You can assume the model is sequential and only uses dense layers. The input and output neurons must be blue circles. The hidden neurons must be green circles. The edges must be directed red arrows.

For example, the output image for

```
    model = keras.models.Sequential()
    model.add(layers.Dense(2, input_dim=2))
    model.add(layers.Dense(1))
    model.compile(loss="binary_crossentropy")
```

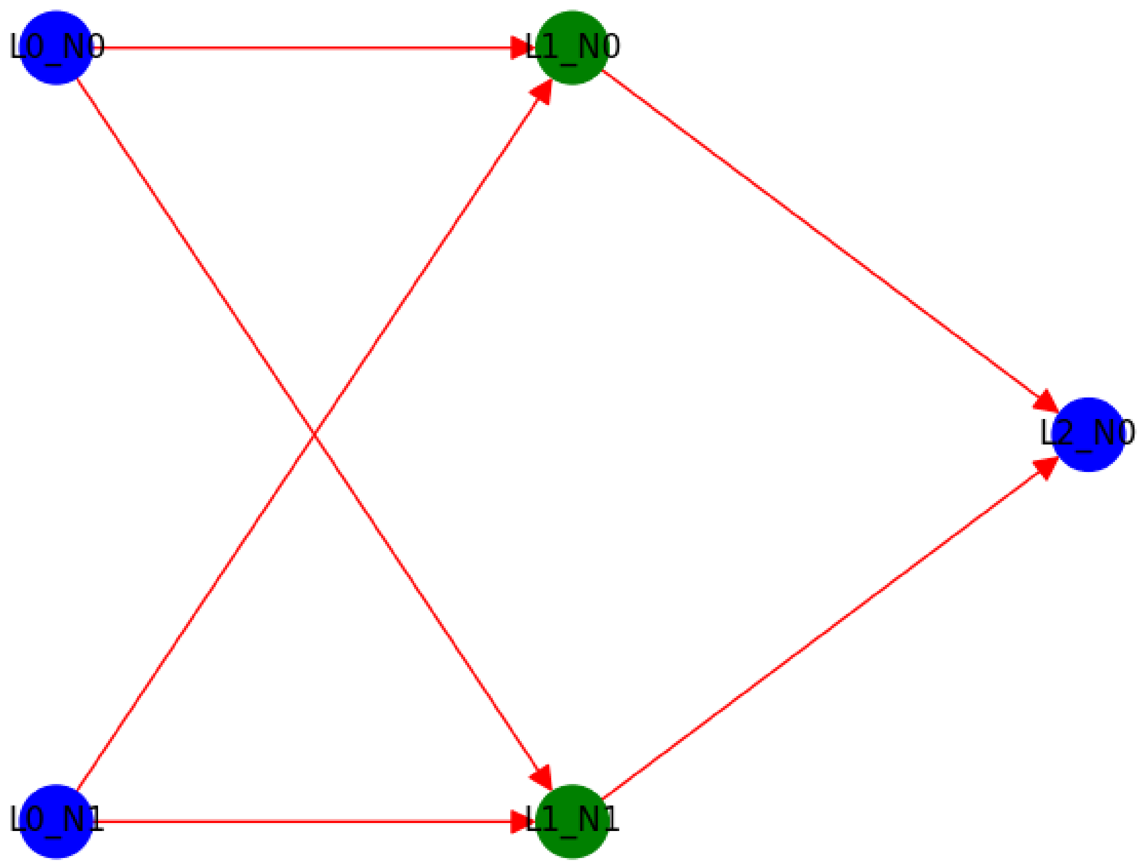
should look exactly like this:

```
from IPython.display import Image
#Image(filename="example.png")
```

Hint: use the networkx library (specifically the to_agraph method)

```
#libraries used in this exercise
import numpy as np
import networkx as nx
from PIL import Image
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation, PillowWriter

from tensorflow.keras import layers, models, activations
from tensorflow.keras.utils import plot_model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from networkx.drawing.nx_agraph import to_agraph

from IPython.display import Image as img_display

def plot_keras_model(model, filename):
    # initialize graph
    G = nx.DiGraph()

    # model layers info
    layers_info = []
    for i, layer in enumerate(model.layers):
```

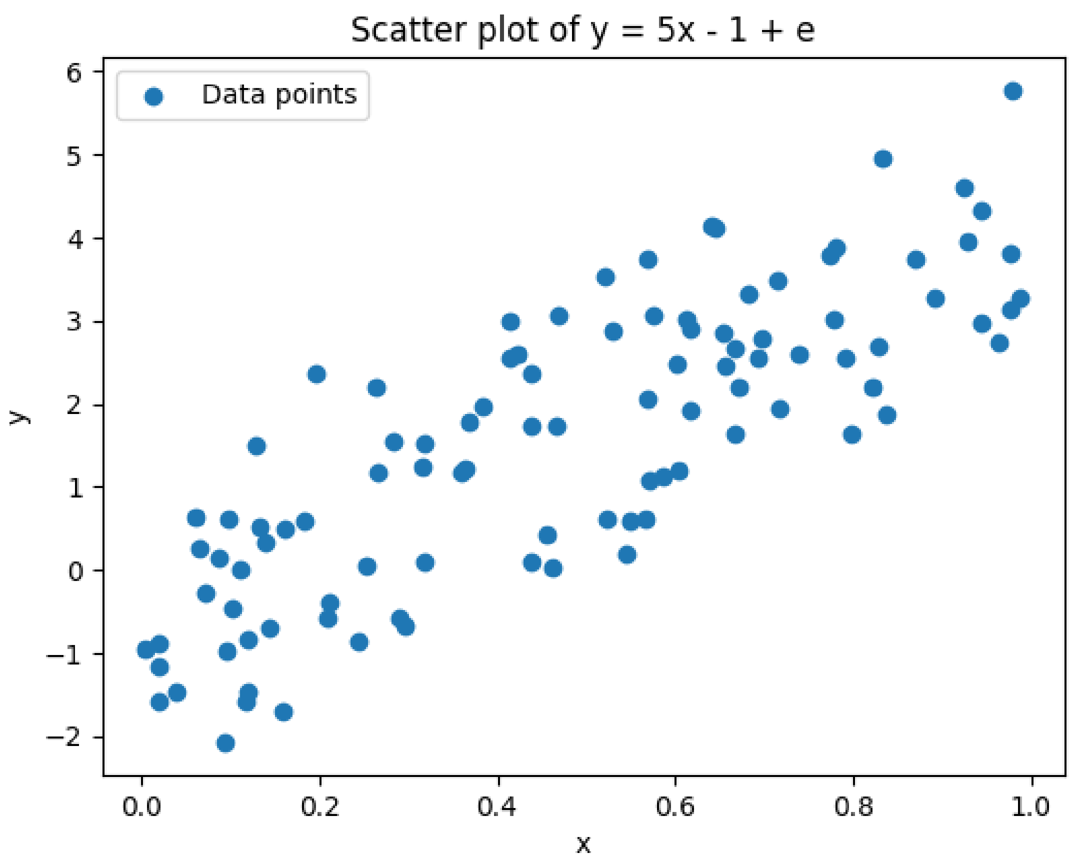
```
        if isinstance(layer, layers.Dense):
            if i == 0:
                # input nodes
                input_shape = layer.input_shape
                if input_shape and len(input_shape) > 1:
                    layers_info.append((input_shape[-1], 'input'))
            # the output layer if final, else hidden
            layers_info.append((layer.units, 'output' if i ==
len(model.layers)-1 else 'hidden'))

    # format nodes with type
    for i, (layer_size, layer_type) in enumerate(layers_info):
        nodes = [('L{}_N{}'.format(i, j), {'layer': i, 'type':
layer_type}) for j in range(layer_size)]
        G.add_nodes_from(nodes)

    # add edges to graph
    for i in range(len(layers_info) - 1):
        for j in range(layers_info[i][0]):
            for k in range(layers_info[i+1][0]):
                G.add_edge(f'L{i}_N{j}', f'L{i+1}_N{k}')

    # color coding
    color_map = ['blue' if node_data['type'] in ['input', 'output']
else 'green'
                 for _, node_data in G.nodes(data=True)]

    # make graph
    pos = nx.multipartite_layout(G, subset_key="layer")
    nx.draw(G, pos, with_labels=True, node_color=color_map,
node_size=700, arrows=True,
            arrowstyle='-|>', arrowsize=20, edge_color='red')

    # Save the figure
    plt.savefig(filename, format='PNG')
    plt.close()  # Close the figure to prevent display in the notebook


    return filename

model_a = models.Sequential()
model_a.add(layers.Dense(2, input_dim=2))
model_a.add(layers.Dense(1))
model_a.compile(loss="binary_crossentropy")

plot_keras_model(model_a, "model_a.png")
Image.open("model_a.png")
```



b) Generate 100 datapoints of the form y = 5x - 1 + e where e ~ N(0, 1) and plot the data in a scatter plot. Create a Neural Network with no hidden layers (just input to ouput each with just one neuron), using the `mean_squared_error` loss and no activation function. Create an image of this model using a) then train this model on the dataset produced such that it learns a good fit to the points. Plot that fitted line. (10points)

```
# y = 5x - 1 + e where e ~ N(0, 1)
np.random.seed(0)
x = np.random.rand(100, 1)
e = np.random.randn(100, 1)
y = 5 * x - 1 + e

# scatter plot
plt.scatter(x, y, label='Data points')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatter plot of y = 5x - 1 + e')
plt.legend()
plt.show()


# Neural Network
```

```python
model = Sequential([
    Dense(units=1, input_shape=(1,), use_bias=True)
])

# mean_squared_error loss and no activation function
model.compile(optimizer='sgd', loss='mean_squared_error')

# train
model.fit(x, y, epochs=200)

# predict
y_pred = model.predict(x)

# plot fitted line
plt.scatter(x, y, label='Data points')
plt.plot(x, y_pred, color='red', label='Fitted line')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatter plot with the fitted line')
plt.legend()
plt.show()

#image of model using (a)
plot_keras_model(model, "model_b.png")
Image.open("model_b.png")
```




```
Epoch 1/200
4/4 [==============================] - 0s 5ms/step - loss: 5.8955
Epoch 2/200
4/4 [==============================] - 0s 4ms/step - loss: 5.3510
Epoch 3/200
4/4 [==============================] - 0s 4ms/step - loss: 4.8597
Epoch 4/200
4/4 [==============================] - 0s 4ms/step - loss: 4.4974
Epoch 5/200
4/4 [==============================] - 0s 3ms/step - loss: 4.1404
Epoch 6/200
4/4 [==============================] - 0s 4ms/step - loss: 3.8473
Epoch 7/200
4/4 [==============================] - 0s 6ms/step - loss: 3.6069
Epoch 8/200
4/4 [==============================] - 0s 4ms/step - loss: 3.4333
Epoch 9/200
4/4 [==============================] - 0s 4ms/step - loss: 3.2595
Epoch 10/200
4/4 [==============================] - 0s 4ms/step - loss: 3.1014
Epoch 11/200
4/4 [==============================] - 0s 4ms/step - loss: 2.9794
Epoch 12/200
```

```
4/4 [==============================] - 0s 4ms/step - loss: 2.8603
Epoch 13/200
4/4 [==============================] - 0s 4ms/step - loss: 2.7270
Epoch 14/200
4/4 [==============================] - 0s 6ms/step - loss: 2.6843
Epoch 15/200
4/4 [==============================] - 0s 6ms/step - loss: 2.6335
Epoch 16/200
4/4 [==============================] - 0s 5ms/step - loss: 2.5888
Epoch 17/200
4/4 [==============================] - 0s 4ms/step - loss: 2.5370
Epoch 18/200
4/4 [==============================] - 0s 4ms/step - loss: 2.4974
Epoch 19/200
4/4 [==============================] - 0s 4ms/step - loss: 2.4476
Epoch 20/200
4/4 [==============================] - 0s 4ms/step - loss: 2.4148
Epoch 21/200
4/4 [==============================] - 0s 4ms/step - loss: 2.3884
Epoch 22/200
4/4 [==============================] - 0s 4ms/step - loss: 2.3760
Epoch 23/200
4/4 [==============================] - 0s 4ms/step - loss: 2.3556
Epoch 24/200
4/4 [==============================] - 0s 4ms/step - loss: 2.3322
Epoch 25/200
4/4 [==============================] - 0s 4ms/step - loss: 2.3141
Epoch 26/200
4/4 [==============================] - 0s 4ms/step - loss: 2.2914
Epoch 27/200
4/4 [==============================] - 0s 4ms/step - loss: 2.2737
Epoch 28/200
4/4 [==============================] - 0s 4ms/step - loss: 2.2621
Epoch 29/200
4/4 [==============================] - 0s 4ms/step - loss: 2.2360
Epoch 30/200
4/4 [==============================] - 0s 4ms/step - loss: 2.2221
Epoch 31/200
4/4 [==============================] - 0s 3ms/step - loss: 2.2093
Epoch 32/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1982
Epoch 33/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1859
Epoch 34/200
4/4 [==============================] - 0s 3ms/step - loss: 2.1750
Epoch 35/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1608
Epoch 36/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1462
```

```
Epoch 37/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1320
Epoch 38/200
4/4 [==============================] - 0s 3ms/step - loss: 2.1187
Epoch 39/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1094
Epoch 40/200
4/4 [==============================] - 0s 5ms/step - loss: 2.1002
Epoch 41/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0862
Epoch 42/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0761
Epoch 43/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0639
Epoch 44/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0514
Epoch 45/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0358
Epoch 46/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0260
Epoch 47/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0174
Epoch 48/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0062
Epoch 49/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9961
Epoch 50/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9878
Epoch 51/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9783
Epoch 52/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9665
Epoch 53/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9563
Epoch 54/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9476
Epoch 55/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9364
Epoch 56/200
4/4 [==============================] - 0s 5ms/step - loss: 1.9282
Epoch 57/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9160
Epoch 58/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9053
Epoch 59/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8935
Epoch 60/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8863
Epoch 61/200
```

```
4/4 [==============================] - 0s 4ms/step - loss: 1.8736
Epoch 62/200
4/4 [==============================] - 0s 5ms/step - loss: 1.8645
Epoch 63/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8527
Epoch 64/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8433
Epoch 65/200
4/4 [==============================] - 0s 5ms/step - loss: 1.8347
Epoch 66/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8243
Epoch 67/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8164
Epoch 68/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8077
Epoch 69/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8035
Epoch 70/200
4/4 [==============================] - 0s 5ms/step - loss: 1.7916
Epoch 71/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7832
Epoch 72/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7761
Epoch 73/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7701
Epoch 74/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7628
Epoch 75/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7551
Epoch 76/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7472
Epoch 77/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7384
Epoch 78/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7287
Epoch 79/200
4/4 [==============================] - 0s 5ms/step - loss: 1.7185
Epoch 80/200
4/4 [==============================] - 0s 5ms/step - loss: 1.7110
Epoch 81/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7047
Epoch 82/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6992
Epoch 83/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6930
Epoch 84/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6861
Epoch 85/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6819
```

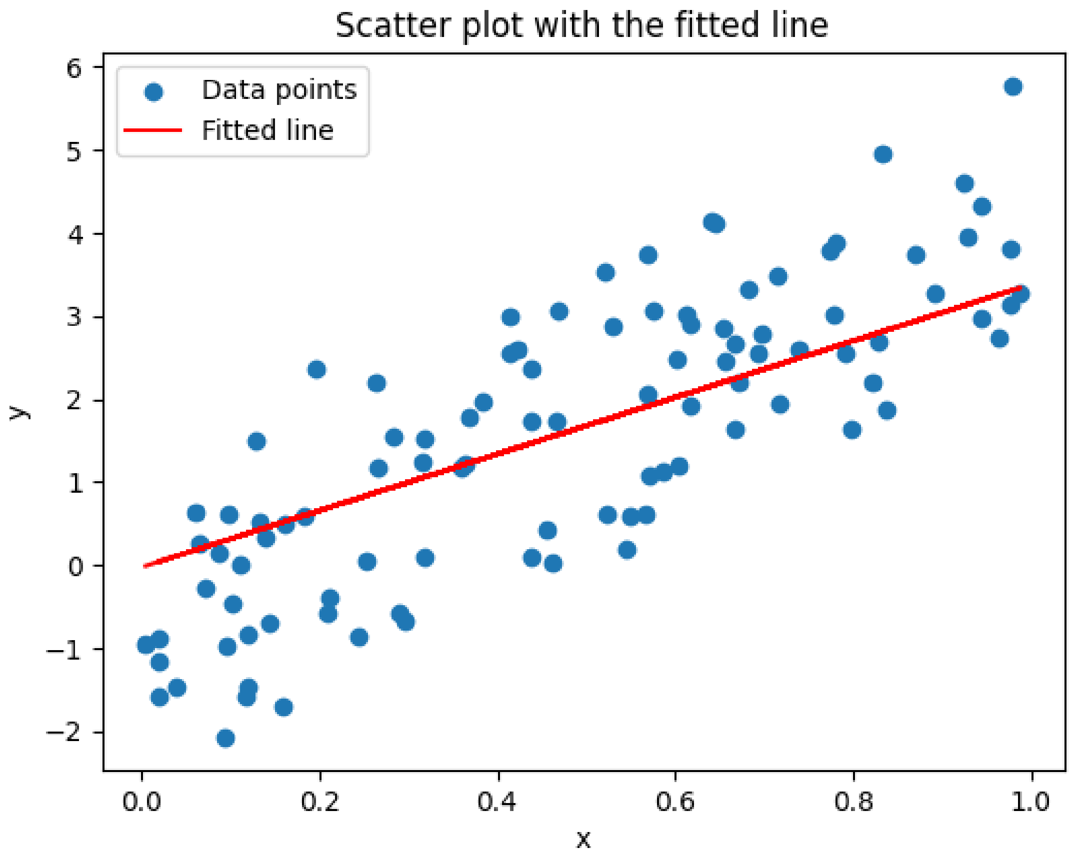
```
Epoch 86/200
4/4 [==============================] - 0s 5ms/step - loss: 1.6736
Epoch 87/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6666
Epoch 88/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6596
Epoch 89/200
4/4 [==============================] - 0s 5ms/step - loss: 1.6559
Epoch 90/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6488
Epoch 91/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6402
Epoch 92/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6296
Epoch 93/200
4/4 [==============================] - 0s 5ms/step - loss: 1.6232
Epoch 94/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6153
Epoch 95/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6108
Epoch 96/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6063
Epoch 97/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5984
Epoch 98/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5898
Epoch 99/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5853
Epoch 100/200
4/4 [==============================] - 0s 5ms/step - loss: 1.5796
Epoch 101/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5728
Epoch 102/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5657
Epoch 103/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5575
Epoch 104/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5507
Epoch 105/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5471
Epoch 106/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5392
Epoch 107/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5337
Epoch 108/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5298
Epoch 109/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5225
Epoch 110/200
```

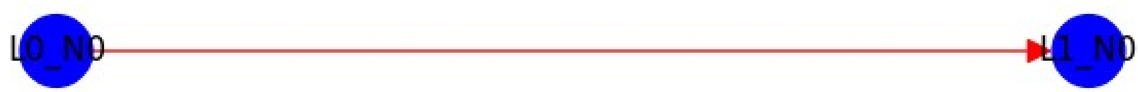
```
4/4 [==============================] - 0s 4ms/step - loss: 1.5174
Epoch 111/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5123
Epoch 112/200
4/4 [==============================] - 0s 5ms/step - loss: 1.5065
Epoch 113/200
4/4 [==============================] - 0s 7ms/step - loss: 1.5016
Epoch 114/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4975
Epoch 115/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4918
Epoch 116/200
4/4 [==============================] - 0s 5ms/step - loss: 1.4868
Epoch 117/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4816
Epoch 118/200
4/4 [==============================] - 0s 6ms/step - loss: 1.4773
Epoch 119/200
4/4 [==============================] - 0s 6ms/step - loss: 1.4745
Epoch 120/200
4/4 [==============================] - 0s 6ms/step - loss: 1.4694
Epoch 121/200
4/4 [==============================] - 0s 5ms/step - loss: 1.4610
Epoch 122/200
4/4 [==============================] - 0s 5ms/step - loss: 1.4573
Epoch 123/200
4/4 [==============================] - 0s 5ms/step - loss: 1.4533
Epoch 124/200
4/4 [==============================] - 0s 5ms/step - loss: 1.4477
Epoch 125/200
4/4 [==============================] - 0s 6ms/step - loss: 1.4429
Epoch 126/200
4/4 [==============================] - 0s 6ms/step - loss: 1.4381
Epoch 127/200
4/4 [==============================] - 0s 6ms/step - loss: 1.4341
Epoch 128/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4265
Epoch 129/200
4/4 [==============================] - 0s 6ms/step - loss: 1.4229
Epoch 130/200
4/4 [==============================] - 0s 6ms/step - loss: 1.4178
Epoch 131/200
4/4 [==============================] - 0s 5ms/step - loss: 1.4123
Epoch 132/200
4/4 [==============================] - 0s 5ms/step - loss: 1.4083
Epoch 133/200
4/4 [==============================] - 0s 6ms/step - loss: 1.4028
Epoch 134/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3964
```

```
Epoch 135/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3918
Epoch 136/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3890
Epoch 137/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3833
Epoch 138/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3801
Epoch 139/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3768
Epoch 140/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3714
Epoch 141/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3691
Epoch 142/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3641
Epoch 143/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3607
Epoch 144/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3587
Epoch 145/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3553
Epoch 146/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3518
Epoch 147/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3487
Epoch 148/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3459
Epoch 149/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3431
Epoch 150/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3394
Epoch 151/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3358
Epoch 152/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3317
Epoch 153/200
4/4 [==============================] - 0s 7ms/step - loss: 1.3275
Epoch 154/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3264
Epoch 155/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3207
Epoch 156/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3205
Epoch 157/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3167
Epoch 158/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3147
Epoch 159/200
```

```
4/4 [==============================] - 0s 5ms/step - loss: 1.3114
Epoch 160/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3061
Epoch 161/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3018
Epoch 162/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2955
Epoch 163/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2905
Epoch 164/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2877
Epoch 165/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2847
Epoch 166/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2811
Epoch 167/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2789
Epoch 168/200
4/4 [==============================] - 0s 6ms/step - loss: 1.2755
Epoch 169/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2721
Epoch 170/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2682
Epoch 171/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2647
Epoch 172/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2618
Epoch 173/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2604
Epoch 174/200
4/4 [==============================] - 0s 6ms/step - loss: 1.2552
Epoch 175/200
4/4 [==============================] - 0s 7ms/step - loss: 1.2523
Epoch 176/200
4/4 [==============================] - 0s 6ms/step - loss: 1.2499
Epoch 177/200
4/4 [==============================] - 0s 6ms/step - loss: 1.2466
Epoch 178/200
4/4 [==============================] - 0s 7ms/step - loss: 1.2452
Epoch 179/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2426
Epoch 180/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2400
Epoch 181/200
4/4 [==============================] - 0s 6ms/step - loss: 1.2368
Epoch 182/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2332
Epoch 183/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2308
```

```
Epoch 184/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2278
Epoch 185/200
4/4 [==============================] - 0s 7ms/step - loss: 1.2258
Epoch 186/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2236
Epoch 187/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2210
Epoch 188/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2212
Epoch 189/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2155
Epoch 190/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2130
Epoch 191/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2101
Epoch 192/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2081
Epoch 193/200
4/4 [==============================] - 0s 6ms/step - loss: 1.2057
Epoch 194/200
4/4 [==============================] - 0s 6ms/step - loss: 1.2012
Epoch 195/200
4/4 [==============================] - 0s 6ms/step - loss: 1.1987
Epoch 196/200
4/4 [==============================] - 0s 5ms/step - loss: 1.1975
Epoch 197/200
4/4 [==============================] - 0s 6ms/step - loss: 1.1950
Epoch 198/200
4/4 [==============================] - 0s 5ms/step - loss: 1.1922
Epoch 199/200
4/4 [==============================] - 0s 6ms/step - loss: 1.1908
Epoch 200/200
4/4 [==============================] - 0s 5ms/step - loss: 1.1899
4/4 [==============================] - 0s 3ms/step
```


Scatter plot with the fitted line
Data points
Fitted line
y
x
6
5
4
3
2
1
0
−1
−2
0.0
0.2
0.4
0.6
0.8
1.0



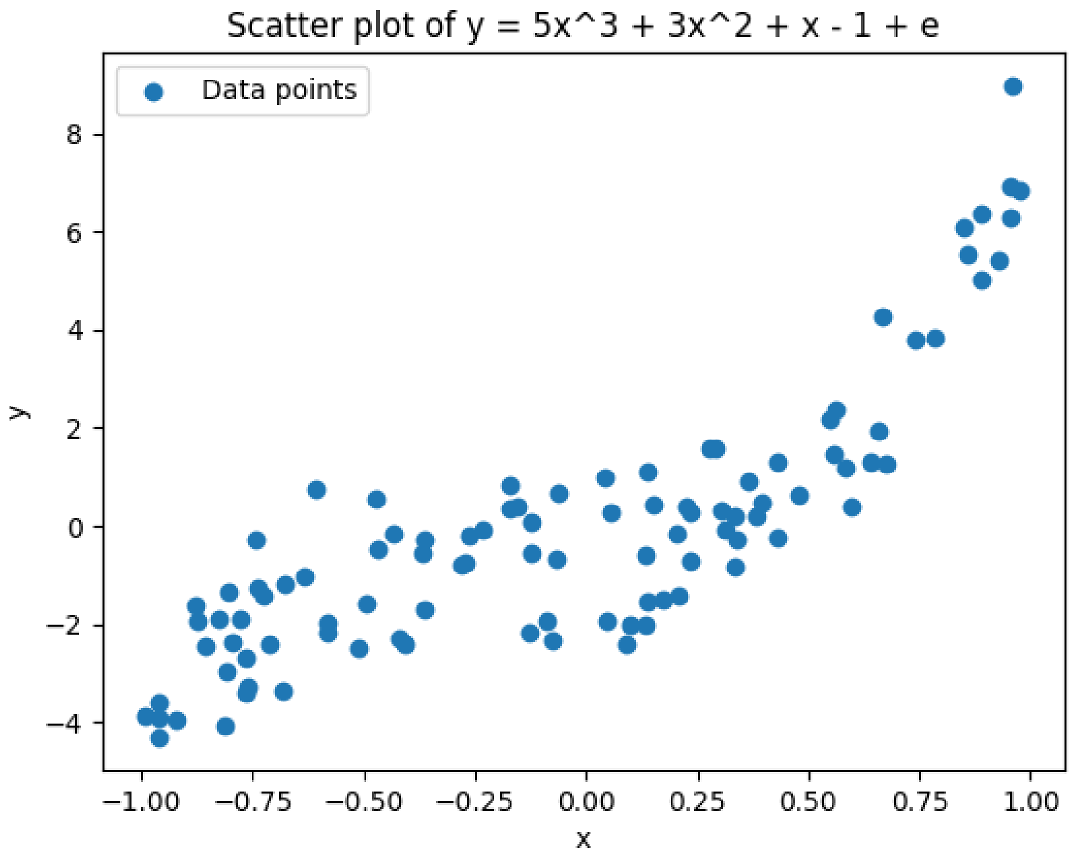
c) (15 points)Create a 3D animation (.gif) of the (weight, bias, loss) point over the training period.

d) Generate data of the form y = 5x^3 + 3x^2 + x - 1 + e where e ~ N(0, 1) and plot the data in a scatter plot. Create and train a neural network on this dataset and plot the resulting curve through the scatter plot. Explain your choice of model architecture (number of layers, and neurons) as well as your choice of activation function. (5points)

I used one input and one output layer as there is one input (x) and one output (y). I used activation function 'relu' since it is a non linear function that can output larger than 1. I used a single hidden layer with 10 neurons to be able to predict non linearity as a single neuron could not.

```
# y = 5x^3 + 3x^2 + x - 1 + e where e ~ N(0, 1)
np.random.seed(0)
x = np.random.rand(100, 1) * 2 - 1
e = np.random.randn(100, 1)
y = 5 * x**3 + 3 * x**2 + x - 1 + e

# scatter plot
```

```
plt.scatter(x, y, label='Data points')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatter plot of y = 5x^3 + 3x^2 + x - 1 + e')
plt.legend()
plt.show()


# Neural Network
model = Sequential()
model.add(Dense(units=10, input_shape=(1,), activation='relu')) #relu
cause non-linear
model.add(Dense(units=1))

# mean_squared_error loss and no activation function
model.compile(optimizer='sgd', loss='mean_squared_error')

# train
model.fit(x, y, epochs=200)


# predict
y_pred = model.predict(x)

sorted_indices = np.argsort(x[:, 0])
sorted_x = x[sorted_indices]
sorted_y_pred = y_pred[sorted_indices]


# plot fitted line
plt.scatter(x, y, label='Data points')
plt.plot(sorted_x, sorted_y_pred, color='red', label='Fitted line')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatter plot with the fitted line')
plt.legend()
plt.show()
```




```
Epoch 1/200
4/4 [==============================] - 0s 5ms/step - loss: 8.1180
Epoch 2/200
4/4 [==============================] - 0s 4ms/step - loss: 7.5792
Epoch 3/200
4/4 [==============================] - 0s 4ms/step - loss: 7.2204
Epoch 4/200
4/4 [==============================] - 0s 4ms/step - loss: 6.8571
Epoch 5/200
4/4 [==============================] - 0s 4ms/step - loss: 6.4758
Epoch 6/200
4/4 [==============================] - 0s 4ms/step - loss: 6.1366
Epoch 7/200
4/4 [==============================] - 0s 5ms/step - loss: 5.9140
Epoch 8/200
4/4 [==============================] - 0s 4ms/step - loss: 5.6925
Epoch 9/200
4/4 [==============================] - 0s 4ms/step - loss: 5.4456
Epoch 10/200
4/4 [==============================] - 0s 4ms/step - loss: 5.2755
Epoch 11/200
4/4 [==============================] - 0s 3ms/step - loss: 5.0972
Epoch 12/200
```

```
4/4 [==============================] - 0s 4ms/step - loss: 4.9142
Epoch 13/200
4/4 [==============================] - 0s 4ms/step - loss: 4.7189
Epoch 14/200
4/4 [==============================] - 0s 4ms/step - loss: 4.6159
Epoch 15/200
4/4 [==============================] - 0s 4ms/step - loss: 4.4756
Epoch 16/200
4/4 [==============================] - 0s 4ms/step - loss: 4.2652
Epoch 17/200
4/4 [==============================] - 0s 5ms/step - loss: 4.1340
Epoch 18/200
4/4 [==============================] - 0s 4ms/step - loss: 4.0162
Epoch 19/200
4/4 [==============================] - 0s 4ms/step - loss: 3.8554
Epoch 20/200
4/4 [==============================] - 0s 4ms/step - loss: 3.7373
Epoch 21/200
4/4 [==============================] - 0s 4ms/step - loss: 3.6118
Epoch 22/200
4/4 [==============================] - 0s 4ms/step - loss: 3.5127
Epoch 23/200
4/4 [==============================] - 0s 5ms/step - loss: 3.3990
Epoch 24/200
4/4 [==============================] - 0s 4ms/step - loss: 3.3081
Epoch 25/200
4/4 [==============================] - 0s 4ms/step - loss: 3.1918
Epoch 26/200
4/4 [==============================] - 0s 4ms/step - loss: 3.0611
Epoch 27/200
4/4 [==============================] - 0s 4ms/step - loss: 2.9624
Epoch 28/200
4/4 [==============================] - 0s 4ms/step - loss: 2.7954
Epoch 29/200
4/4 [==============================] - 0s 4ms/step - loss: 2.6804
Epoch 30/200
4/4 [==============================] - 0s 5ms/step - loss: 2.6218
Epoch 31/200
4/4 [==============================] - 0s 6ms/step - loss: 2.5259
Epoch 32/200
4/4 [==============================] - 0s 5ms/step - loss: 2.4959
Epoch 33/200
4/4 [==============================] - 0s 4ms/step - loss: 2.4436
Epoch 34/200
4/4 [==============================] - 0s 5ms/step - loss: 2.4226
Epoch 35/200
4/4 [==============================] - 0s 4ms/step - loss: 2.3716
Epoch 36/200
4/4 [==============================] - 0s 4ms/step - loss: 2.3418
```

```
Epoch 37/200
4/4 [==============================] - 0s 4ms/step - loss: 2.3378
Epoch 38/200
4/4 [==============================] - 0s 4ms/step - loss: 2.3003
Epoch 39/200
4/4 [==============================] - 0s 4ms/step - loss: 2.2594
Epoch 40/200
4/4 [==============================] - 0s 4ms/step - loss: 2.2391
Epoch 41/200
4/4 [==============================] - 0s 4ms/step - loss: 2.2327
Epoch 42/200
4/4 [==============================] - 0s 4ms/step - loss: 2.2247
Epoch 43/200
4/4 [==============================] - 0s 4ms/step - loss: 2.2202
Epoch 44/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1836
Epoch 45/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1706
Epoch 46/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1782
Epoch 47/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1589
Epoch 48/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1341
Epoch 49/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1414
Epoch 50/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1346
Epoch 51/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1145
Epoch 52/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1135
Epoch 53/200
4/4 [==============================] - 0s 4ms/step - loss: 2.1170
Epoch 54/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0964
Epoch 55/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0963
Epoch 56/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0906
Epoch 57/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0827
Epoch 58/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0674
Epoch 59/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0659
Epoch 60/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0525
Epoch 61/200
```

```
4/4 [==============================] - 0s 4ms/step - loss: 2.0562
Epoch 62/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0221
Epoch 63/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0196
Epoch 64/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0110
Epoch 65/200
4/4 [==============================] - 0s 4ms/step - loss: 2.0096
Epoch 66/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9929
Epoch 67/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9813
Epoch 68/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9805
Epoch 69/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9591
Epoch 70/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9615
Epoch 71/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9372
Epoch 72/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9166
Epoch 73/200
4/4 [==============================] - 0s 4ms/step - loss: 1.9068
Epoch 74/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8955
Epoch 75/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8978
Epoch 76/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8797
Epoch 77/200
4/4 [==============================] - 0s 5ms/step - loss: 1.8715
Epoch 78/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8530
Epoch 79/200
4/4 [==============================] - 0s 6ms/step - loss: 1.8387
Epoch 80/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8178
Epoch 81/200
4/4 [==============================] - 0s 4ms/step - loss: 1.8087
Epoch 82/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7758
Epoch 83/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7606
Epoch 84/200
4/4 [==============================] - 0s 5ms/step - loss: 1.7552
Epoch 85/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7458
```

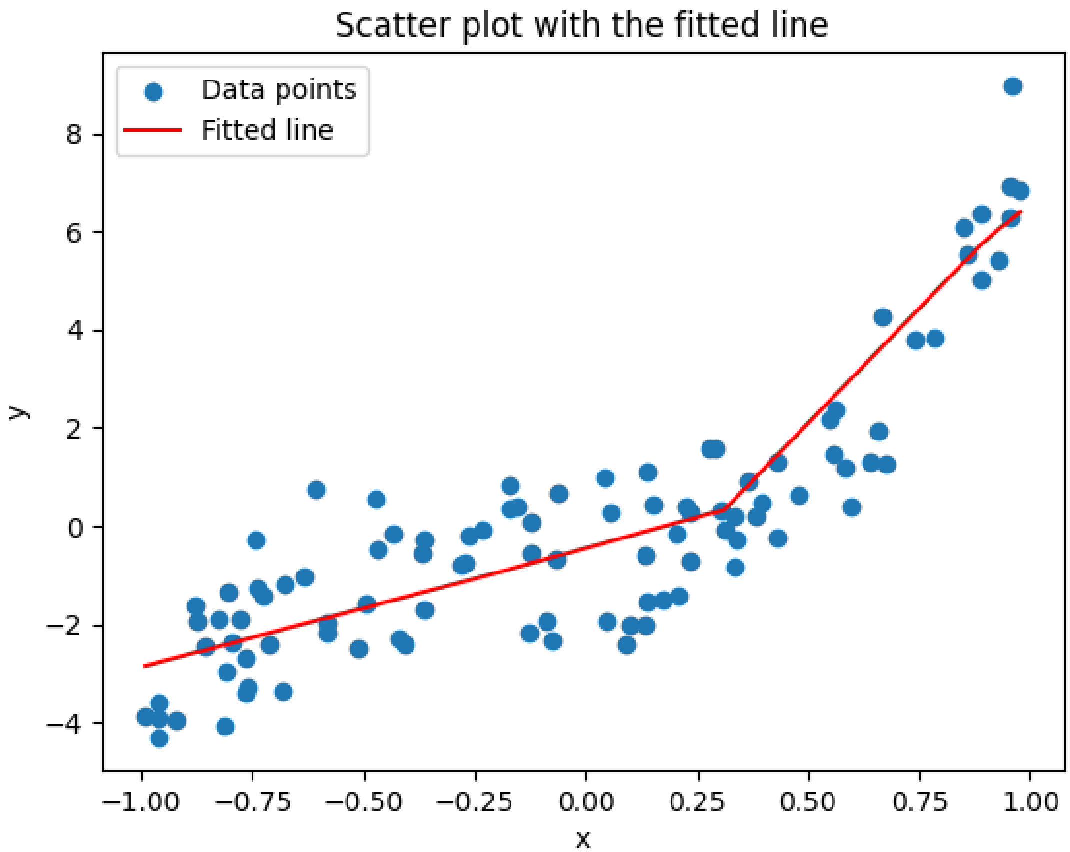
```
Epoch 86/200
4/4 [==============================] - 0s 5ms/step - loss: 1.7312
Epoch 87/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7222
Epoch 88/200
4/4 [==============================] - 0s 4ms/step - loss: 1.7589
Epoch 89/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6914
Epoch 90/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6781
Epoch 91/200
4/4 [==============================] - 0s 5ms/step - loss: 1.6616
Epoch 92/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6576
Epoch 93/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6551
Epoch 94/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6284
Epoch 95/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6193
Epoch 96/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6065
Epoch 97/200
4/4 [==============================] - 0s 4ms/step - loss: 1.6061
Epoch 98/200
4/4 [==============================] - 0s 5ms/step - loss: 1.6078
Epoch 99/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5843
Epoch 100/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5847
Epoch 101/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5852
Epoch 102/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5566
Epoch 103/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5478
Epoch 104/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5400
Epoch 105/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5493
Epoch 106/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5369
Epoch 107/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5489
Epoch 108/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5289
Epoch 109/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5328
Epoch 110/200
```

```
4/4 [==============================] - 0s 4ms/step - loss: 1.5087
Epoch 111/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5081
Epoch 112/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5055
Epoch 113/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4924
Epoch 114/200
4/4 [==============================] - 0s 4ms/step - loss: 1.5032
Epoch 115/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4980
Epoch 116/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4895
Epoch 117/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4903
Epoch 118/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4791
Epoch 119/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4910
Epoch 120/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4592
Epoch 121/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4671
Epoch 122/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4654
Epoch 123/200
4/4 [==============================] - 0s 5ms/step - loss: 1.4551
Epoch 124/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4672
Epoch 125/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4487
Epoch 126/200
4/4 [==============================] - 0s 5ms/step - loss: 1.4527
Epoch 127/200
4/4 [==============================] - 0s 6ms/step - loss: 1.4513
Epoch 128/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4361
Epoch 129/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4381
Epoch 130/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4376
Epoch 131/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4392
Epoch 132/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4277
Epoch 133/200
4/4 [==============================] - 0s 5ms/step - loss: 1.4309
Epoch 134/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4154
```

```
Epoch 135/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4191
Epoch 136/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4248
Epoch 137/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4472
Epoch 138/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4247
Epoch 139/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4246
Epoch 140/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4158
Epoch 141/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4137
Epoch 142/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4082
Epoch 143/200
4/4 [==============================] - 0s 4ms/step - loss: 1.4047
Epoch 144/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3923
Epoch 145/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3909
Epoch 146/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3900
Epoch 147/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3811
Epoch 148/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3890
Epoch 149/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3881
Epoch 150/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3822
Epoch 151/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3767
Epoch 152/200
4/4 [==============================] - 0s 6ms/step - loss: 1.3747
Epoch 153/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3839
Epoch 154/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3744
Epoch 155/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3700
Epoch 156/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3726
Epoch 157/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3706
Epoch 158/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3595
Epoch 159/200
```

```
4/4 [==============================] - 0s 4ms/step - loss: 1.3588
Epoch 160/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3581
Epoch 161/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3513
Epoch 162/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3659
Epoch 163/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3525
Epoch 164/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3456
Epoch 165/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3600
Epoch 166/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3468
Epoch 167/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3473
Epoch 168/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3424
Epoch 169/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3407
Epoch 170/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3355
Epoch 171/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3383
Epoch 172/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3336
Epoch 173/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3505
Epoch 174/200
4/4 [==============================] - 0s 7ms/step - loss: 1.3232
Epoch 175/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3265
Epoch 176/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3219
Epoch 177/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3133
Epoch 178/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3184
Epoch 179/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3237
Epoch 180/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3094
Epoch 181/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3094
Epoch 182/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3104
Epoch 183/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3313
```

```
Epoch 184/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3049
Epoch 185/200
4/4 [==============================] - 0s 5ms/step - loss: 1.3048
Epoch 186/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3164
Epoch 187/200
4/4 [==============================] - 0s 4ms/step - loss: 1.3192
Epoch 188/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2962
Epoch 189/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2946
Epoch 190/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2944
Epoch 191/200
4/4 [==============================] - 0s 5ms/step - loss: 1.2877
Epoch 192/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2841
Epoch 193/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2831
Epoch 194/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2830
Epoch 195/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2782
Epoch 196/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2867
Epoch 197/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2784
Epoch 198/200
4/4 [==============================] - 0s 6ms/step - loss: 1.2721
Epoch 199/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2734
Epoch 200/200
4/4 [==============================] - 0s 4ms/step - loss: 1.2799
4/4 [==============================] - 0s 3ms/step
```



e) Create an animation of the resulting curve learned by your model throughout the training process. (15points)

```
from tensorflow.keras.callbacks import LambdaCallback

predictions_per_epoch = []

def on_epoch_end(epoch, logs):
    sorted_indices = np.argsort(x[:, 0])
    sorted_x = x[sorted_indices]

    # predict results
    y_pred = model.predict(sorted_x)
    predictions_per_epoch.append(y_pred)

epoch_callback = LambdaCallback(on_epoch_end=on_epoch_end)

# train
model.fit(x, y, epochs=200, callbacks=[epoch_callback], verbose=0)

# animation
fig, ax = plt.subplots()
ax.scatter(x, y, label='Data points')
line, = ax.plot([], [], color='red', label='Fitted line')
```

```python
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_title('Training Progress')
ax.legend()

# initialization function
def init():
    line.set_data([], [])
    return line,

# update function
def update(frame):
    sorted_indices = np.argsort(x[:, 0])
    sorted_x = x[sorted_indices]
    y_pred = predictions_per_epoch[frame]
    line.set_data(sorted_x, y_pred)
    return line,

# animation
ani = FuncAnimation(fig, update, frames=len(predictions_per_epoch),
                    init_func=init, blit=True)
ani.save('model_training_animation.gif', writer='imagemagick', fps=15)

# plot
plt.show()
# show with html
HTML(ani.to_html5_video())
```

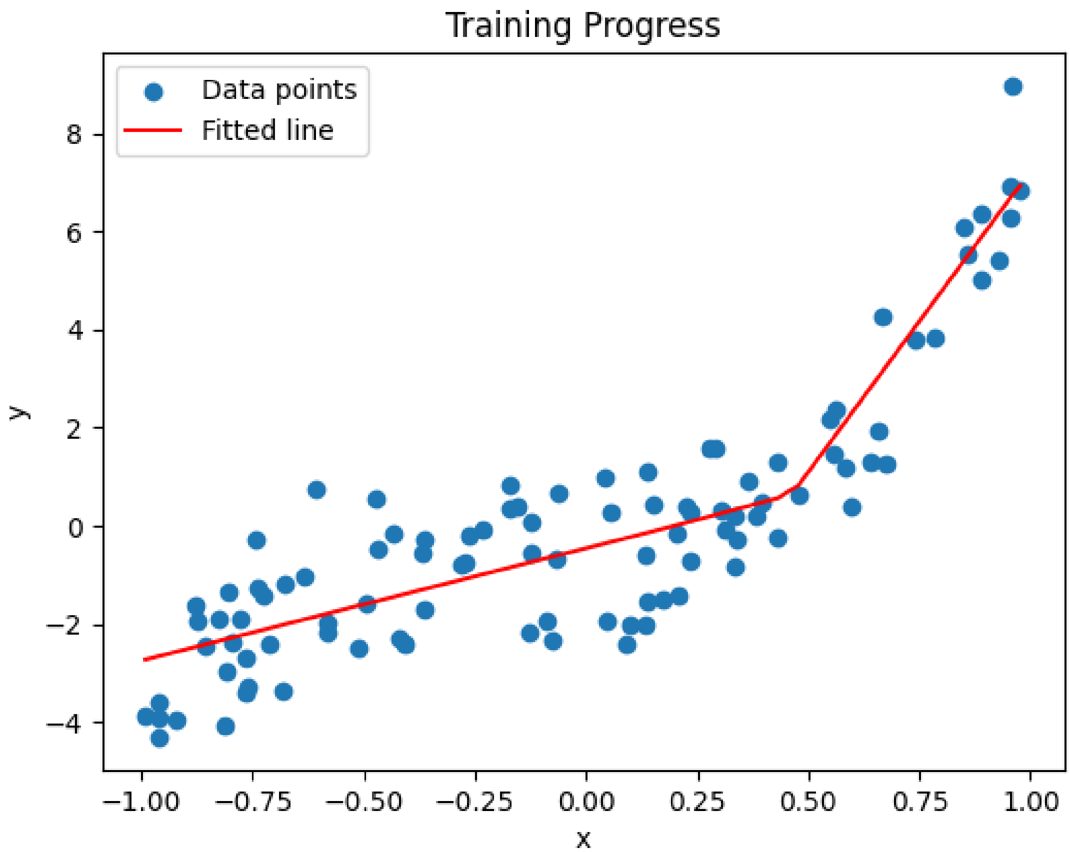
```
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
```

```
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
```

```
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 5ms/step
4/4 [==============================] - 0s 6ms/step
4/4 [==============================] - 0s 5ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 10ms/step
4/4 [==============================] - 0s 5ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
```

```
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
```

```
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
```




```
<IPython.core.display.HTML object>
```

f) Below is code to create a Generative Adversarial Network (GAN). The goal of the GAN is to generate data that is fake but looks real. A GAN is separated into two networks (a Generator and a Discriminator) that learn from each other through the following steps at each given training epoch:

1. The Generator generates data
2. The Discriminator is trained to learn how to distinguish real data from the fake data that the generator just generated.
3. The Generator is then trained to improve its ability to generate fake data by being informed by the Discriminators new ability to distinguish real from fake.

Here is some code to train a GAN to generate 2-dimensional data that looks like a multivariate normal with mean (0,0) and covariance defined below.

The code has one major flaw though that will prevent it from ever generating data that looks like the real data. Something is wrong with the architecture of the model (layers, activation etc). Find and fix that flaw and explain your reasoning below. (15points)

Since the mean should be (0, 0), this implies that the data can be negative or positive. The major flaw is that the activation function 'sigmoid' is used, which can only produce output between 0 and 1 as it is used for probability. Instead, negative outputs should be allowed by using elu, which can output negatives and values greater than 1.

```
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense
from PIL import Image as im

TEMPFILE = 'temp.png'

# Define the parameters
np.random.seed(0)
gen_input_dim = 100
epochs = 100
batch_size = 128
images = []

# Define the generator model
generator = Sequential()
generator.add(Dense(32, input_dim=gen_input_dim, activation='tanh'))

### MAJOR FLAW FIX###
generator.add(Dense(2, activation='elu'))

# Define the discriminator model
discriminator = Sequential()
discriminator.add(Dense(16, input_dim=2))
discriminator.add(Dense(1, activation='sigmoid'))

# Compile the models
generator.compile(loss='mse')
discriminator.compile(loss='binary_crossentropy')

# Define the GAN model
gan = Sequential()
gan.add(generator)
gan.add(discriminator)
gan.compile(loss='binary_crossentropy')

# Define the real data
x_real = np.random.multivariate_normal([0, 0], [[1, 0.5], [0.5, 1]],
1000)

# Train the GAN
# don't change the code below
for epoch in range(epochs):
    # Generate fake data
    z = np.random.normal(size=(batch_size, gen_input_dim))
    x_fake = generator.predict(z)
```

```
    # Train the discriminator
    discriminator.trainable = True
    discriminator.train_on_batch(x_real, np.ones((len(x_real), 1)))
    discriminator.train_on_batch(x_fake, np.zeros((batch_size, 1)))

    # Train the generator
    discriminator.trainable = False
    gan.train_on_batch(z, np.ones((batch_size, 1)))
print(x_real)
print('XXXXXXXXXXXXXXXXXXXXXXXXXXXXX')
print(x_fake)
```

```
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
```

```
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
```

```
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
[[-1.72779275 -1.32763554]
 [-1.96805856  0.27283464]
 [-1.12871372 -2.1059916 ]
 ...
 [-0.22019962 -0.12244882]
 [-1.29297181 -1.13453796]
 [ 1.64440082  0.33343045]]
XXXXXXXXXXXXXXXXXXXXXXXXXXXX
[[ 0.40095857 -0.5951562 ]
 [ 0.36588073  0.62420595]
 [-0.83166564 -0.11860833]
 [ 0.49415237 -0.21220726]
 [ 0.99143857 -0.01703348]
 [ 1.2496055  -0.16377456]
 [-0.15794812 -0.7282462 ]
 [ 0.38093987  0.64454097]
 [ 0.15894641 -0.2805776 ]
 [ 1.4512906  -0.38220158]
 [-0.7395365   0.25769272]
 [-0.619932   -0.46563238]
 [ 1.1190181   0.09061004]
 [-0.13478813 -0.7913577 ]
 [ 0.7149507   1.8430364 ]
 [-0.54339063  0.1776873 ]
 [-0.39484912  0.18618028]
 [-0.5003447   1.5721554 ]
 [ 1.084795   -0.22125463]
 [-0.64562047  0.20409267]
 [-0.37743068 -0.0813012 ]
 [-0.50767905  0.44804353]
 [ 0.7259755   0.4196157 ]
 [ 0.9804109   0.41786602]
 [ 1.1875378   0.13427469]
 [ 0.28496563 -0.25457484]
 [ 0.16474183 -0.04365679]
```

```
 [-0.6022046   0.6752514 ]
 [-0.8522341   0.9347736 ]
 [ 1.0036905   0.953901  ]
 [-0.61130345 -0.7020988 ]
 [ 0.5661567  -0.53452086]
 [-0.03699553 -0.39883918]
 [ 0.03933326 -0.47167233]
 [-0.0886663   1.2100397 ]
 [ 0.6219297   1.1154451 ]
 [-0.5907927   0.34002313]
 [-0.06212594  0.18999138]
 [ 0.4788149  -0.02032962]
 [ 0.42843312 -0.41796988]
 [ 0.39360976 -0.6390299 ]
 [ 0.96341413  1.098176  ]
 [ 0.41098672  2.1740077 ]
 [ 1.1374992  -0.47312352]
 [ 0.7154594  -0.8117871 ]
 [ 0.44256157 -0.6975273 ]
 [-0.7906791   1.3057567 ]
 [ 0.48417807  0.45574114]
 [-0.48321402  0.19545239]
 [ 0.7499466  -0.36795318]
 [ 0.03354977 -0.7742638 ]
 [ 0.660781   -0.20722726]
 [ 1.3449053   0.5481536 ]
 [ 0.62986577  0.5699562 ]
 [ 0.3849794   0.35212472]
 [ 0.19652231 -0.56653404]
 [ 0.09655528 -0.7580524 ]
 [-0.40125418  0.1519302 ]
 [ 0.70385617 -0.47305733]
 [ 0.7786417   0.09623734]
 [ 1.2168465  -0.5670844 ]
 [-0.02999017 -0.39299488]
 [ 0.5313928   1.1171563 ]
 [ 0.2891335   1.2566924 ]
 [ 0.49507165  1.0896764 ]
 [ 0.20705798 -0.33780646]
 [-0.78864616 -0.5621461 ]
 [-0.59840465 -0.46212554]
 [ 0.67504174  1.064584  ]
 [-0.7307278  -0.59375954]
 [ 0.7980951   1.994732  ]
 [ 0.39524528  0.00875134]
 [ 0.5270381  -0.6151427 ]
 [ 0.01138988 -0.29358032]
 [ 0.7230505   0.5286132 ]
 [ 0.91309637 -0.5149902 ]
```

```
 [-0.71585804  0.83996475]
 [ 0.5642892  -0.6917865 ]
 [-0.46194565  0.8661509 ]
 [-0.33641052  0.86561847]
 [-0.68454087 -0.05053628]
 [-0.5945012   0.7360333 ]
 [-0.10835905 -0.00850812]
 [-0.792463   -0.86897486]
 [ 0.34381428  1.4480336 ]
 [ 0.5657866   0.6304372 ]
 [ 0.19628316 -0.6637031 ]
 [-0.36764947 -0.05441792]
 [-0.44261137  0.8263563 ]
 [ 0.38074526 -0.6439109 ]
 [-0.23726176 -0.57827044]
 [-0.25272596 -0.5660938 ]
 [ 0.00540836 -0.5526245 ]
 [ 1.4201652   0.26867834]
 [ 0.38449     1.1339296 ]
 [ 0.26023397 -0.628899  ]
 [-0.25121304  0.45297286]
 [ 0.7640475   0.41745633]
 [ 1.3467212   0.8311205 ]
 [ 0.92236453 -0.01602192]
 [-0.20024762 -0.25838932]
 [-0.58479357 -0.6574608 ]
 [-0.42017138 -0.3459037 ]
 [ 0.37250388  0.37979528]
 [ 0.82533246  0.29410604]
 [-0.2962513   0.09689286]
 [ 0.82787126 -0.20725437]
 [ 0.55879825 -0.13596644]
 [ 0.39932513  0.08202972]
 [ 1.3442069   1.1844355 ]
 [ 0.58617836  0.96485925]
 [-0.3219779  -0.07604754]
 [-0.0964244  -0.6374477 ]
 [ 0.5204957  -0.4440889 ]
 [-0.50009036  1.087709  ]
 [ 0.07689099 -0.53778136]
 [ 0.49214637 -0.44627172]
 [ 1.0214386  -0.16452642]
 [-0.01152916 -0.6202908 ]
 [ 0.53025043  0.74139    ]
 [-0.38935164 -0.0655814 ]
 [ 1.2873056  -0.29821953]
 [-0.16981025 -0.37612107]
 [-0.5973433   0.41831115]
 [ 0.97303003 -0.558729  ]
```

```
 [-0.18279311 -0.5843772 ]
 [-0.6045989  -0.05453739]
 [-0.6595532  -0.4823406 ]]
```

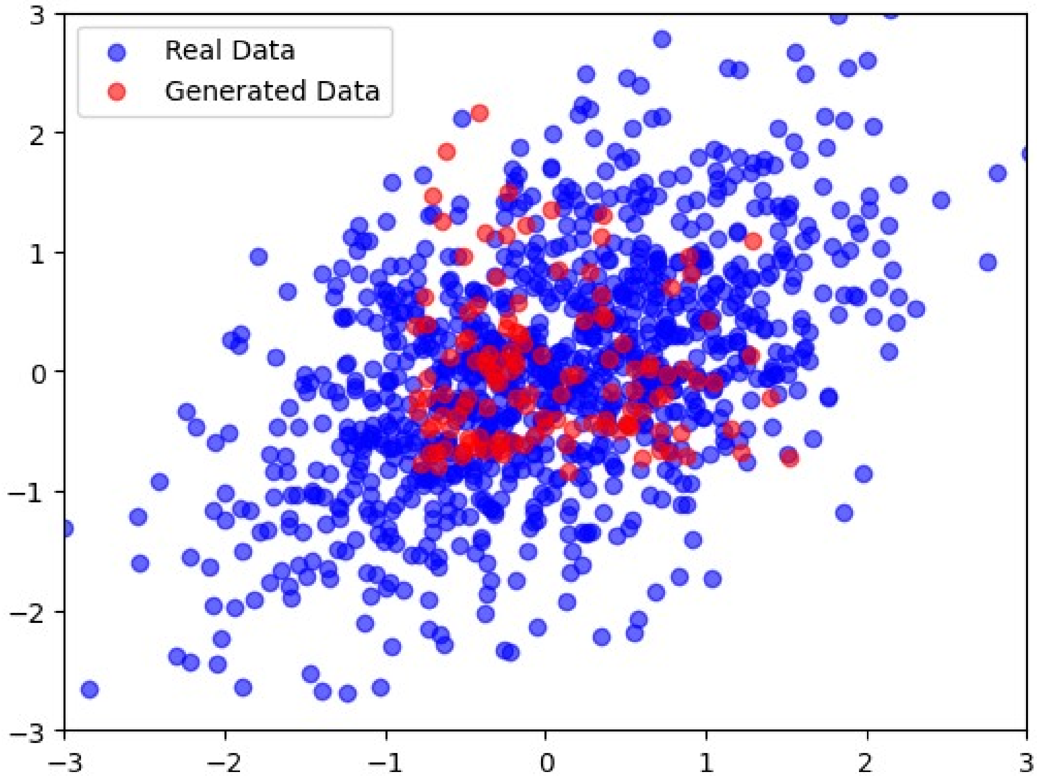
g) Create an animation of the generated data over the course of the training process (with the real data plotted in a different color for reference). (15points)

```
generated_data = []

# loop for each epoch
for epoch in range(epochs):
    z = np.random.normal(size=(batch_size, gen_input_dim))
    x_fake = generator.predict(z)
    generated_data.append(x_fake)

    discriminator.trainable = True
    discriminator.train_on_batch(x_real, np.ones((len(x_real), 1)))
    discriminator.train_on_batch(x_fake, np.zeros((batch_size, 1)))

    discriminator.trainable = False
    gan.train_on_batch(z, np.ones((batch_size, 1)))

# animation
fig, ax = plt.subplots()
real_data_scatter = ax.scatter(x_real[:, 0], x_real[:, 1],
color='blue', alpha=0.6, label='Real Data')
fake_data_scatter = ax.scatter([], [], color='red', alpha=0.6,
label='Generated Data')
ax.legend()
ax.set_xlim([-3, 3])
ax.set_ylim([-3, 3])

# update function
def update(epoch):
    fake_data = generated_data[epoch]
    fake_data_scatter.set_offsets(fake_data)
    return fake_data_scatter,

# animation
ani = FuncAnimation(fig, update, frames=epochs, interval=100,
blit=True)

ani.save('gan_training_animation.gif', writer='imagemagick')

# plot
plt.show()
# show using html
HTML(ani.to_html5_video())
```

```
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 6ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
```

```
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 4ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 5ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 2ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
```

```
4/4 [==============================] - 0s 3ms/step
4/4 [==============================] - 0s 4ms/step
```




```
<IPython.core.display.HTML object>
```

h) Tune the above model in order to generate data as close as possible to the real data. (15points)

```
TEMPFILE = 'temp.png'

# Define the parameters
np.random.seed(0)
gen_input_dim = 100
epochs = 100
batch_size = 128
images = []

# Define the generator model
generator = Sequential()
generator.add(Dense(32, input_dim=gen_input_dim, activation='tanh'))

### MAJOR FLAW FIX###
generator.add(Dense(2, activation='elu'))
```

```
# Define the discriminator model
discriminator = Sequential()
discriminator.add(Dense(16, input_dim=2))
discriminator.add(Dense(1, activation='sigmoid'))

# Compile the models
generator.compile(loss='mse')
discriminator.compile(loss='binary_crossentropy')

# Define the GAN model
gan = Sequential()
gan.add(generator)
gan.add(discriminator)
gan.compile(loss='binary_crossentropy')

# Define the real data
x_real = np.random.multivariate_normal([0, 0], [[1, 0.5], [0.5, 1]],
1000)

# Train the GAN
# don't change the code below
for epoch in range(epochs):
    # Generate fake data
    z = np.random.normal(size=(batch_size, gen_input_dim))
    x_fake = generator.predict(z)

    # Train the discriminator
    discriminator.trainable = True
    discriminator.train_on_batch(x_real, np.ones((len(x_real), 1)))
    discriminator.train_on_batch(x_fake, np.zeros((batch_size, 1)))

    # Train the generator
    discriminator.trainable = False
    gan.train_on_batch(z, np.ones((batch_size, 1)))
```

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
last)
/Users/thomaslawton/Thomas-Lawton-CS506/ps5/homework5.ipynb Cell 21
line 4
      <a href='vscode-notebook-cell:/Users/thomaslawton/Thomas-Lawton-
CS506/ps5/homework5.ipynb#X26sZmlsZQ%3D%3D?line=0'>1</a> TEMPFILE =
'temp.png'
      <a href='vscode-notebook-cell:/Users/thomaslawton/Thomas-Lawton-
CS506/ps5/homework5.ipynb#X26sZmlsZQ%3D%3D?line=2'>3</a> # Define the
parameters
----> <a href='vscode-notebook-cell:/Users/thomaslawton/Thomas-Lawton-
CS506/ps5/homework5.ipynb#X26sZmlsZQ%3D%3D?line=3'>4</a>
```

```
np.random.seed(0)
      <a href='vscode-notebook-cell:/Users/thomaslawton/Thomas-Lawton-CS506/ps5/homework5.ipynb#X26sZmlsZQ%3D%3D?line=4'>5</a> gen_input_dim = 100
      <a href='vscode-notebook-cell:/Users/thomaslawton/Thomas-Lawton-CS506/ps5/homework5.ipynb#X26sZmlsZQ%3D%3D?line=5'>6</a> epochs = 100

NameError: name 'np' is not defined
```