```
!pip install -U scikit-fuzzy
```

```
Requirement already satisfied: scikit-fuzzy in /usr/local/lib/python3.10/dist-packages (0.4.2)
Requirement already satisfied: numpy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-fuzzy) (1.25.2)
Requirement already satisfied: scipy>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from scikit-fuzzy) (1.11.4)
Requirement already satisfied: networkx>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from scikit-fuzzy) (3.3)
```

```
import numpy as np
import skfuzzy as fuzz
import tensorflow as tf
from skfuzzy import control as ctrl
import matplotlib.pyplot as plt
```

```
# Generating Example Data randomly selected temperature
temperature = np.random.randint(0, 101, 100)
desired_temperature = np.random.randint(0, 101, 100)
```

```
temperature.view()
```

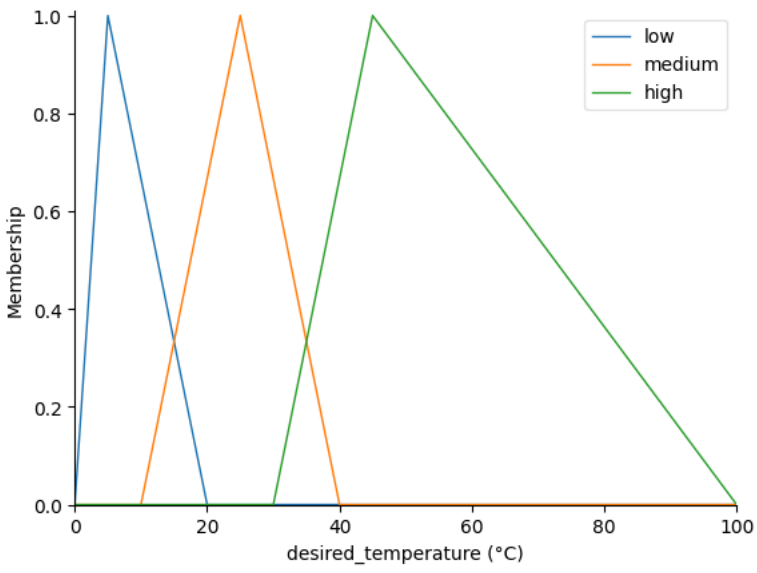
```
array([62, 95, 64, 79, 29, 40, 28, 39, 95, 51, 97, 91, 91, 97,  6, 37, 12,
        9, 85, 38, 89, 60, 80, 14, 72, 59, 91, 72, 50,  7, 30,  7, 20, 81,
       82, 32, 79, 65, 24, 68, 82, 22, 51, 90, 98, 49, 73, 30, 63, 83, 39,
       92, 66, 50, 57, 86, 63,  9, 25, 24, 14, 49, 17, 86, 83, 41, 28, 79,
       47, 39, 17, 16, 70, 66, 82, 32, 75, 73, 59, 56, 25, 72,  4,  6, 27,
       94, 11, 80, 61, 48, 69, 48, 55, 63,  3, 25, 69,  1, 74, 93])
```

```
desired_temperature.view()
```

```
array([  1,  66,  43,  69,  45,  73,  66,  51,  20,  43,  36,  90,  51,
        68,  61,  46,  64,  34,  54,  98,  28,  58,  58,  83,  27,  53,
        99,  97,   7,  78,  86,  72,  12,  40,  95,  38,  60,  27,  96,
        79,  81,  11,  13,  55,  45,  36,   9,  42,  79, 100,  27,  12,
        95,  57,  94,  17,  65,  78,  88,  65,   3,  12,  17,   3,  76,
        44,  46,  81,  88,  26,  29,  53,  72, 100,  75,   7,  83,  95,
        58,  86,  86,  19,  37,  93,  33,  24,  35,  40,  33,  49,  36,
        32,   6,  82,   7,  48,   2,  30, 100,  51])
```

```
# Defining Fuzzy Variables
temperature_input = ctrl.Antecedent(np.arange(0, 101, 1), 'temperature (°C)')
desired_temp_input = ctrl.Antecedent(np.arange(0, 101, 1), 'desired_temperature (°C)')
fan_speed = ctrl.Consequent(np.arange(0, 101, 1), 'fan_speed')
```

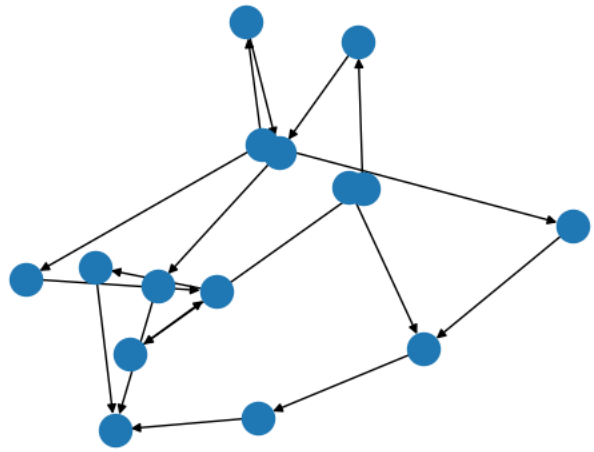
```
desired_temp_input.view()
```




```
# Define membership functions

# Temperature input
```

```
temperature_input['low'] = fuzz.trimf(temperature_input.universe, [0, 5, 20])
temperature_input['medium'] = fuzz.trimf(temperature_input.universe, [10, 25, 40])
temperature_input['high'] = fuzz.trimf(temperature_input.universe, [30, 45, 100])

# Desired temperature input
desired_temp_input['low'] = fuzz.trimf(desired_temp_input.universe, [0, 5, 20])
desired_temp_input['medium'] = fuzz.trimf(desired_temp_input.universe, [10, 25, 40])
desired_temp_input['high'] = fuzz.trimf(desired_temp_input.universe, [30, 45, 100])

# Fan speed output
fan_speed = ctrl.Consequent(np.arange(0, 101, 1), 'fan_speed')
fan_speed['low'] = fuzz.trimf(fan_speed.universe, [0, 0, 50])
fan_speed['medium'] = fuzz.trimf(fan_speed.universe, [0, 50, 100])
fan_speed['high'] = fuzz.trimf(fan_speed.universe, [50, 100, 100])


# Defining Fuzzy Rules
rule1 = ctrl.Rule(temperature_input['low'] | desired_temp_input['low'], fan_speed['high'])
rule2 = ctrl.Rule(temperature_input['medium'] & desired_temp_input['medium'], fan_speed['medium'])
rule3 = ctrl.Rule(temperature_input['high'] | desired_temp_input['high'], fan_speed['low'])

# Creating Fuzzy Control System
fan_speed_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
fan_speed = ctrl.ControlSystemSimulation(fan_speed_ctrl)
```

```
#View membership functions
fan_speed_ctrl.view()
```



```
# Generate some example data for prediction
test_data = np.column_stack((temperature, desired_temperature))

# Predicting using the Fuzzy Model
predicted_fan_speed = []
for data in test_data:
  fan_speed.input['temperature (°C)'] = data[0]
  fan_speed.input['desired_temperature (°C)'] = data[1]
  # Simulate the fuzzy system
  fan_speed.compute()
  # the actual output value
  predicted_fan_speed.append(fan_speed.output['fan_speed'])

print(predicted_fan_speed)
```

```
[34.8299465240642, 18.425039872408295, 16.928104575163395, 18.876102800153426, 16.666666666666664, 18.055555555555554, 18.425039872
```

```
print(fan_speed.output)
```

```
OrderedDict([('fan_speed', 16.845504222553398)])
```

```
# Generate data randomly selected temperature
temperature = np.random.randint(0, 101, 100)
desired_temperature = np.random.randint(0, 101, 100)
```

```
# Combine Fuzzy System Data for ANN
predicted_fuzzy_speed = []
for i in range(len(temperature)):
  # Simulate fuzzy system
  fan_speed.input['temperature (°C)'] = temperature[i]
  fan_speed.input['desired_temperature (°C)'] = desired_temperature[i]
  fan_speed.compute()

  # Access the fuzzy output
  predicted_fuzzy_speed.append(fan_speed.output['fan_speed'])

# Prepare data for ANN
combined_data = np.column_stack((temperature, desired_temperature, predicted_fuzzy_speed))

# Define and Train ANN Model
model = tf.keras.Sequential([
  tf.keras.layers.Dense(64, activation='relu', input_shape=(3,)),
  tf.keras.layers.Dense(32, activation='relu'),
  tf.keras.layers.Dense(1, activation='linear')
])

# Mean squared error loss
model.compile(optimizer='adam', loss='mse')

# Train the model
model.fit(combined_data, np.random.rand(100), epochs=100)

# Generate some example data for prediction
test_data = np.column_stack((temperature, desired_temperature, np.zeros(len(temperature))))

# Predict fan speed (fuzzy system + ANN)
predicted_fan_speed = []
for i in range(len(test_data)):
  # Simulate fuzzy system
  fan_speed.input['temperature (°C)'] = test_data[i][0]
  fan_speed.input['desired_temperature (°C)'] = test_data[i][1]
  fan_speed.compute()
  test_data[i][2] = fan_speed.output['fan_speed']

  # Predict using the ANN model
  predicted_fan_speed.append(model.predict(np.array([test_data[i]]))[0][0])

# Print the predicted fan speed
print(predicted_fan_speed)
```

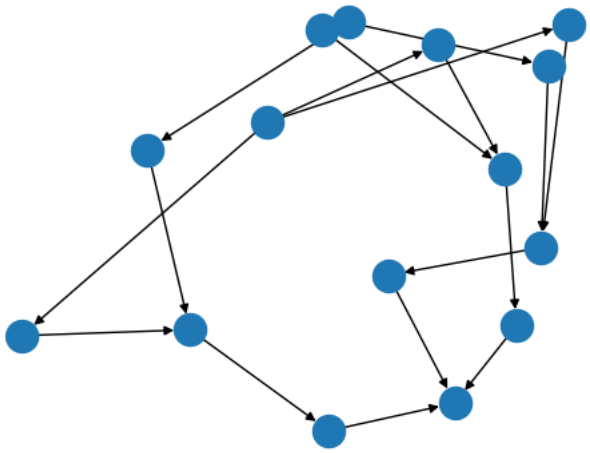
```
Epoch 1/100
4/4 [==============================] - 2s 16ms/step - loss: 25.2840
Epoch 2/100
4/4 [==============================] - 0s 8ms/step - loss: 12.6721
Epoch 3/100
4/4 [==============================] - 0s 9ms/step - loss: 8.5734
Epoch 4/100
4/4 [==============================] - 0s 11ms/step - loss: 2.0820
Epoch 5/100
4/4 [==============================] - 0s 15ms/step - loss: 6.2187
Epoch 6/100
4/4 [==============================] - 0s 10ms/step - loss: 1.4825
Epoch 7/100
4/4 [==============================] - 0s 10ms/step - loss: 1.4179
Epoch 8/100
4/4 [==============================] - 0s 23ms/step - loss: 1.3796
Epoch 9/100
4/4 [==============================] - 0s 6ms/step - loss: 0.6334
Epoch 10/100
4/4 [==============================] - 0s 13ms/step - loss: 1.1022
Epoch 11/100
4/4 [==============================] - 0s 15ms/step - loss: 0.3883
Epoch 12/100
4/4 [==============================] - 0s 14ms/step - loss: 0.4875
Epoch 13/100
4/4 [==============================] - 0s 10ms/step - loss: 0.3167
Epoch 14/100
4/4 [==============================] - 0s 11ms/step - loss: 0.3472
Epoch 15/100
4/4 [==============================] - 0s 21ms/step - loss: 0.2711
Epoch 16/100
4/4 [==============================] - 0s 11ms/step - loss: 0.2230
Epoch 17/100
4/4 [==============================] - 0s 11ms/step - loss: 0.2357
Epoch 18/100
4/4 [==============================] - 0s 13ms/step - loss: 0.2143
Epoch 19/100
4/4 [==============================] - 0s 7ms/step - loss: 0.1865
Epoch 20/100
```

```
4/4 [==============================] - 0s 11ms/step - loss: 0.2150
Epoch 21/100
4/4 [==============================] - 0s 10ms/step - loss: 0.2208
Epoch 22/100
4/4 [==============================] - 0s 11ms/step - loss: 0.1747
Epoch 23/100
4/4 [==============================] - 0s 10ms/step - loss: 0.1526
Epoch 24/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1578
Epoch 25/100
4/4 [==============================] - 0s 7ms/step - loss: 0.1655
Epoch 26/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1440
Epoch 27/100
4/4 [==============================] - 0s 17ms/step - loss: 0.1205
Epoch 28/100
4/4 [==============================] - 0s 11ms/step - loss: 0.1501
Epoch 29/100
```

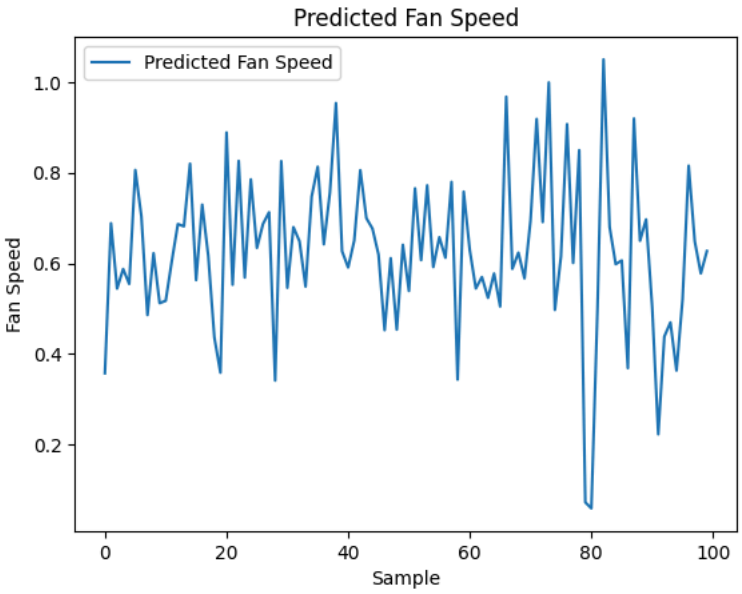
```
# Print the predicted fan speed
print(predicted_fan_speed)
```

```
[0.35777944, 0.688528, 0.54441875, 0.5876457, 0.5547177, 0.80595106, 0.70375246, 0.48615962, 0.62241715, 0.51234907, 0.51761025, 0.6
```

```
#View membership functions after ANN
fan_speed_ctrl.view()
```



```
# Plot predicted fan speed
plt.figure()
plt.plot(predicted_fan_speed, label='Predicted Fan Speed')
plt.xlabel('Sample')
plt.ylabel('Fan Speed')
plt.title('Predicted Fan Speed')
plt.legend()
plt.show()
```


Predicted Fan Speed
Predicted Fan Speed
Fan Speed
Sample
1.0
0.8
0.6
0.4
0.2
0
20
40
60
80
100