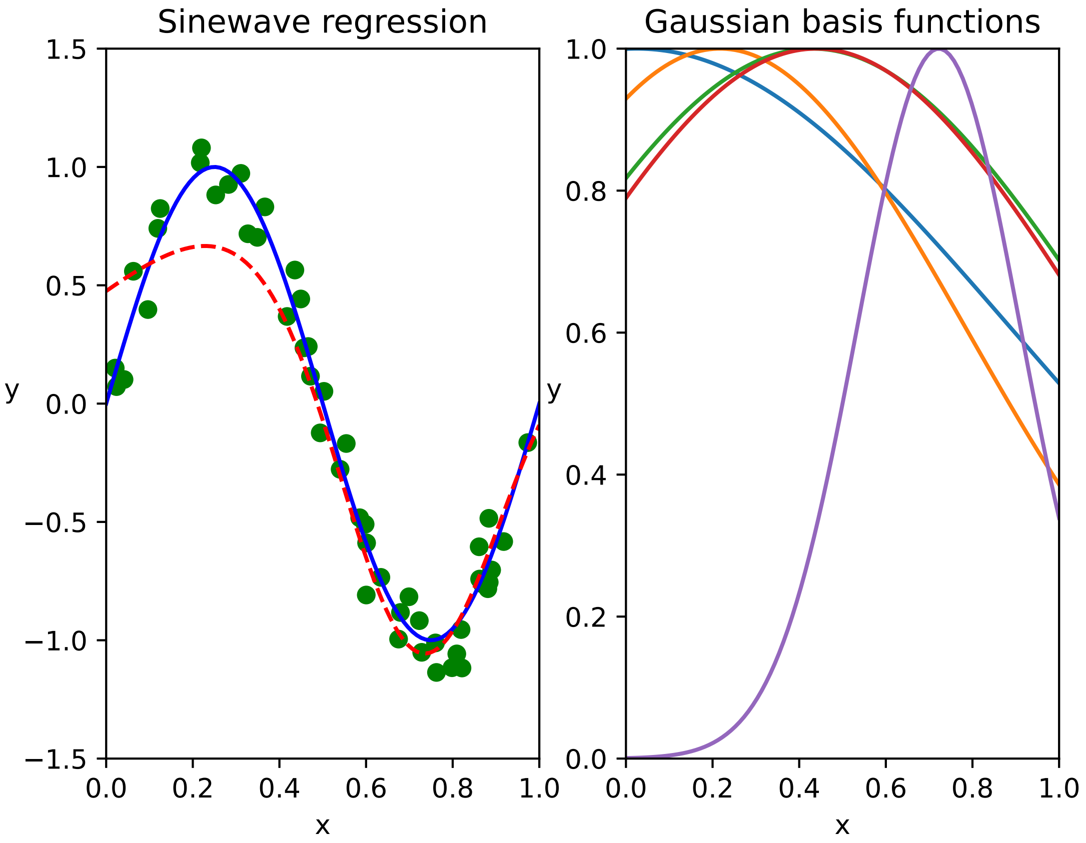
```python
# ECE472-Samuel Maltz
# Assignment 1: Linear regression of noisy sinewave on gaussian basis functions

import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

N = 50
M = 5
batch_size = 16
num_iter = 300
learning_rate = 0.1
sigma_noise = 0.1


class Model(tf.Module):
    def __init__(self, num_gaussians):
        # Learnable parameters
        self.w = tf.Variable(tf.random.normal(shape=[num_gaussians, 1]))
        self.b = tf.Variable(tf.zeros(shape=[1, 1]))
        self.mu = tf.Variable(tf.linspace(0.0, 1.0, num_gaussians))
        self.sigma = tf.Variable(tf.ones(shape=[1, num_gaussians]))

    def __call__(self, x):
        gaussians = tf.exp(-(((x - self.mu) / self.sigma) ** 2))
        return tf.squeeze(gaussians @ self.w + self.b)


def main():
    index = np.arange(N)

    # Create dataset
    x_data = np.random.uniform(0.0, 1.0, size=(N, 1)).astype("float32")
    y_data = np.sin(2 * np.pi * x_data) + np.random.normal(
        scale=sigma_noise, size=(N, 1)
    )

    model = Model(M)
    optimizer = tf.optimizers.SGD(learning_rate=learning_rate)

    for i in range(num_iter):
        # Select random batch
        ind = np.random.choice(index, batch_size)
        x = x_data[ind]
        y = y_data[ind].flatten()
        with tf.GradientTape() as tape:
            y_hat = model(x)
            loss = 0.5 * tf.reduce_mean((y - y_hat) ** 2)

        # Apply automatic differentiation
        gradients = tape.gradient(loss, model.trainable_variables)
        optimizer.apply_gradients(zip(gradients, model.trainable_variables))

    # Plots
    x_noiseless = np.linspace(0.0, 1.0, 100)
    y_noiseless = np.sin(2 * np.pi * x_noiseless)
    y_gaussians = np.exp(
        -(((x_noiseless.reshape(-1, 1) - model.mu.numpy()) / model.sigma.numpy()
) ** 2)
    )
    y_regression = y_gaussians @ model.w.numpy() + model.b.numpy()
```

```python
    plt.figure()
    plt.subplot(121)
    plt.plot(
        x_data,
        y_data,
        "go",
        x_noiseless,
        y_noiseless,
        "b",
        x_noiseless,
        y_regression,
        "r--",
    )
    plt.xlim((0, 1))
    plt.ylim((-1.5, 1.5))
    plt.xlabel("x")
    y_label = plt.ylabel("y")
    y_label.set_rotation(0)
    plt.title("Sinewave regression")

    plt.subplot(122)
    plt.plot(x_noiseless, y_gaussians)
    plt.xlim((0, 1))
    plt.ylim((0, 1))
    plt.xlabel("x")
    y_label = plt.ylabel("y")
    y_label.set_rotation(0)
    plt.title("Gaussian basis functions")
    plt.savefig("fit.pdf")


if __name__ == "__main__":
    main()
```


Sinewave regression
Gaussian basis functions
y
x
y
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.0
0.2
0.4
0.6
0.8
1.0