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


## want to initalize our weights in a very specific way
class Linear(tf.Module):
    def __init__(self, num_inputs, num_outputs, first_layer=True, bias=True):
        super().__init__()
        rng = tf.random.get_global_generator()

        self.first_layer = first_layer

        # Look into what this needs to be
        if self.first_layer:
            self.w = tf.Variable(
                rng.uniform(
                    shape=[num_inputs, num_outputs],
                    minval=(-1 / num_inputs),
                    maxval=(1 / num_inputs),
                ),
                trainable=True,
                name="Linear/w",
            )

            """
            Hence, we propose to draw weights with c=6
            so that wi  U(- â\210\2326/n, â\210\2326/n). This ensures that
            the input to each sine activation is normal
            distributed with a standard deviation of 1.
            """
        else:
            self.w = tf.Variable(
                rng.uniform(
                    shape=[num_inputs, num_outputs],
                    minval=-np.sqrt(6 / num_inputs) / 30,
                    maxval=np.sqrt(6 / num_inputs) / 30,
                ),
                trainable=True,
                name="Linear/w",
            )

        self.bias = bias

        if self.bias:
            self.b = tf.Variable(
                tf.zeros(
                    shape=[1, num_outputs],
                ),
                trainable=True,
                name="Linear/b",
            )

    def __call__(self, x):
        z = x @ self.w

        if self.bias:
            z += self.b

        return z
```