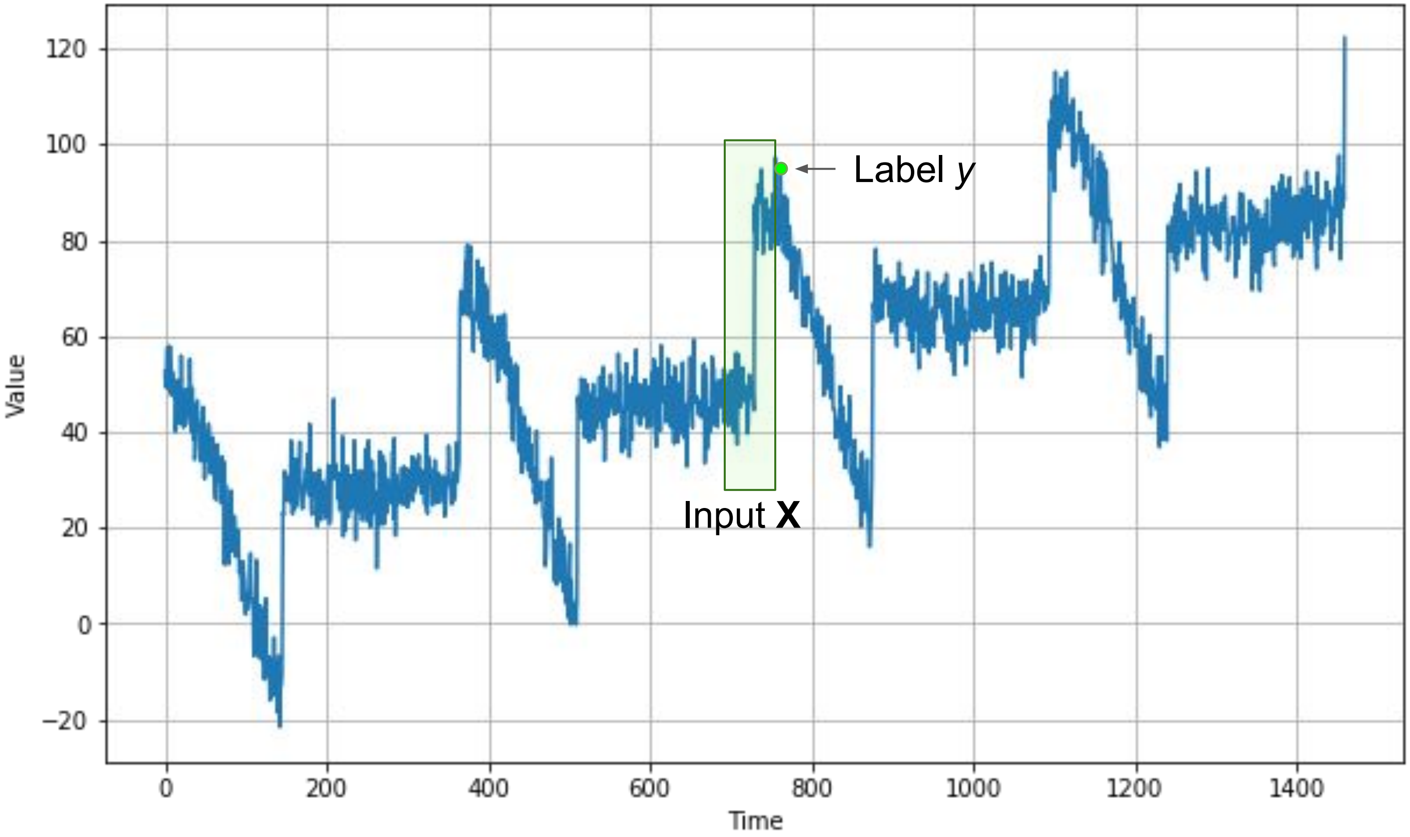
Copyright Notice

Machine Learning on Time Windows



```
dataset = tf.data.Dataset.range(10)
for val in dataset:
   print(val.numpy())
```

```python
dataset = tf.data.Dataset.range(10)
for val in dataset:
   print(val.numpy())
```

```
0
1
2
3
4
5
6
7
8
9
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1)
for window_dataset in dataset:
  for val in window_dataset:
    print(val.numpy(), end=" ")
  print()
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1)
for window_dataset in dataset:
  for val in window_dataset:
    print(val.numpy(), end=" ")
  print()
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1)
for window_dataset in dataset:
 for val in window_dataset:
  print(val.numpy(), end=" ")
 print()
```

```
0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
6 7 8 9
7 8 9
8 9
9
```

```python
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
for window_dataset in dataset:
  for val in window_dataset:
    print(val.numpy(), end=" ")
  print()
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
for window_dataset in dataset:
  for val in window_dataset:
    print(val.numpy(), end=" ")
  print()
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
for window_dataset in dataset:
  for val in window_dataset:
    print(val.numpy(), end=" ")
  print()
```

```
0 1 2 3 4
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
```

```python
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
for window in dataset:
  print(window.numpy())
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
for window in dataset:
  print(window.numpy())
```

```python
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
for window in dataset:
  print(window.numpy())
```

```
[0 1 2 3 4]
[1 2 3 4 5]
[2 3 4 5 6]
[3 4 5 6 7]
[4 5 6 7 8]
[5 6 7 8 9]
```

```python
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1]))
for x,y in dataset:
  print(x.numpy(), y.numpy())
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1]))
for x,y in dataset:
  print(x.numpy(), y.numpy())
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1]))
for x,y in dataset:
  print(x.numpy(), y.numpy())
```

```
[0 1 2 3] [4]
[1 2 3 4] [5]
[2 3 4 5] [6]
[3 4 5 6] [7]
[4 5 6 7] [8]
[5 6 7 8] [9]
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1]))
dataset = dataset.shuffle(buffer_size=10)
for x,y in dataset:
  print(x.numpy(), y.numpy())
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1]))
dataset = dataset.shuffle(buffer_size=10)
for x,y in dataset:
  print(x.numpy(), y.numpy())
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1:]))
dataset = dataset.shuffle(buffer_size=10)
for x,y in dataset:
  print(x.numpy(), y.numpy())
```

```
    [3 4 5 6] [7]
    [4 5 6 7] [8]
    [1 2 3 4] [5]
    [2 3 4 5] [6]
    [5 6 7 8] [9]
    [0 1 2 3] [4]
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1]))
dataset = dataset.shuffle(buffer_size=10)
dataset = dataset.batch(2).prefetch(1)
for x,y in dataset:
  print("x = ", x.numpy())
  print("y = ", y.numpy())
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1]))
dataset = dataset.shuffle(buffer_size=10)
dataset = dataset.batch(2).prefetch(1)
for x,y in dataset:
  print("x = ", x.numpy())
  print("y = ", y.numpy())
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1]))
dataset = dataset.shuffle(buffer_size=10)
dataset = dataset.batch(2).prefetch(1)
for x,y in dataset:
  print("x = ", x.numpy())
  print("y = ", y.numpy())
```

```
x =  [[4 5 6 7] [1 2 3 4]]
y =  [[8] [5]]
x =  [[3 4 5 6] [2 3 4 5]]
y =  [[7] [6]]
x =  [[5 6 7 8] [0 1 2 3]]
y =  [[9] [4]]
```

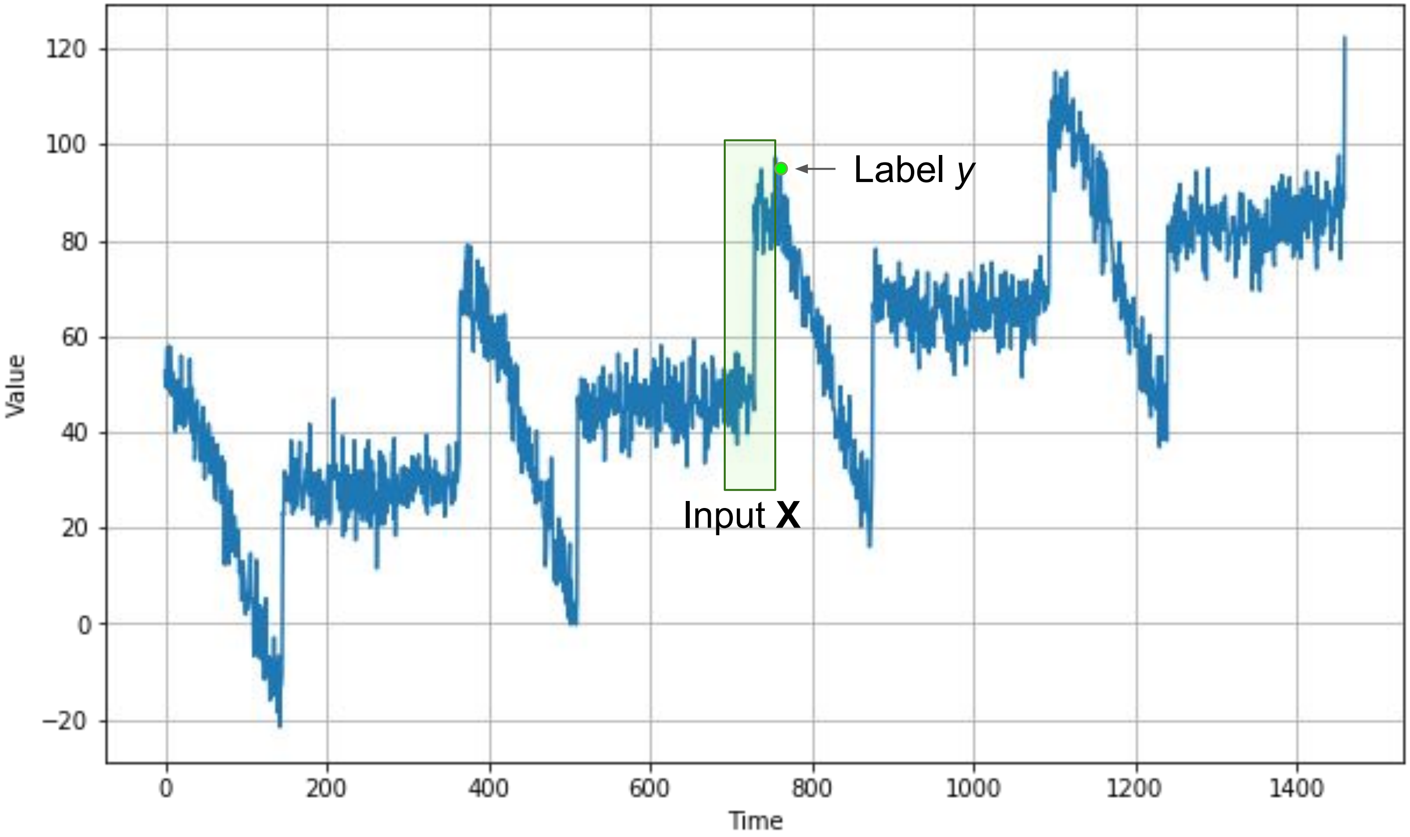
```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1]))
dataset = dataset.shuffle(buffer_size=10)
dataset = dataset.batch(2).prefetch(1)
for x,y in dataset:
  print("x = ", x.numpy())
  print("y = ", y.numpy())
```

```
x =  [[4 5 6 7] [1 2 3 4]]
y =  [[8] [5]]
x =  [[3 4 5 6] [2 3 4 5]]
y =  [[7] [6]]
x =  [[5 6 7 8] [0 1 2 3]]
y =  [[9] [4]]
```

```
dataset = tf.data.Dataset.range(10)
dataset = dataset.window(5, shift=1, drop_remainder=True)
dataset = dataset.flat_map(lambda window: window.batch(5))
dataset = dataset.map(lambda window: (window[:-1], window[-1]))
dataset = dataset.shuffle(buffer_size=10)
dataset = dataset.batch(2).prefetch(1)
for x,y in dataset:
  print("x = ", x.numpy())
  print("y = ", y.numpy())
```

```
x =  [[4 5 6 7] [1 2 3 4]]
y =  [[8] [5]]
x =  [[3 4 5 6] [2 3 4 5]]
y =  [[7] [6]]
x =  [[5 6 7 8] [0 1 2 3]]
y =  [[9] [4]]
```

Machine Learning on Time Windows



```python
def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
  dataset = tf.data.Dataset.from_tensor_slices(series)
  dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
  dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
  dataset = dataset.shuffle(shuffle_buffer)
  dataset = dataset.map(lambda window: (window[:-1], window[-1]))
  dataset = dataset.batch(batch_size).prefetch(1)
  return dataset
```

```python
def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
  dataset = tf.data.Dataset.from_tensor_slices(series)
  dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
  dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
  dataset = dataset.shuffle(shuffle_buffer)
  dataset = dataset.map(lambda window: (window[:-1], window[-1]))
  dataset = dataset.batch(batch_size).prefetch(1)
  return dataset
```

```
def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
  dataset = tf.data.Dataset.from_tensor_slices(series)
  dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
  dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
  dataset = dataset.shuffle(shuffle_buffer)
  dataset = dataset.map(lambda window: (window[:-1], window[-1]))
  dataset = dataset.batch(batch_size).prefetch(1)
  return dataset
```

```python
def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
  dataset = tf.data.Dataset.from_tensor_slices(series)
  dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
  dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
  dataset = dataset.shuffle(shuffle_buffer)
  dataset = dataset.map(lambda window: (window[:-1], window[-1]))
  dataset = dataset.batch(batch_size).prefetch(1)
  return dataset
```

```
def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
  dataset = tf.data.Dataset.from_tensor_slices(series)
  dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
  dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
  dataset = dataset.shuffle(shuffle_buffer)
  dataset = dataset.map(lambda window: (window[:-1], window[-1]))
  dataset = dataset.batch(batch_size).prefetch(1)
  return dataset
```

```python
def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
  dataset = tf.data.Dataset.from_tensor_slices(series)
  dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
  dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
  dataset = dataset.shuffle(shuffle_buffer)
  dataset = dataset.map(lambda window: (window[:-1], window[-1]))
  dataset = dataset.batch(batch_size).prefetch(1)
  return dataset
```

```
def windowed_dataset(series, window_size, batch_size, shuffle_buffer):
  dataset = tf.data.Dataset.from_tensor_slices(series)
  dataset = dataset.window(window_size + 1, shift=1, drop_remainder=True)
  dataset = dataset.flat_map(lambda window: window.batch(window_size + 1))
  dataset = dataset.shuffle(shuffle_buffer)
  dataset = dataset.map(lambda window: (window[:-1], window[-1]))
  dataset = dataset.batch(batch_size).prefetch(1)
  return dataset
```

```
split_time = 1000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]
```

```python
split_time = 1000
time_train = time[:split_time]
x_train = series[:split_time]
time_valid = time[split_time:]
x_valid = series[split_time:]
```

```
window_size = 20
batch_size = 32
shuffle_buffer_size = 1000

dataset = windowed_dataset(series, window_size, batch_size, shuffle_buffer_size)
l0 = tf.keras.layers.Dense(1)
model = tf.keras.models.Sequential([
     tf.keras.Input(shape=(window_size,)),
     l0,
])
```

```
window_size = 20
batch_size = 32
shuffle_buffer_size = 1000

dataset = windowed_dataset(series, window_size, batch_size, shuffle_buffer_size)
l0 = tf.keras.layers.Dense(1)
model = tf.keras.models.Sequential([
     tf.keras.Input(shape=(window_size,)),
     l0,
])
```

```
window_size = 20
batch_size = 32
shuffle_buffer_size = 1000

dataset = windowed_dataset(series, window_size, batch_size, shuffle_buffer_size)
l0 = tf.keras.layers.Dense(1)
model = tf.keras.models.Sequential([
     tf.keras.Input(shape=(window_size,)),
     l0,
])
```

```
window_size = 20
batch_size = 32
shuffle_buffer_size = 1000

dataset = windowed_dataset(series, window_size, batch_size, shuffle_buffer_size)
l0 = tf.keras.layers.Dense(1)
model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size,)),
    l0,
])
```

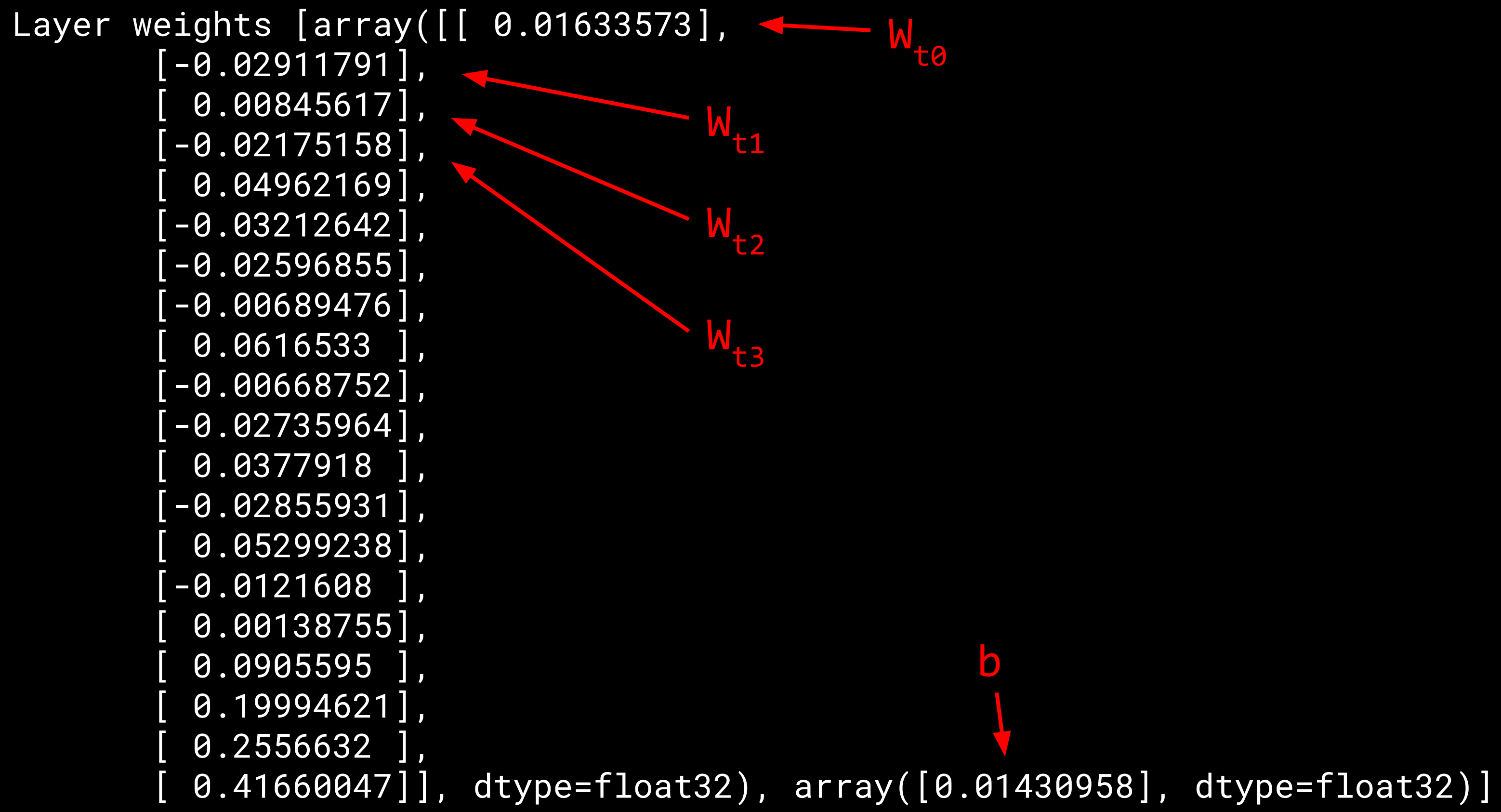
```
window_size = 20
batch_size = 32
shuffle_buffer_size = 1000

dataset = windowed_dataset(series, window_size, batch_size, shuffle_buffer_size)
l0 = tf.keras.layers.Dense(1)
model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size,)),
    l0,
])
```

```
model.compile(
     loss="mse",
     optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9)
)
model.fit(dataset, epochs=100)
model.compile(
    loss="mse",
    optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9)
)
model.fit(dataset, epochs=100)
model.compile(
    loss="mse",
    optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9)
)
model.fit(dataset, epochs=100)
model.compile(
     loss="mse",
     optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9)
)
model.fit(dataset, epochs=100)
```

```
print("Layer weights {}".format(l0.get_weights()))
print("Layer weights {}".format(l0.get_weights()))
```

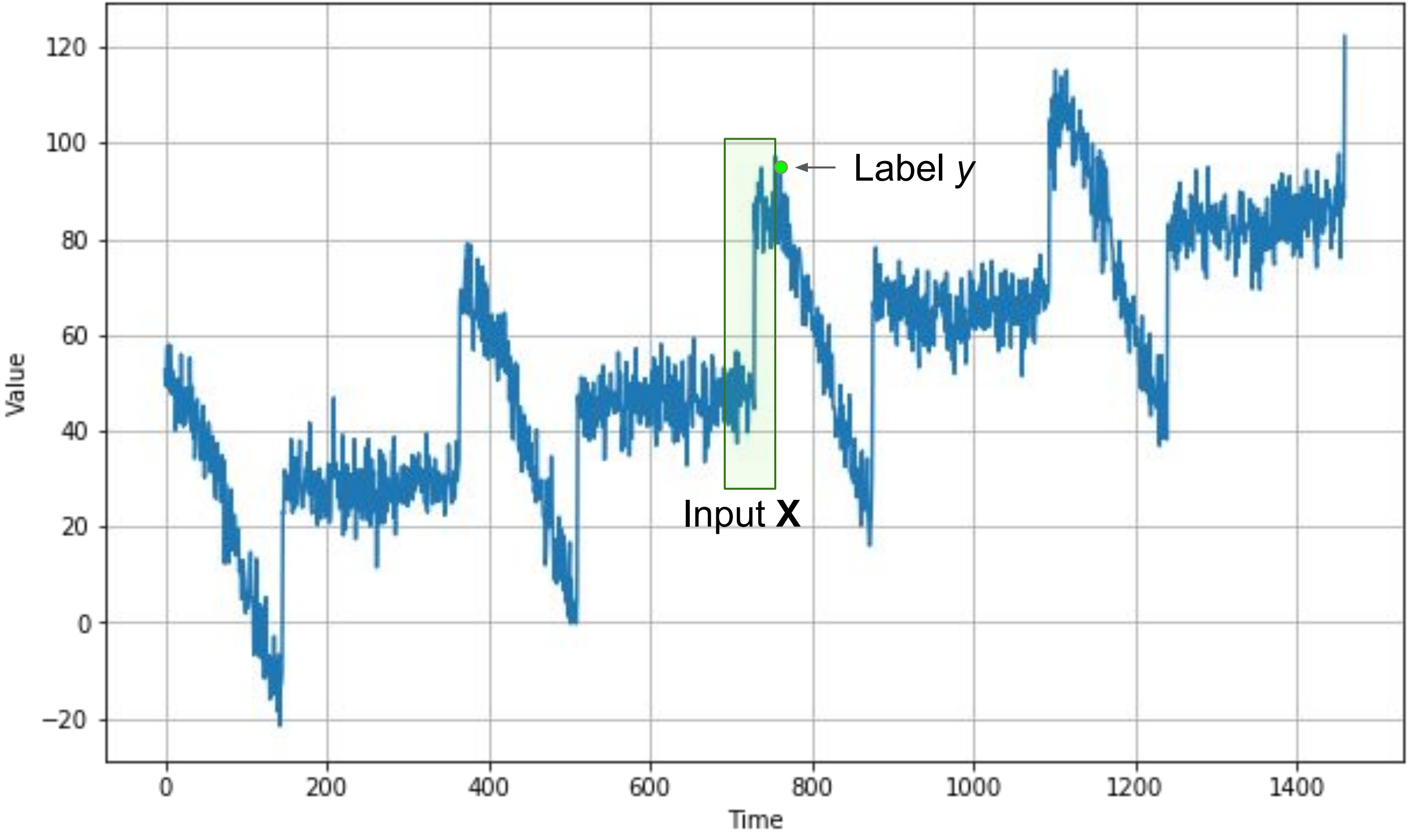
```
Layer weights [array([[ 0.01633573],
       [-0.02911791],
       [ 0.00845617],
       [-0.02175158],
       [ 0.04962169],
       [-0.03212642],
       [-0.02596855],
       [-0.00689476],
       [ 0.0616533 ],
       [-0.00668752],
       [-0.02735964],
       [ 0.0377918 ],
       [-0.02855931],
       [ 0.05299238],
       [-0.0121608 ],
       [ 0.00138755],
       [ 0.0905595 ],
       [ 0.19994621],
       [ 0.2556632 ],
       [ 0.41660047]], dtype=float32), array([0.01430958], dtype=float32)]
```

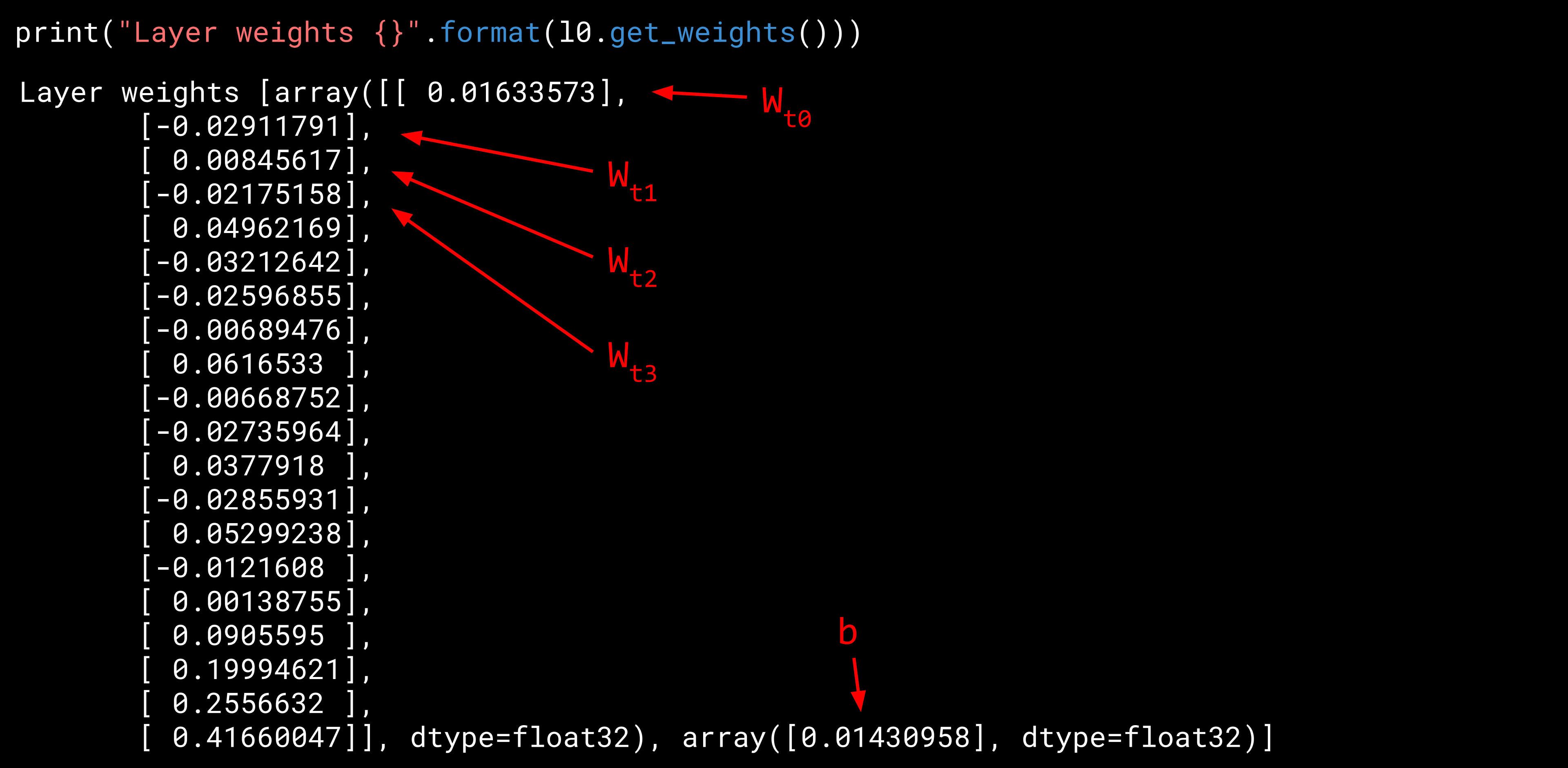
```
print("Layer weights {}".format(l0.get_weights()))
```



```
Layer weights [array([[ 0.01633573],
       [-0.02911791],
       [ 0.00845617],
       [-0.02175158],
       [ 0.04962169],
       [-0.03212642],
       [-0.02596855],
       [-0.00689476],
       [ 0.0616533 ],
       [-0.00668752],
       [-0.02735964],
       [ 0.0377918 ],
       [-0.02855931],
       [ 0.05299238],
       [-0.0121608 ],
       [ 0.00138755],
       [ 0.0905595 ],
       [ 0.19994621],
       [ 0.2556632 ],
       [ 0.41660047]], dtype=float32), array([0.01430958], dtype=float32)]
```

Machine Learning on Time Windows





```
print("Layer weights {}".format(l0.get_weights()))

Layer weights [array([[ 0.01633573],
       [-0.02911791],
       [ 0.00845617],
       [-0.02175158],
       [ 0.04962169],
       [-0.03212642],
       [-0.02596855],
       [-0.00689476],
       [ 0.0616533 ],
       [-0.00668752],
       [-0.02735964],
       [ 0.0377918 ],
       [-0.02855931],
       [ 0.05299238],
       [-0.0121608 ],
       [ 0.00138755],
       [ 0.0905595 ],
       [ 0.19994621],
       [ 0.2556632 ],
       [ 0.41660047]], dtype=float32), array([0.01430958], dtype=float32)]
```

```
print("Layer weights {}".format(l0.get_weights()))

Layer weights [array([[ 0.01633573],
        [-0.02911791],
        [ 0.00845617],
        [-0.02175158],
        [ 0.04962169],
        [-0.03212642],
        [-0.02596855],
        [-0.00689476],
        [ 0.0616533 ],
        [-0.00668752],
        [-0.02735964],
        [ 0.0377918 ],
        [-0.02855931],
        [ 0.05299238],
        [-0.0121608 ],
        [ 0.00138755],
        [ 0.0905595 ],
        [ 0.19994621],
        [ 0.2556632 ],
        [ 0.41660047]], dtype=float32), array([0.01430958], dtype=float32)]
```

W_{t0}

W_{t1}

W_{t2}

W_{t3}

$$Y = W_{t0}X_0 + W_{t1}X_1 + W_{t2}X_2 + \ldots.. + W_{t19}X_{19} + b$$

b

```
print(series[1:21])
model.predict(series[1:21][np.newaxis])
```

```
print(series[1:21])
model.predict(series[1:21][np.newaxis])
```

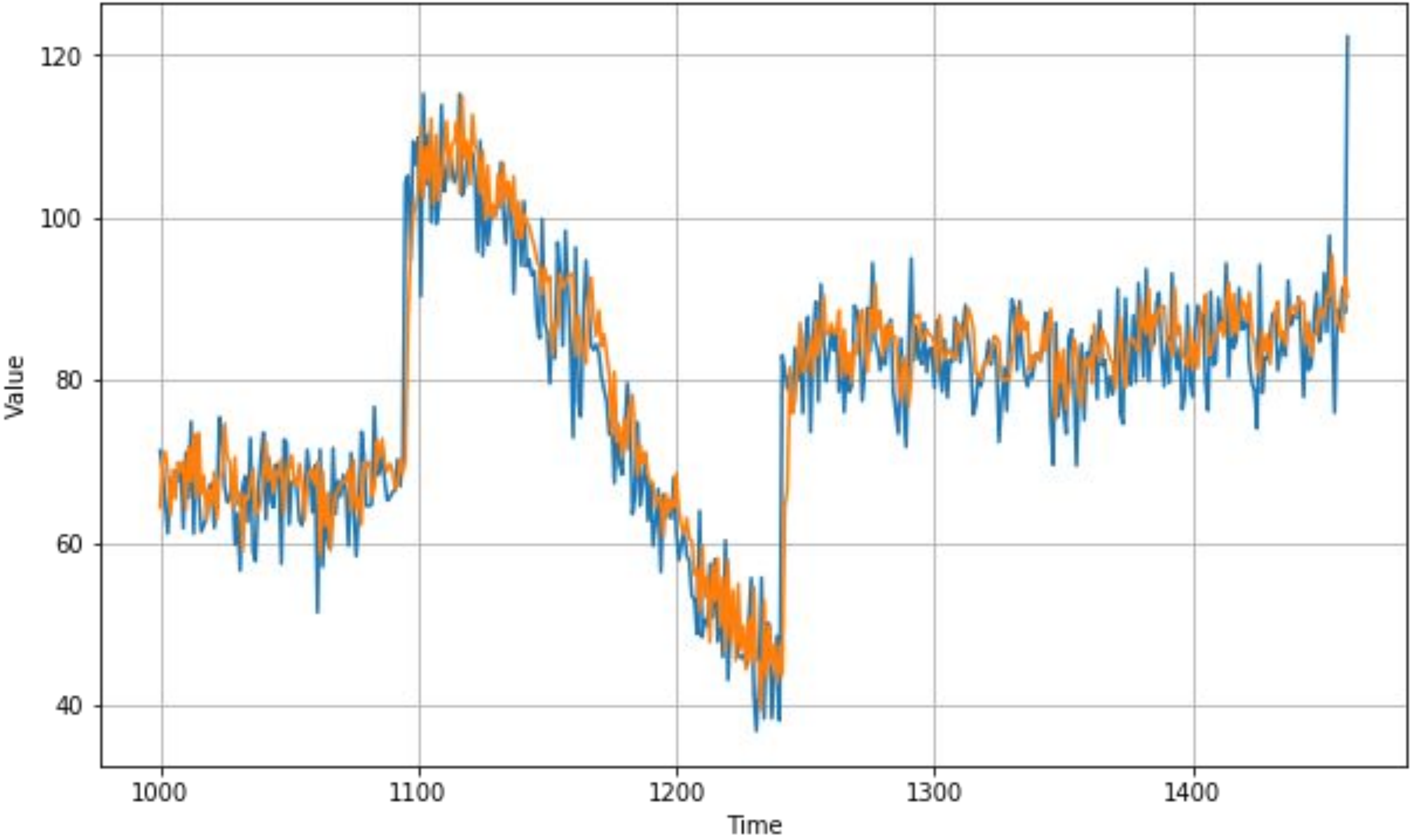
```
print(series[1:21])
model.predict(series[1:21][np.newaxis])



[49.35275   53.314735 57.711823 48.934444 48.931244 57.982895 53.897125
 47.67393   52.68371   47.591717 47.506374 50.959415 40.086178 40.919415
 46.612473 44.228207 50.720642 44.454983 41.76799   55.980938]

array([[49.08478]], dtype=float32)
```

```
forecast = []
for time in range(len(series) - window_size):
  forecast.append(model.predict(series[time:time + window_size][np.newaxis]))

forecast = forecast[split_time-window_size:]
results = np.array(forecast).squeeze()
```

```
forecast = []
for time in range(len(series) - window_size):
  forecast.append(model.predict(series[time:time + window_size][np.newaxis]))

forecast = forecast[split_time-window_size:]
results = np.array(forecast).squeeze()
```

```
forecast = []
for time in range(len(series) - window_size):
  forecast.append(model.predict(series[time:time + window_size][np.newaxis]))

forecast = forecast[split_time-window_size:]
results = np.array(forecast).squeeze()
```

```
forecast = []
for time in range(len(series) - window_size):
  forecast.append(model.predict(series[time:time + window_size][np.newaxis]))

forecast = forecast[split_time-window_size:]
results = np.array(forecast).squeeze()
```


120
100
80
60
40
Value
1000
1100
1200
1300
1400
Time

```
tf.keras.metrics.mae(x_valid, results).numpy()
```

```
    4.9526777
```

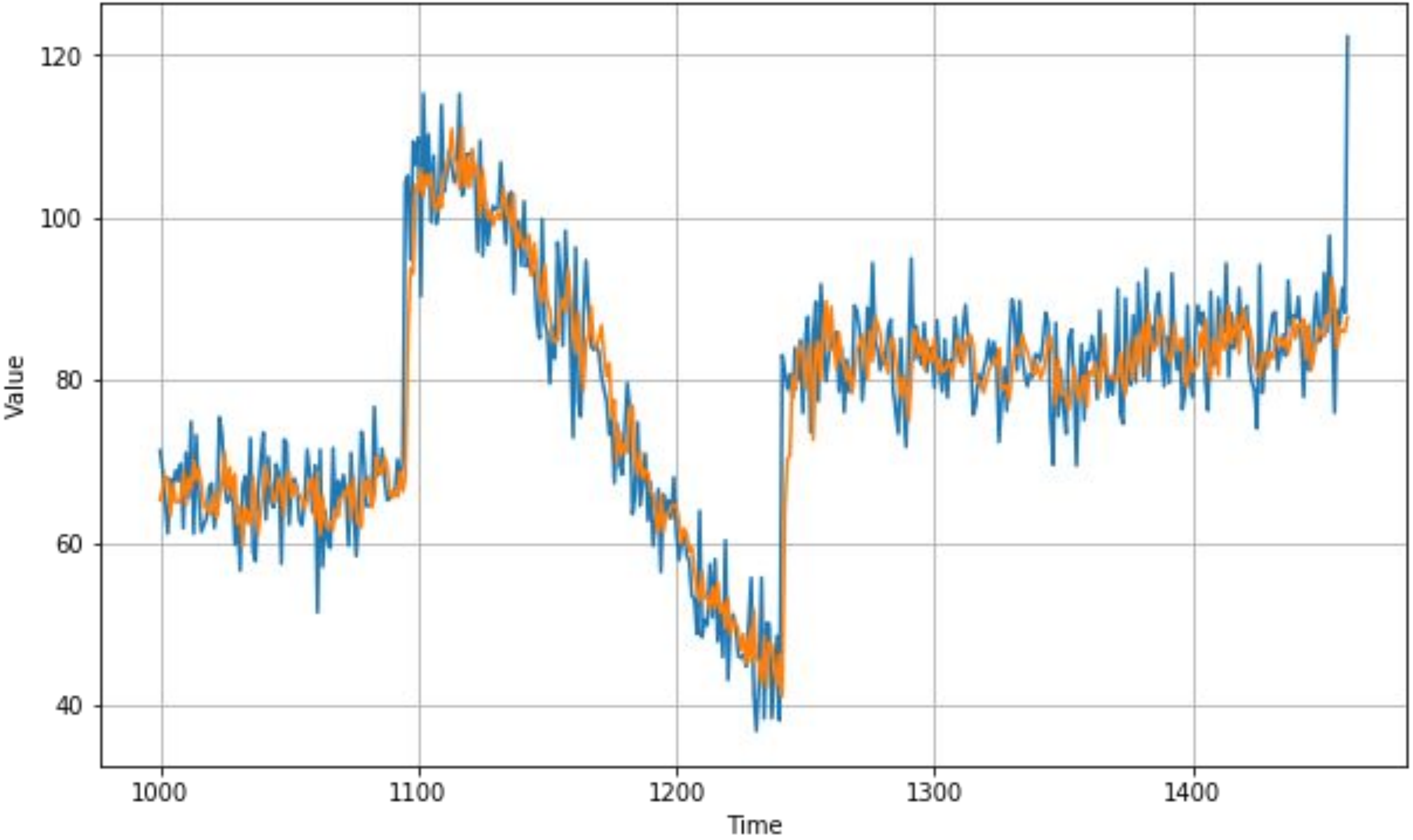
```
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size,)),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

model.compile(
     loss="mse",
     optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9)
)
model.fit(dataset, epochs=100, verbose=0)
```

```
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size,)),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

model.compile(
    loss="mse",
    optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9)
)
model.fit(dataset, epochs=100, verbose=0)
```

```
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size,)),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

model.compile(
     loss="mse",
     optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9)
)
model.fit(dataset, epochs=100, verbose=0)
```

```
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size,)),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

model.compile(
     loss="mse",
     optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9)
)
model.fit(dataset, epochs=100, verbose=0)
```

```
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size,)),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

model.compile(
    loss="mse",
    optimizer=tf.keras.optimizers.SGD(learning_rate=1e-6, momentum=0.9)
)
model.fit(dataset, epochs=100, verbose=0)
```


120
100
80
60
40
Value
1000
1100
1200
1300
1400
Time

```
tf.keras.metrics.mae(x_valid, results).numpy()
```

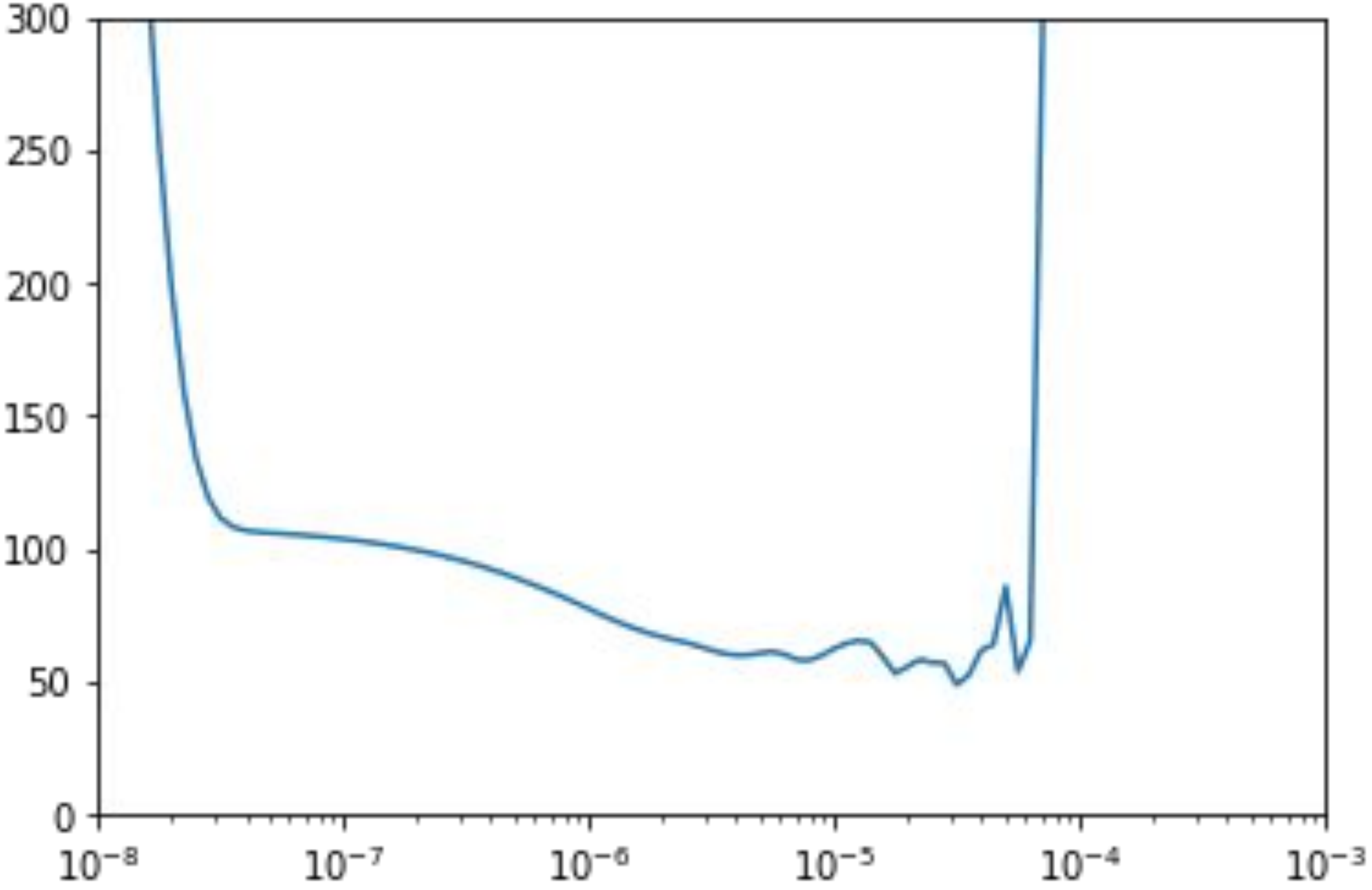
```
4.9833784
```

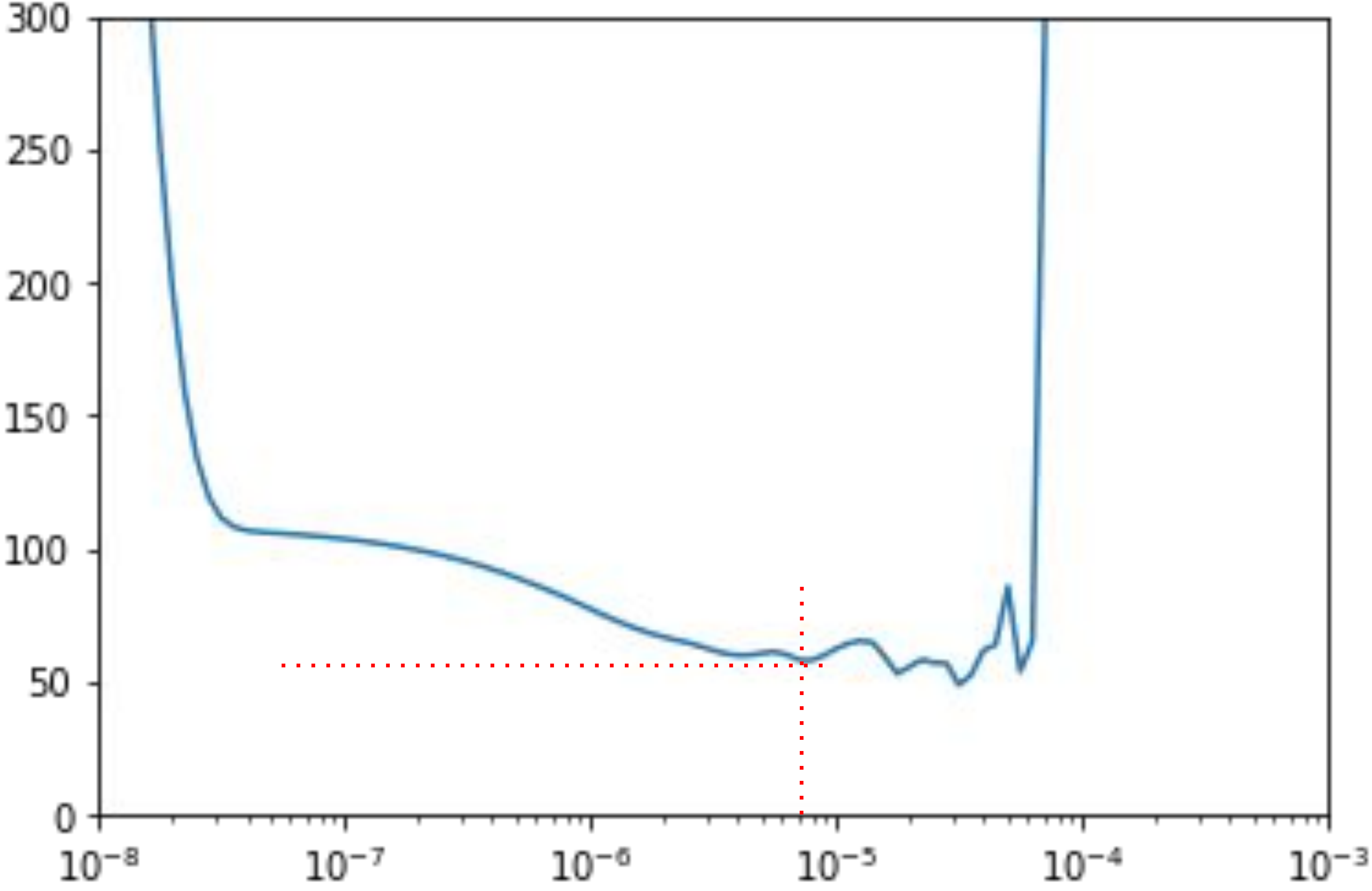
```
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size,)),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch / 20))

optimizer = tf.keras.optimizers.SGD(momentum=0.9)

model.compile(loss="mse", optimizer=optimizer)

history = model.fit(dataset, epochs=100, callbacks=[lr_schedule])
```

```python
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size,)),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch / 20))

optimizer = tf.keras.optimizers.SGD(momentum=0.9)

model.compile(loss="mse", optimizer=optimizer)

history = model.fit(dataset, epochs=100, callbacks=[lr_schedule])
```

```
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size,)),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

lr_schedule = tf.keras.callbacks.LearningRateScheduler(
    lambda epoch: 1e-8 * 10**(epoch / 20))

optimizer = tf.keras.optimizers.SGD(momentum=0.9)

model.compile(loss="mse", optimizer=optimizer)

history = model.fit(dataset, epochs=100, callbacks=[lr_schedule])
```

```python
lrs = 1e-8 * (10 ** (np.arange(100) / 20))
plt.semilogx(lrs, history.history["loss"])
plt.axis([1e-8, 1e-3, 0, 300])
```

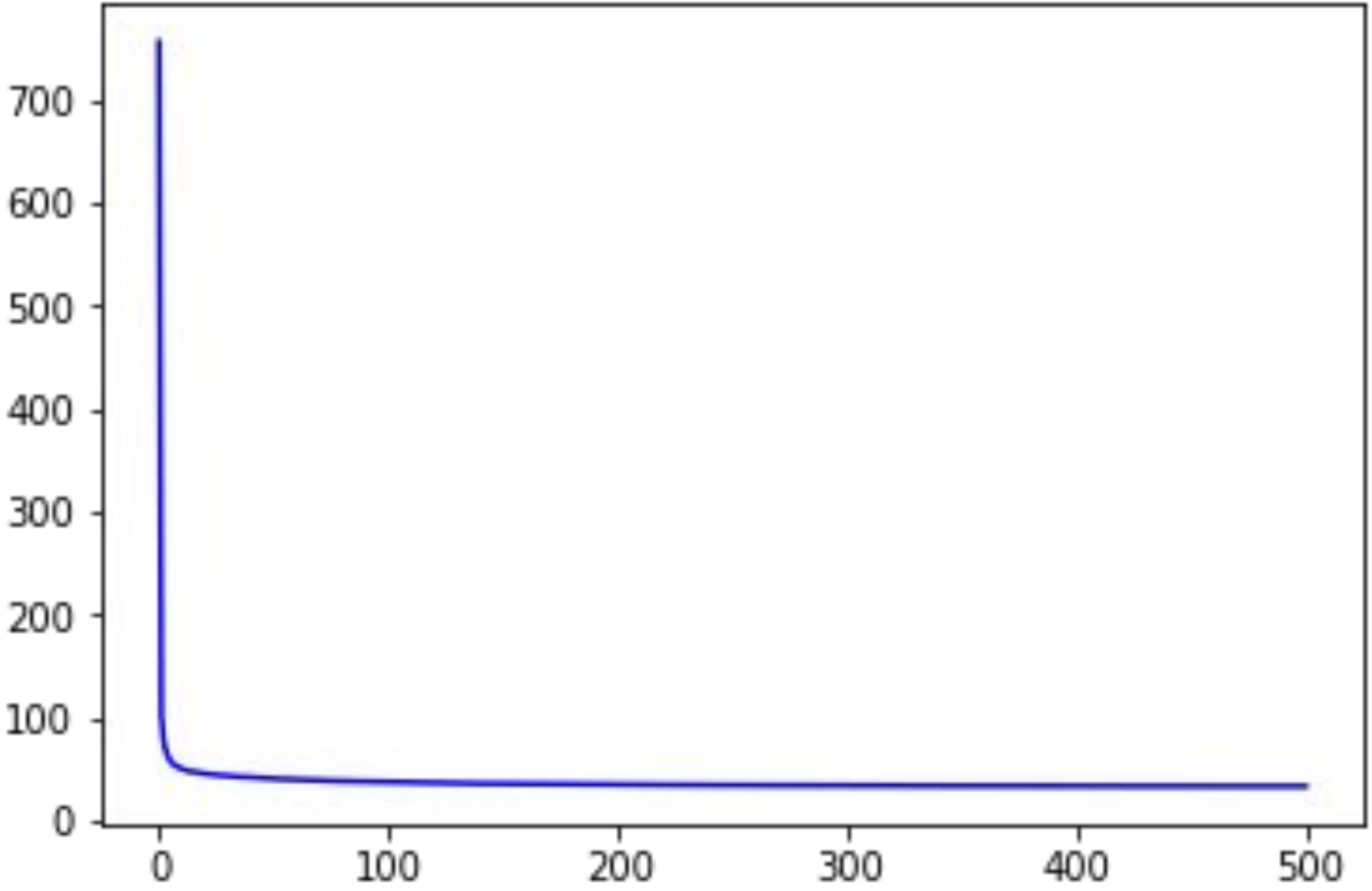

300
250
200
150
100
50
0
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}


300
250
200
150
100
50
0
10^{-8}
10^{-7}
10^{-6}
10^{-5}
10^{-4}
10^{-3}

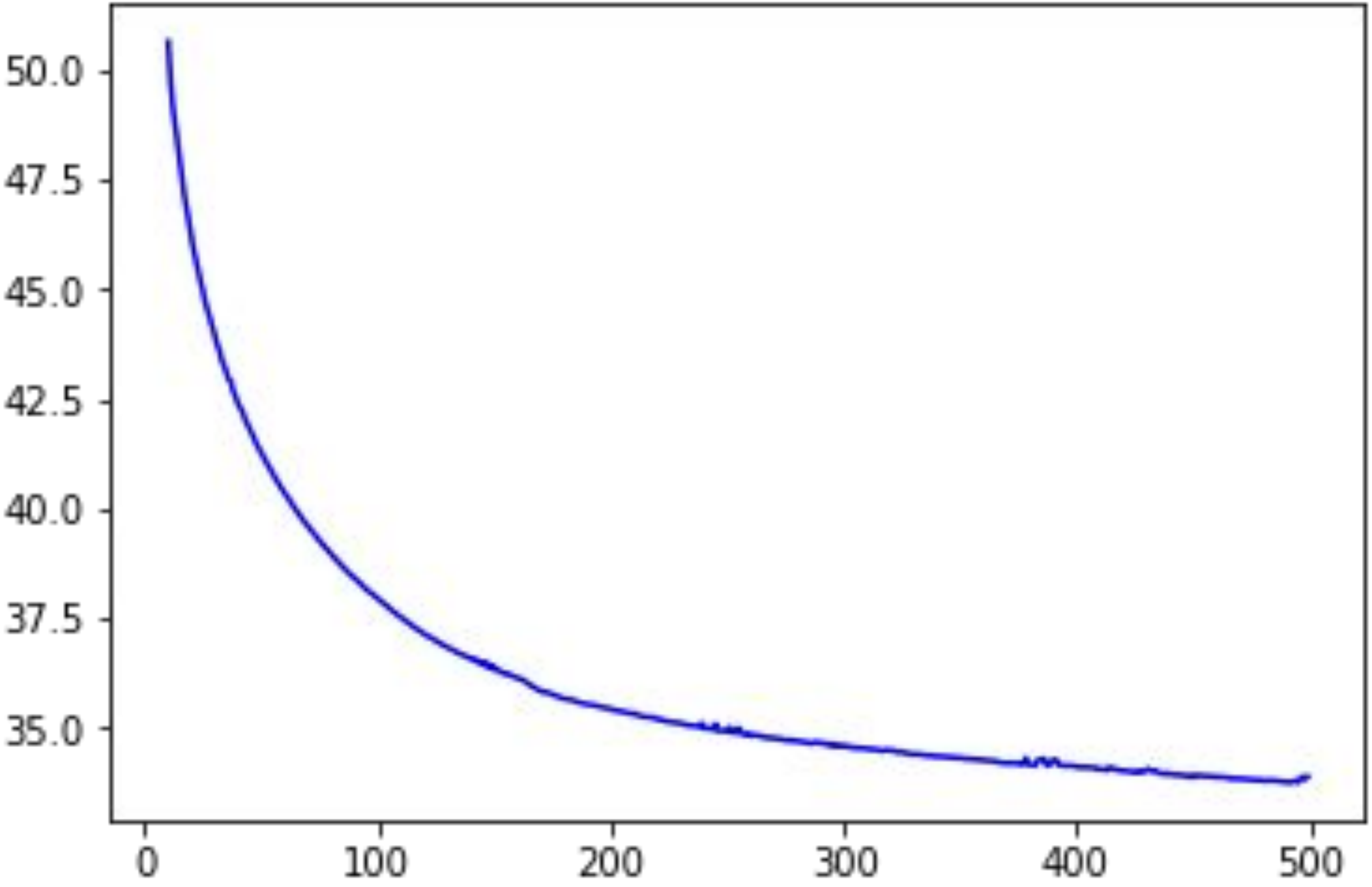
```python
window_size = 30
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size,)),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

optimizer = tf.keras.optimizers.SGD(learning_rate=7e-6, momentum=0.9)

model.compile(loss="mse", optimizer=optimizer)

history = model.fit(dataset, epochs=500)
```

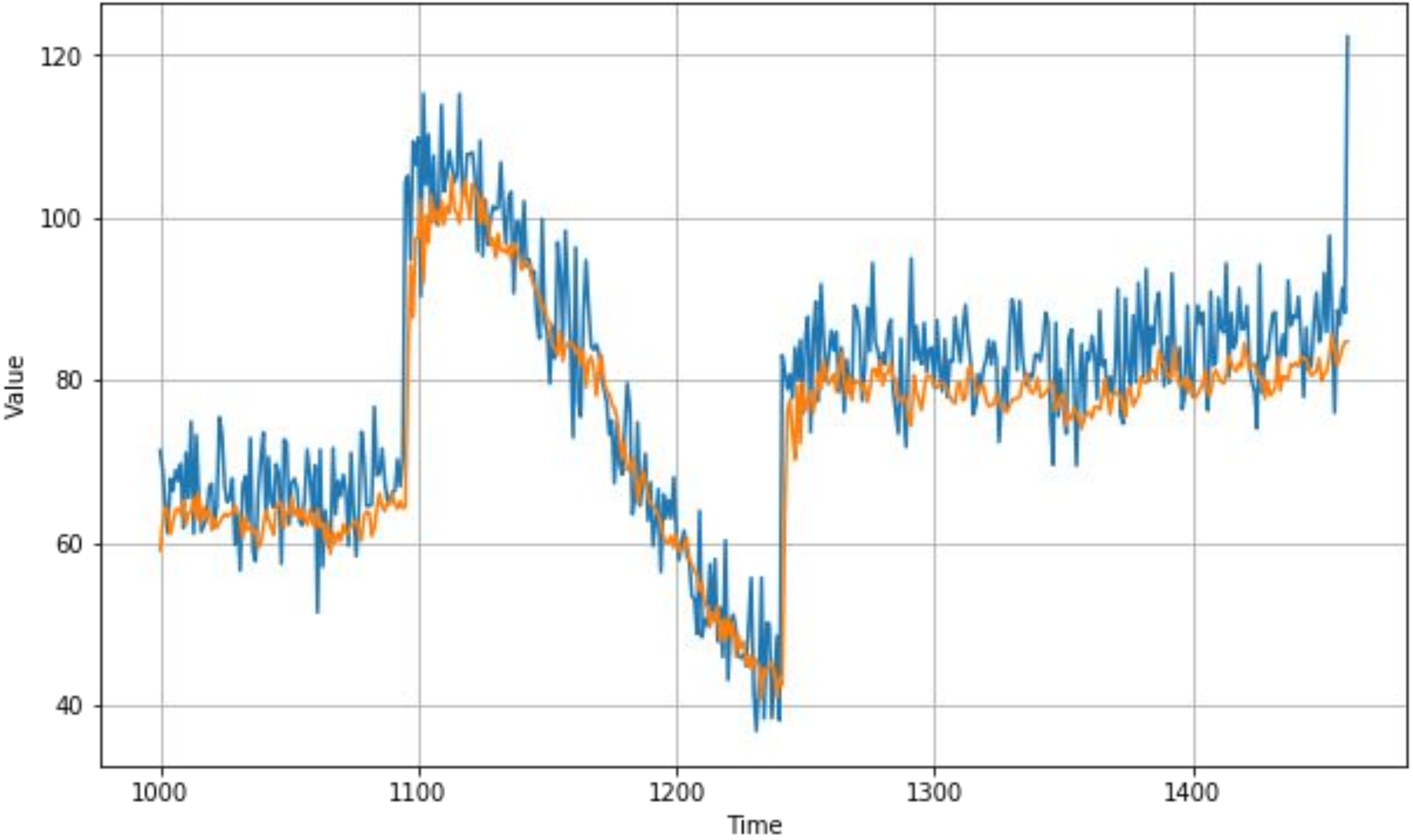
```
window_size = 30
dataset = windowed_dataset(x_train, window_size, batch_size, shuffle_buffer_size)

model = tf.keras.models.Sequential([
    tf.keras.Input(shape=(window_size,)),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(10, activation="relu"),
    tf.keras.layers.Dense(1)
])

optimizer = tf.keras.optimizers.SGD(learning_rate=7e-6, momentum=0.9)

model.compile(loss="mse", optimizer=optimizer)

history = model.fit(dataset, epochs=500)
```

```python
loss = history.history['loss']
epochs = range(len(loss))
plt.plot(epochs, loss, 'b', label='Training Loss')
plt.show()
```


700
600
500
400
300
200
100
0
0
100
200
300
400
500

```python
loss = history.history['loss']
epochs = range(10, len(loss))
plot_loss = loss[10:]
plt.plot(epochs, loss, 'b', label='Training Loss')
plt.show()
```


50.0
47.5
45.0
42.5
40.0
37.5
35.0
0
100
200
300
400
500



120
100
80
60
40
Value
1000
1100
1200
1300
1400
Time

```
tf.keras.metrics.mae(x_valid, results).numpy()
```

```
4.4847784
```