**WALC 2024**
**Applied AI**

# Introduction to RNN - Time Series

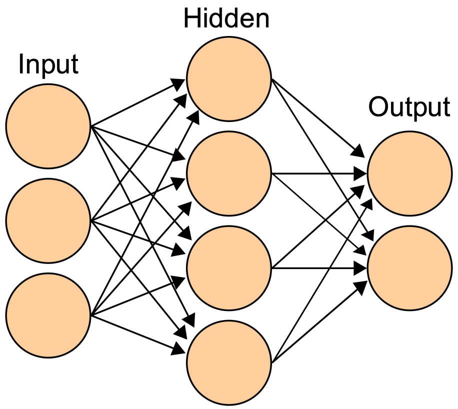
Prof. Marcelo J. Rovai
rovai@unifei.edu.br

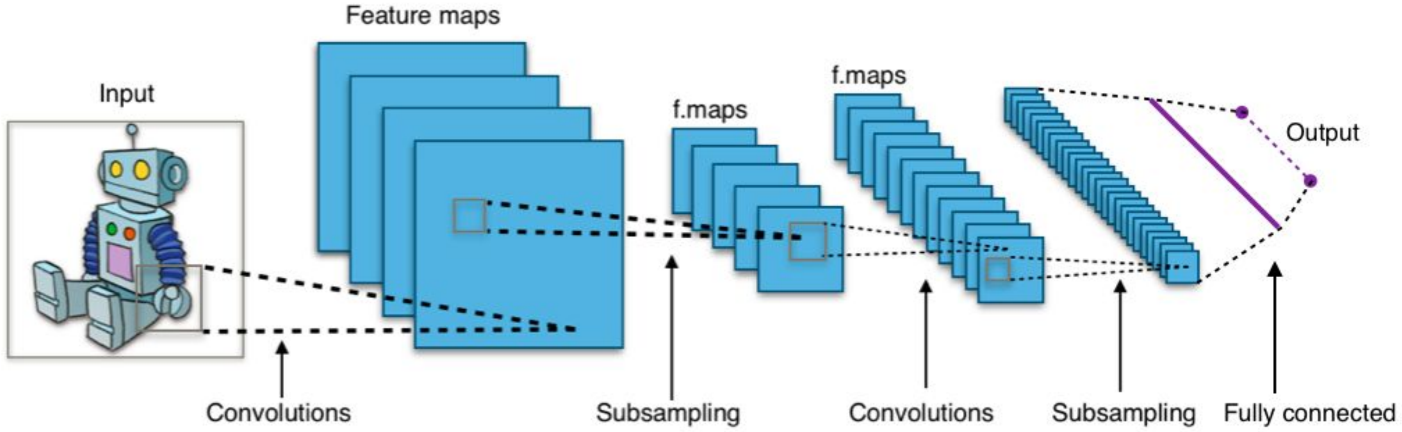
UNIFEI - Federal University of Itajuba, Brazil
TinyML4D Academic Network Co-Chair

TINYML4D

# **Deep Learning** models (or artificial neural networks)

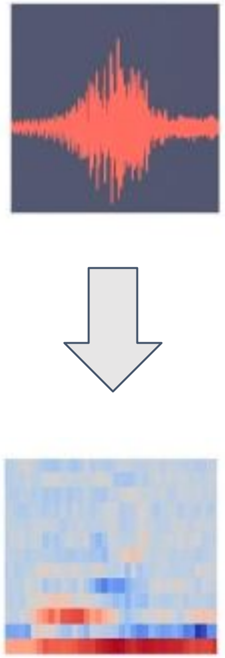
**Fully Connected Neural Networks (FCNNs)**: Networks where **each neuron in one layer is connected to every neuron in the following layer**, useful for complex pattern recognition across diverse datasets.



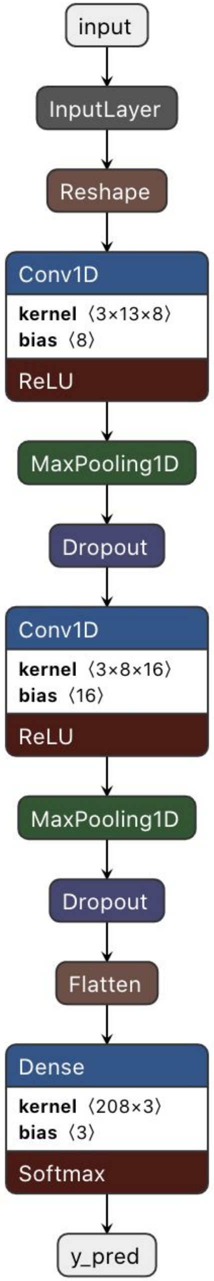




**Convolutional Neural Networks (CNNs)**: Specialized for **grid-like data such as images**, using convolutional layers to detect and learn spatial hierarchies of features.

# CNN for Audio:



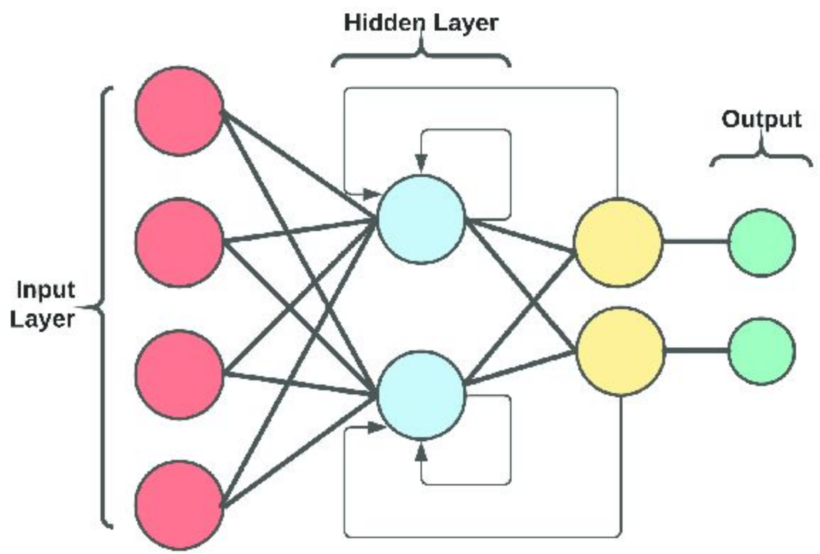
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape (Reshape)            (None, 50, 13)            0
_________________________________________________________________
conv1d (Conv1D)              (None, 50, 8)             320
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 25, 8)             0
_________________________________________________________________
dropout (Dropout)            (None, 25, 8)             0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 25, 16)            400
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 13, 16)            0
_________________________________________________________________
dropout_1 (Dropout)          (None, 13, 16)            0
_________________________________________________________________
flatten (Flatten)            (None, 208)               0
_________________________________________________________________
y_pred (Dense)               (None, 3)                 627
=================================================================
Total params: 1,347
Trainable params: 1,347
Non-trainable params: 0
_________________________________________________________________
```



# RNN (LSTM/GRU)

# **RNN** (Recurrent Neural Network) architecture:

*Think of RNN like a person reading a book - they remember what happened in previous pages to understand the current one. It's designed to work with sequences, like numbers and words in a sentence or notes in a melody.*



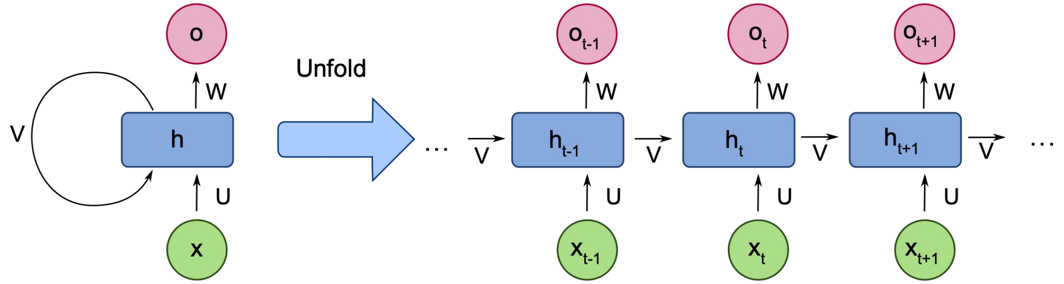

**Key components:**

1. **Input Layer**: Receives data one element at a time (like words in a sentence)
2. **Hidden Layer**: Contains the "memory cells" that remember important past information
3. **Output Layer**: Produces predictions based on current input and stored memory

The unique feature is the "**feedback loop (v)**" - each step's output influences the next step's processing. This helps the network understand context and patterns over time.

**Real-world applications include**:

- Text prediction (like smartphone keyboards)
- Language translation
- Speech recognition
- Music generation
- Weather forecasting



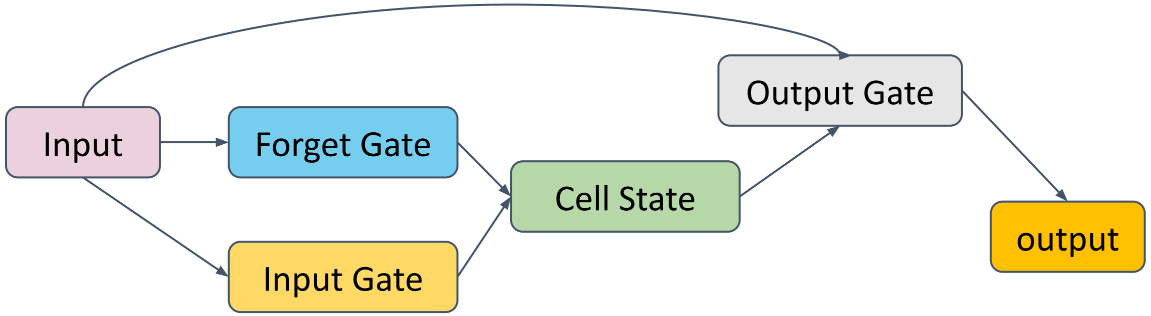
# LSTM (Long Short-Term Memory):

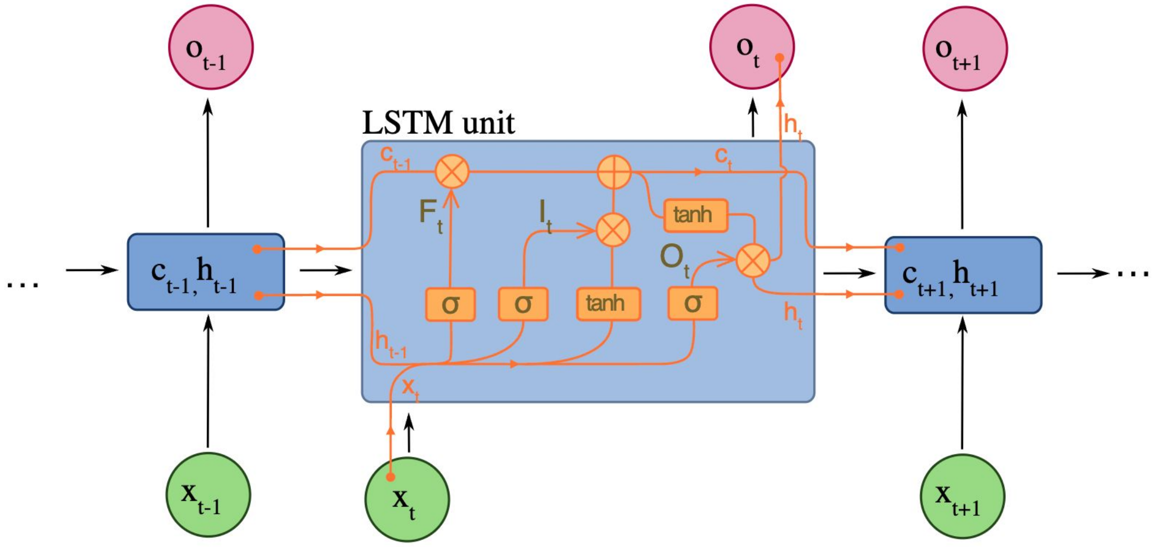
LSTM is an advanced version of RNN with three special "gates":

1. **Forget Gate**: Decides what information to discard
2. **Input Gate**: Decides what new information to store
3. **Output Gate**: Determines what parts of the cell state to output

Think of it like a smart note-taking system:
- Forget Gate: Erasing outdated notes
- Input Gate: Writing new important information
- Output Gate: Choosing which notes to share

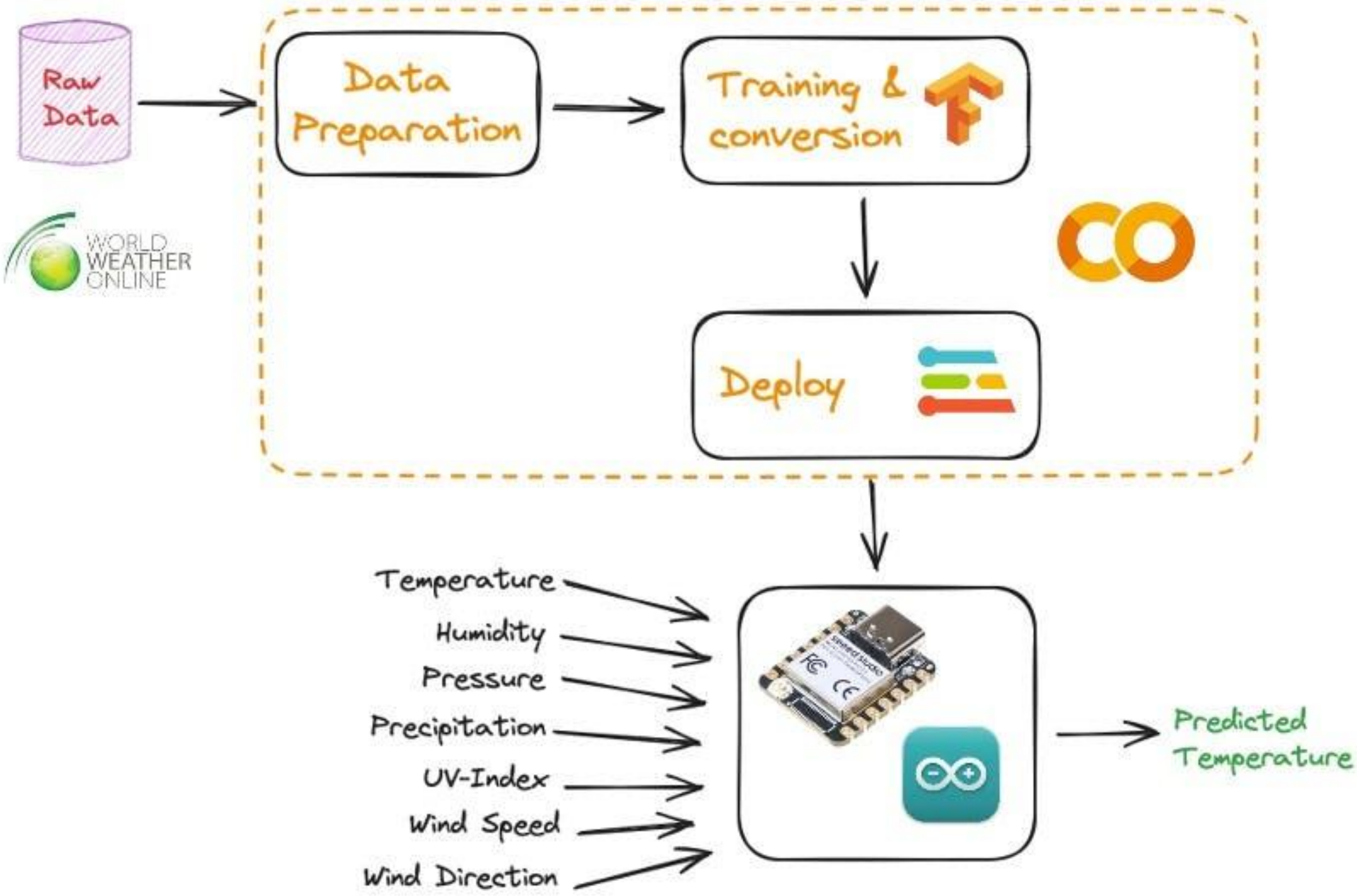
This architecture helps solve the "vanishing gradient" problem of basic RNNs, allowing LSTMs to remember information for much longer periods. They're particularly good at:
- Long text generation
- Time series prediction [Predicción Tráfico de Bicicletas usando DL/LSTM]
- Speech recognition
- Music composition

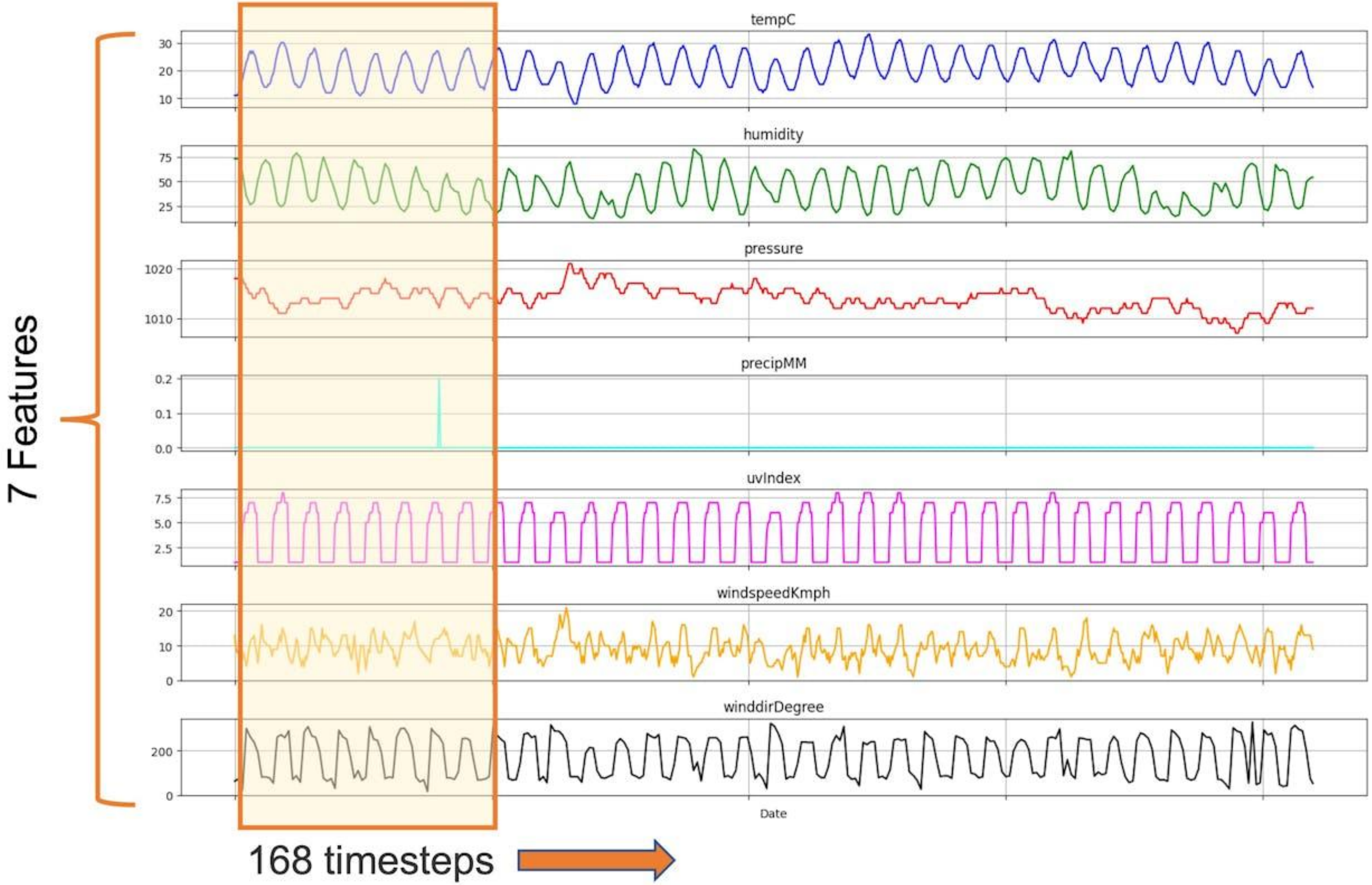


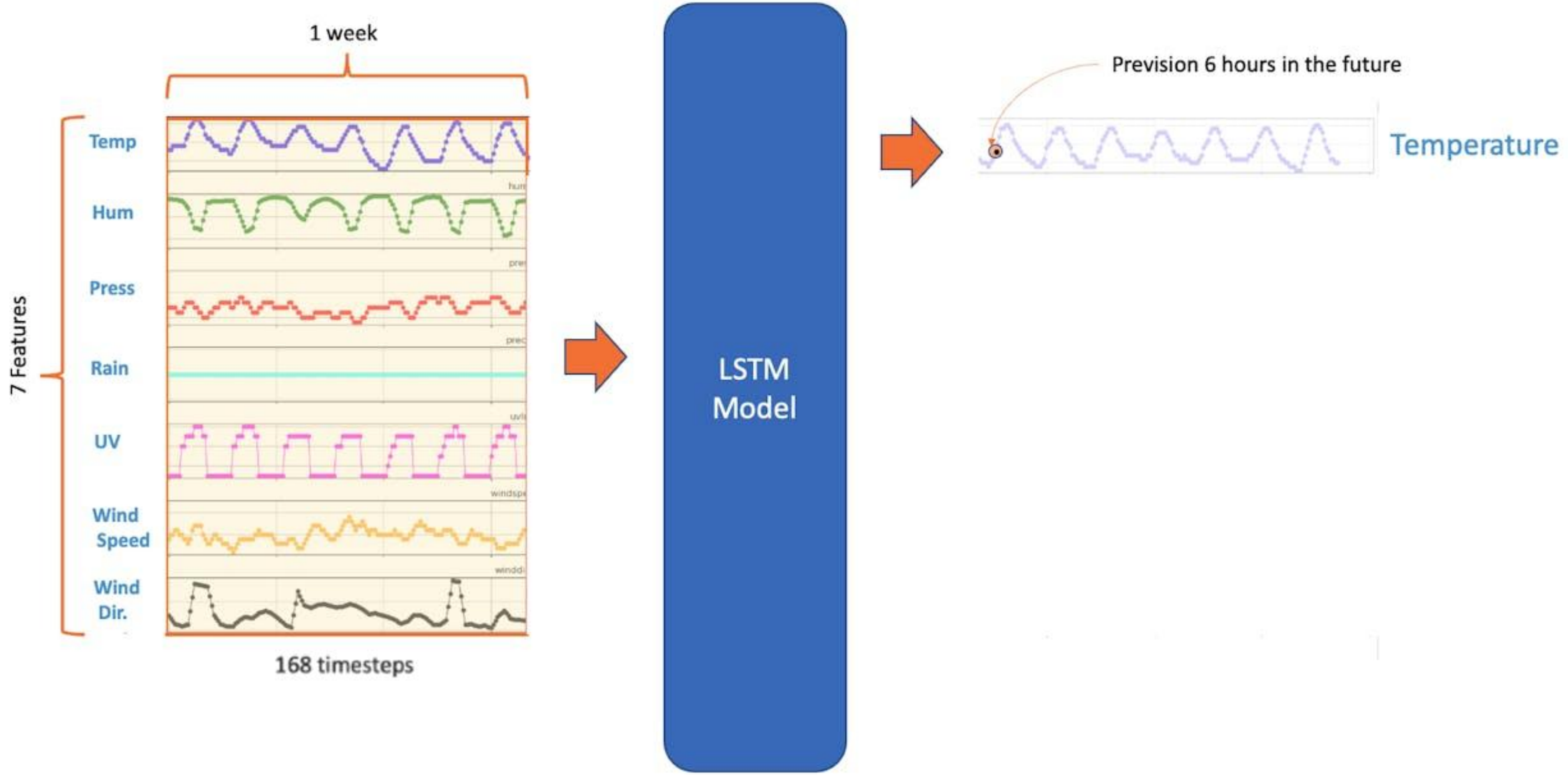



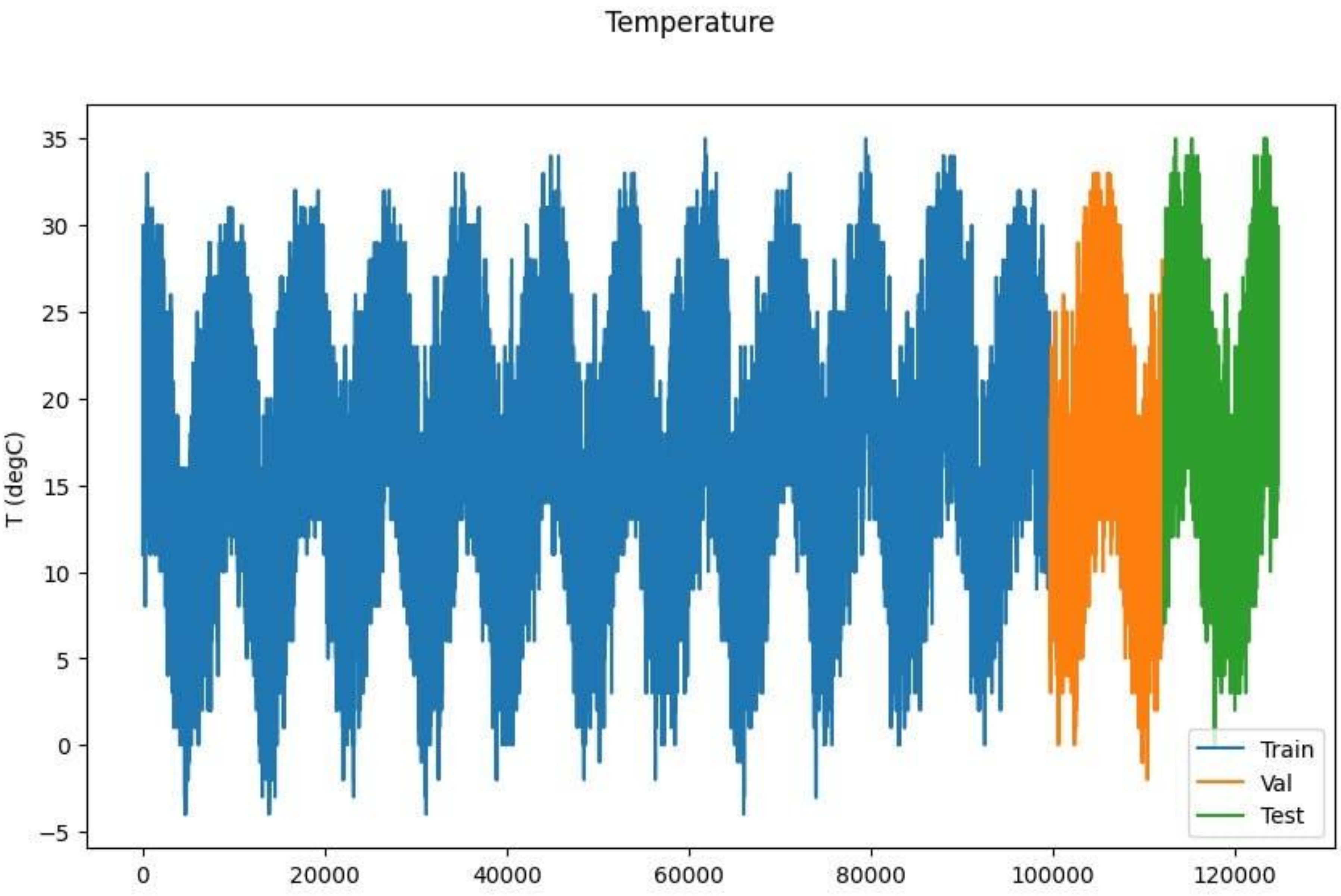
## Temperature Prediction using an LSTM model

*The trained LSTM model will be converted with TFLite-Micro and* ***Edge Impulse Python SDK*** *and deployed on an XIAO ESP32S3.*


Raw Data
WORLD WEATHER ONLINE
Data Preparation
Training & conversion
Deploy
Temperature
Humidity
Pressure
Precipitation
UV-Index
Wind Speed
Wind Direction
Predicted Temperature


7 Features
tempC
30
20
10
humidity
75
50
25
pressure
1020
1010
precipMM
0.2
0.1
0.0
uvIndex
7.5
5.0
2.5
windspeedKmph
20
10
0
winddirDegree
200
0
Date
168 timesteps

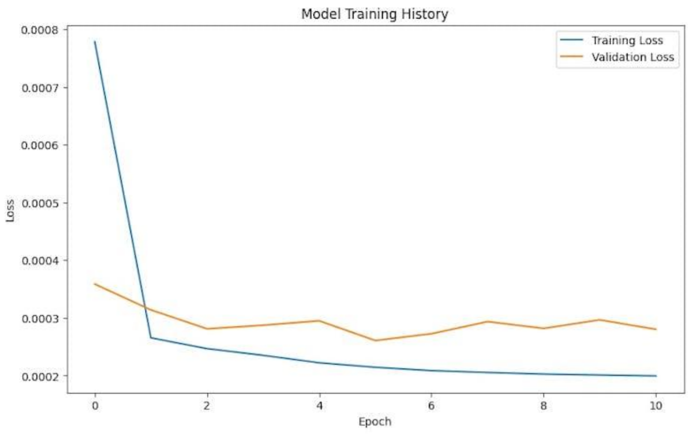

1 week
7 Features
Temp
Hum
Press
Rain
UV
Wind Speed
Wind Dir.
168 timesteps
LSTM Model
Prevision 6 hours in the future
Temperature


Temperature
T (degC)
35
30
25
20
15
10
5
0
−5
0
20000
40000
60000
80000
100000
120000
Train
Val
Test

```
model = Sequential([
    LSTM(128,
         input_shape=(n_steps, X_train.shape[2])),
    Dense(1)
])
```

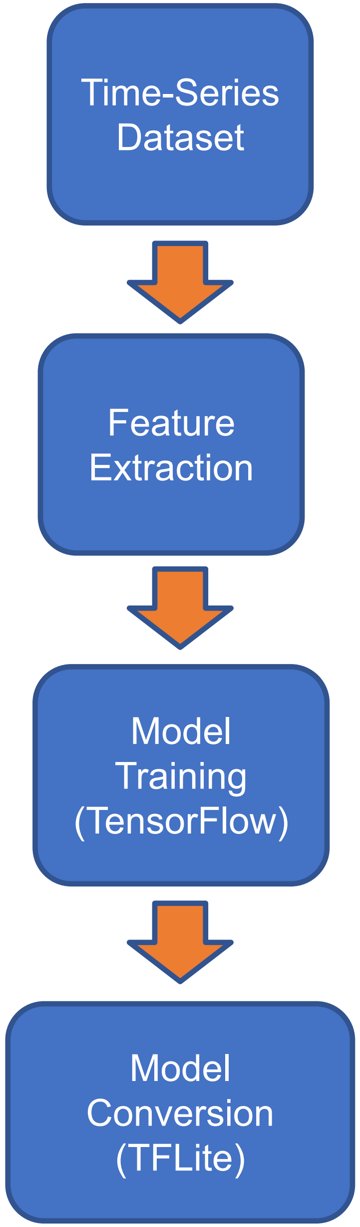
```
model.compile(optimizer='adam', loss='mse')
```

```
history = model.fit(
    X_train, y_train,
    validation_data=(X_val, y_val),
    epochs=20,
    batch_size=32,
    callbacks=[early_stopping]
)
```

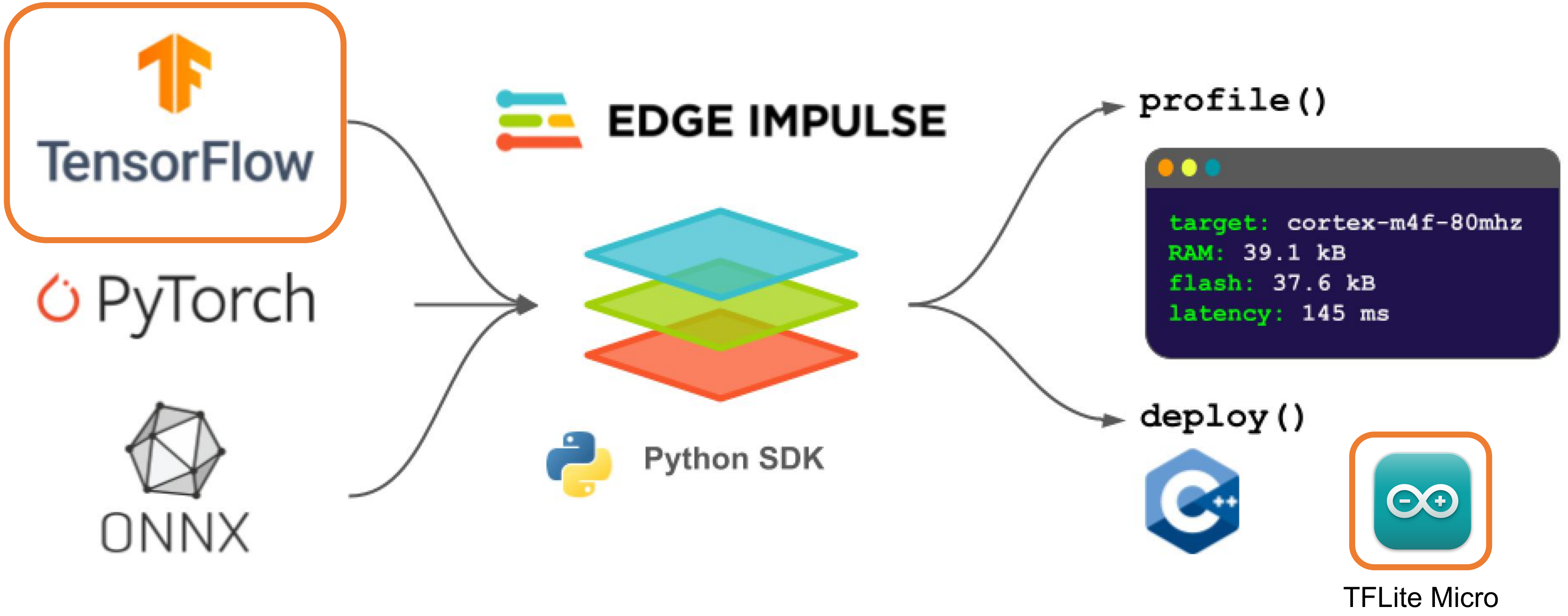



```
prediccion = model.predict(X_test)
```

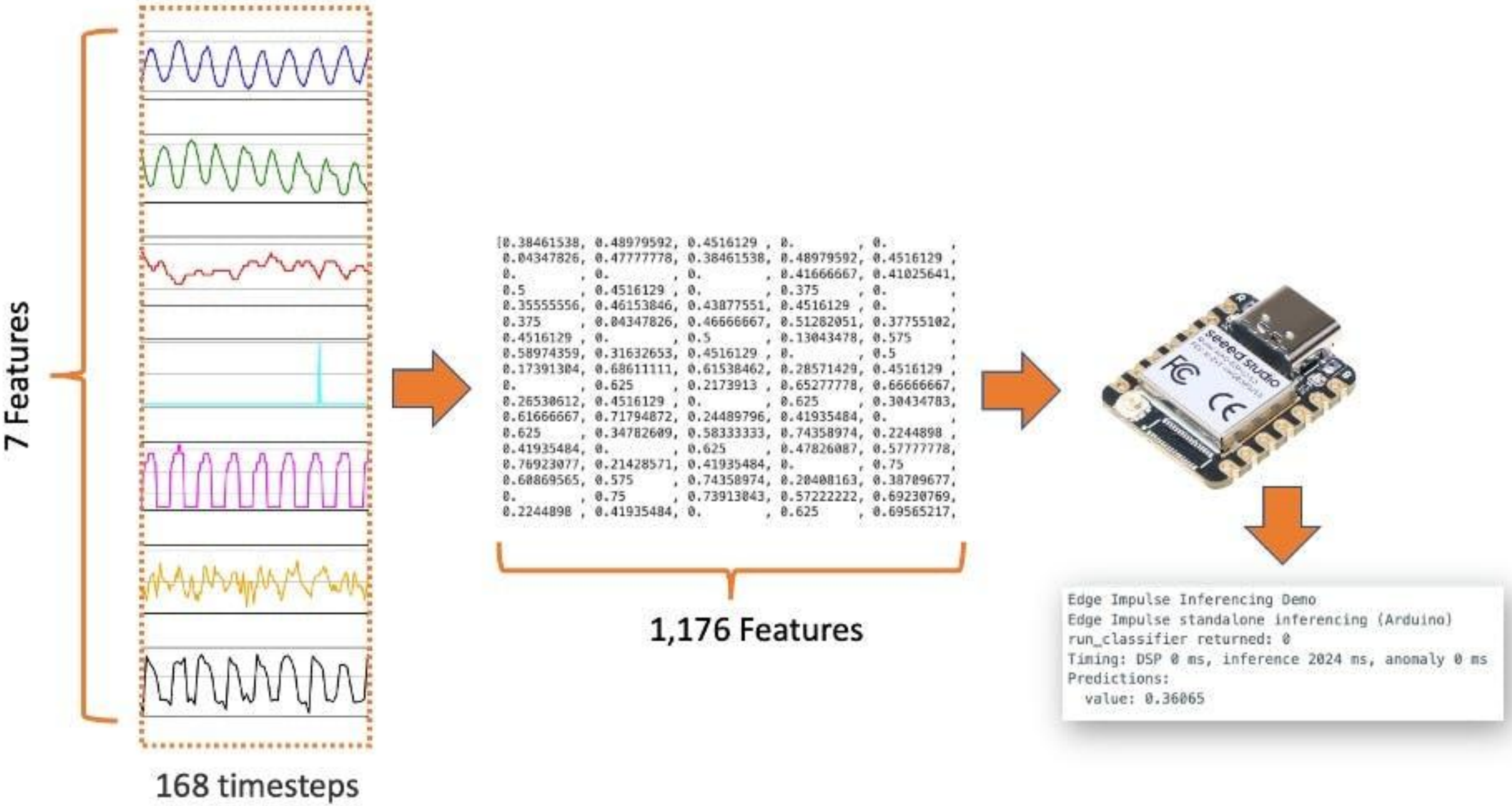
```
converter = tf.lite.TFLiteConverter.from_saved_model(MODEL_DIR)
tflite_model = converter.convert()

# Save the converted model to file
tflite_model_file = 'converted_model.tflite'
with open(tflite_model_file, 'wb') as f:
    f.write(tflite_model)
```

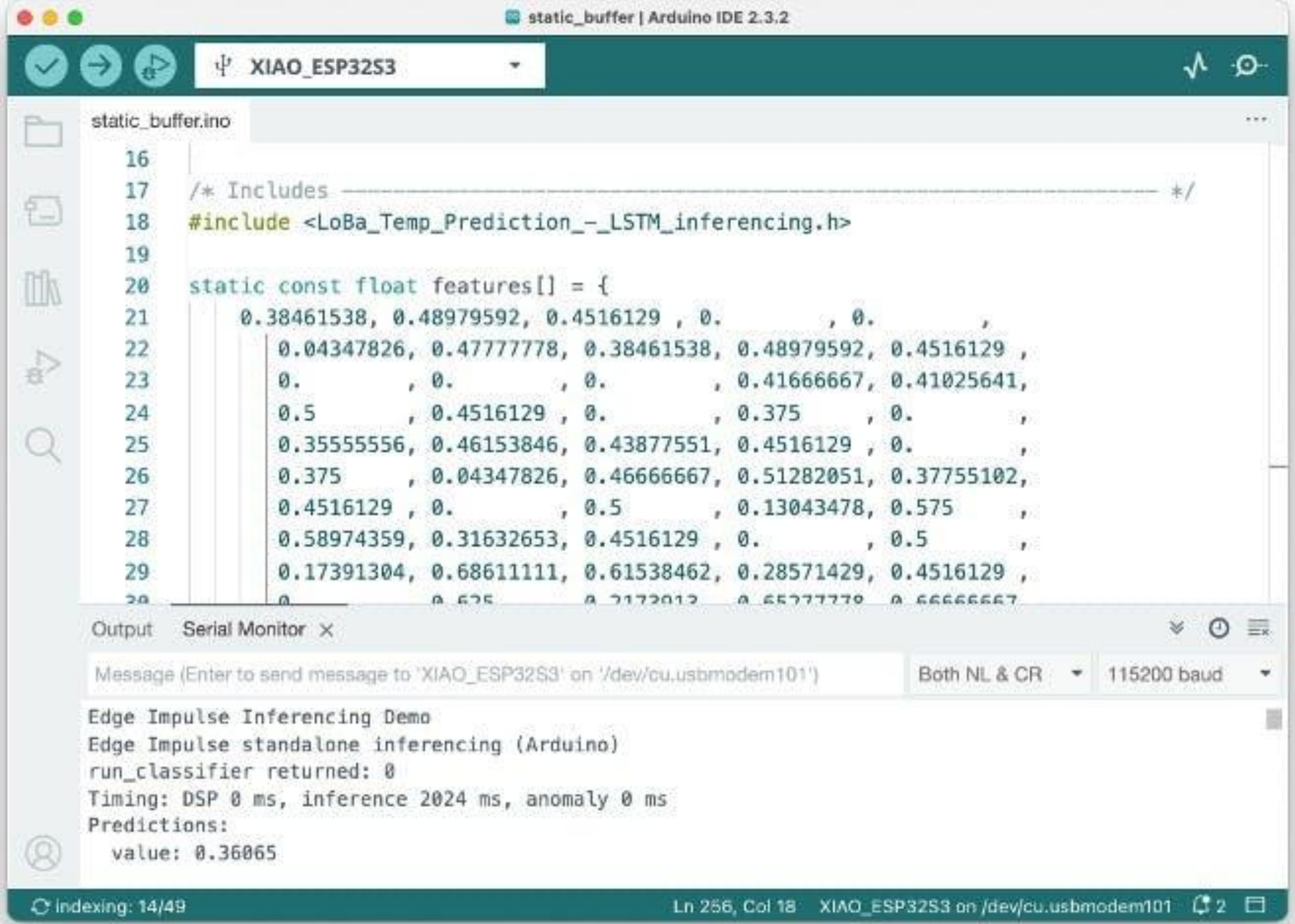


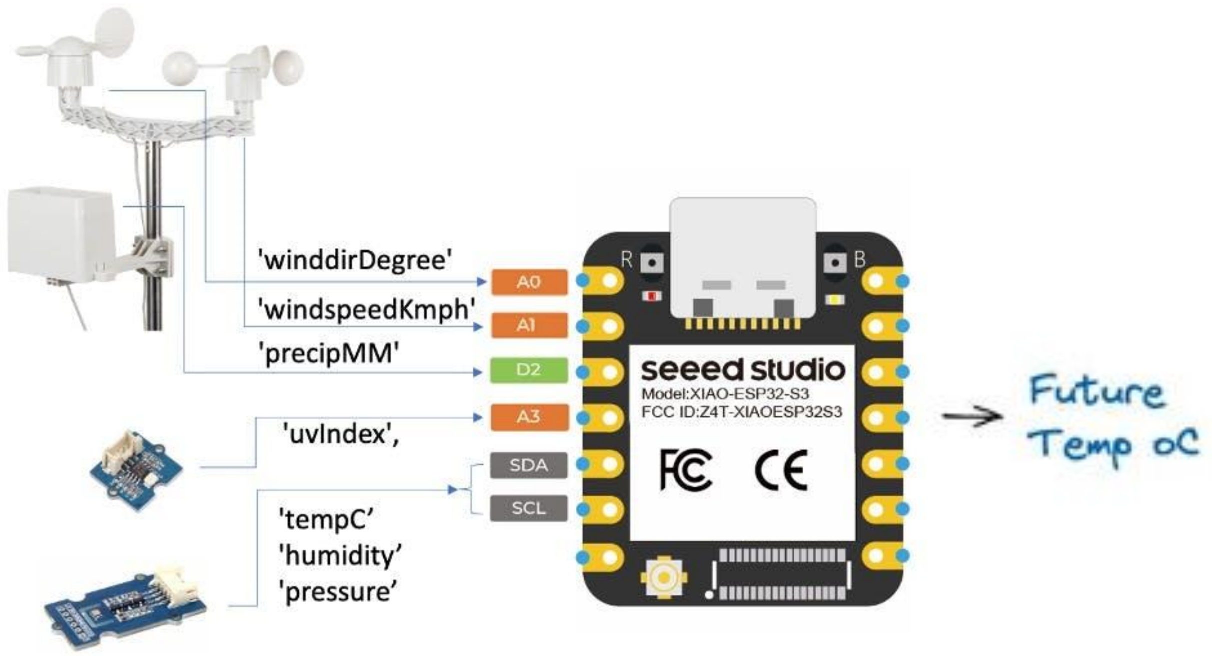
# Edge Impulse Python SDK

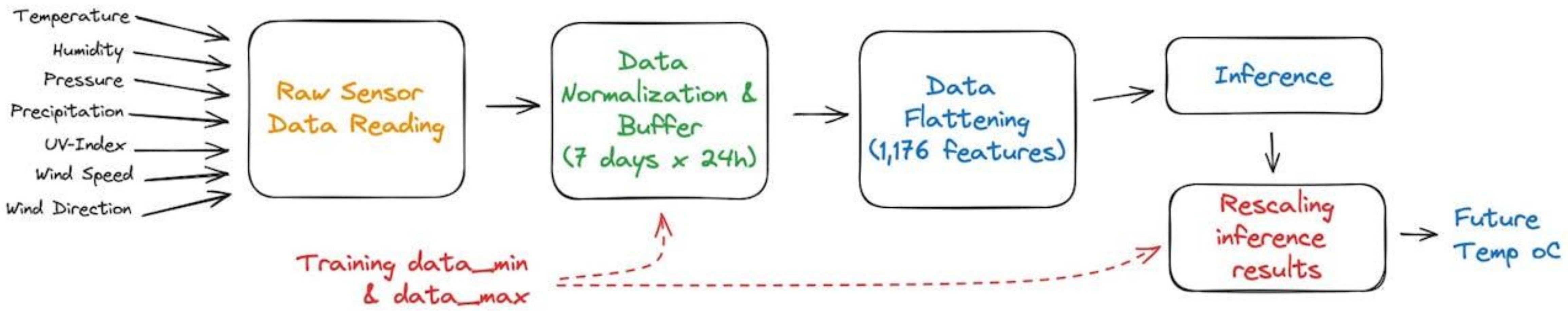



converted_model.tflite → EI Python SDK → lstm_float32_model.zip


7 Features
168 timesteps
[0.38461538, 0.48979592, 0.4516129 , 0. , 0. ,
0.04347826, 0.47777778, 0.38461538, 0.48979592, 0.4516129 ,
0. , 0. , 0. , 0.41666667, 0.41025641,
0.5 , 0.4516129 , 0. , 0.375 , 0. ,
0.35555556, 0.46153846, 0.43877551, 0.4516129 , 0. ,
0.375 , 0.04347826, 0.46666667, 0.51282051, 0.37755102,
0.4516129 , 0. , 0.5 , 0.13043478, 0.575 ,
0.58974359, 0.31632653, 0.4516129 , 0. , 0.5 ,
0.17391304, 0.68611111, 0.61538462, 0.28571429, 0.4516129 ,
0. , 0.625 , 0.2173913 , 0.65277778, 0.66666667,
0.26530612, 0.4516129 , 0. , 0.625 , 0.30434783,
0.61666667, 0.71794872, 0.24489796, 0.41935484, 0. ,
0.625 , 0.34782609, 0.58333333, 0.74358974, 0.2244898 ,
0.41935484, 0. , 0.625 , 0.47826087, 0.57777778,
0.76923077, 0.21428571, 0.41935484, 0. , 0.75 ,
0.60869565, 0.575 , 0.74358974, 0.20408163, 0.38709677,
0. , 0.75 , 0.73913043, 0.57222222, 0.69230769,
0.2244898 , 0.41935484, 0. , 0.625 , 0.69565217,
1,176 Features
Edge Impulse Inferencing Demo
Edge Impulse standalone inferencing (Arduino)
run_classifier returned: 0
Timing: DSP 0 ms, inference 2024 ms, anomaly 0 ms
Predictions:
value: 0.36065


static_buffer | Arduino IDE 2.3.2
XIAO_ESP32S3
static_buffer.ino
/* Includes ---------------------------------------------------------------- */
#include <LoBa_Temp_Prediction_-_LSTM_inferencing.h>
static const float features[] = {
0.38461538, 0.48979592, 0.4516129 , 0.        , 0.        ,
0.04347826, 0.47777778, 0.38461538, 0.48979592, 0.4516129 ,
0.        , 0.        , 0.        , 0.41666667, 0.41025641,
0.5       , 0.4516129 , 0.        , 0.375     , 0.        ,
0.35555556, 0.46153846, 0.43877551, 0.4516129 , 0.        ,
0.375     , 0.04347826, 0.46666667, 0.51282051, 0.37755102,
0.4516129 , 0.        , 0.5       , 0.13043478, 0.575     ,
0.58974359, 0.31632653, 0.4516129 , 0.        , 0.5       ,
0.17391304, 0.68611111, 0.61538462, 0.28571429, 0.4516129 ,
Output
Serial Monitor
Message (Enter to send message to 'XIAO_ESP32S3' on '/dev/cu.usbmodem101')
Both NL & CR
115200 baud
Edge Impulse Inferencing Demo
Edge Impulse standalone inferencing (Arduino)
run_classifier returned: 0
Timing: DSP 0 ms, inference 2024 ms, anomaly 0 ms
Predictions:
value: 0.36065
indexing: 14/49
Ln 256, Col 18
XIAO_ESP32S3 on /dev/cu.usbmodem101


'winddirDegree'
'windspeedKmph'
'precipMM'
'uvIndex',
'tempC'
'humidity'
'pressure'
A0
A1
D2
A3
SDA
SCL
seeed studio
Model:XIAO-ESP32-S3
FCC ID:Z4T-XIAOESP32S3
Future Temp oC

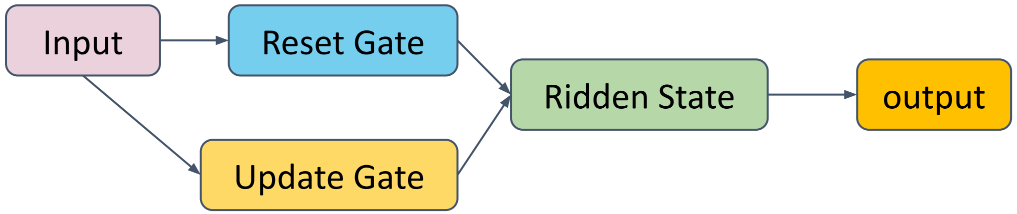

Temperature
Humidity
Pressure
Precipitation
UV-Index
Wind Speed
Wind Direction
Raw Sensor Data Reading
Data Normalization & Buffer (7 days x 24h)
Data Flattening (1,176 features)
Inference
Rescaling inference results
Future Temp oC
Training data_min & data_max

# GRU (Gated Recurrent Unit)

GRU (Gated Recurrent Unit) is a simplified version of LSTM with just two gates:

1. Reset Gate: Controls how much past information to forget
2. Update Gate: Decides how much past information to pass along

Key differences from LSTM:




- No separate cell state
- Fewer parameters, making it faster to train
- Often performs similarly to LSTM despite being simpler

Common applications:

- Machine translation
- Text generation
- Time series analysis
- Speech processing

The simpler architecture makes GRU a good choice when computational resources are limited or when working with smaller datasets.

# *VerneBOT*: GENERATING TEXTS LIKE *JULES VERNE*

An Introduction to Language Models

Prof. Marcelo Rovai, UNIFEI

# Questions?

TINYML4D