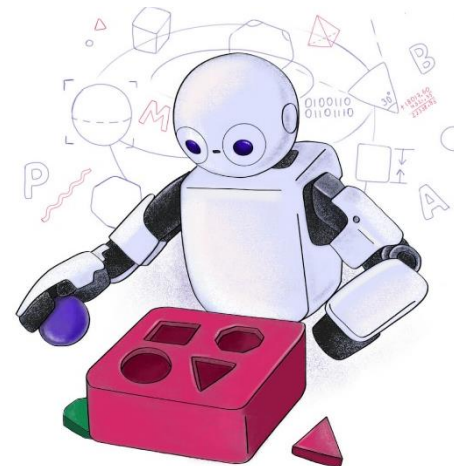


TP557 - Tópicos avançados em IoT e Machine Learning: *TensorFlow Lite*



Inatel

Créditos: Prof. Marcelo Rovai UNIFEI

Samuel Baraldi Mafra
samuelbmafra@inatel.br

Samuel Baraldi Mafra

- Possui pós-doutorado na Universidade Federal do Paraná (2015-2018).
- Doutor em Ciências na área de telecomunicações e redes pela Universidade Tecnológica Federal do Paraná (2015).
- Mestre em Engenharia Elétrica com ênfase em telecomunicações pela Universidade Federal do Paraná (2012).
- Engenheiro Eletricista pela Universidade do Estado de Santa Catarina (2010).

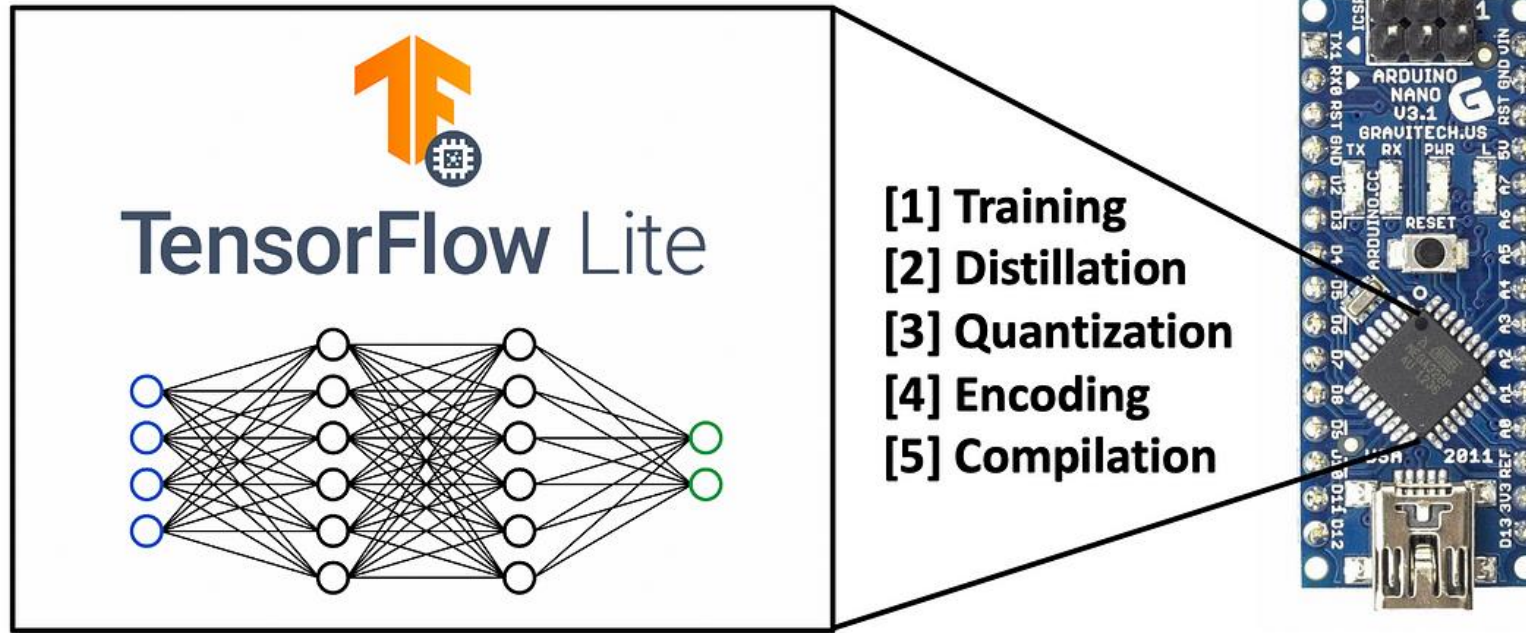


Samuel Baraldi Mafra

- Instituto Nacional de Telecomunicações - Inatel.
 - Professor no curso de graduação em engenharia de computação;
 - Professor e pesquisador nos cursos de mestrado e doutorado em telecomunicações;
 - Professor no cursos de especialização em 5G, IoT e redes de telecomunicação;
 - Coordenador do IoT Research Group – Inatel.



TinyML



TinyML

Processamento na borda com dispositivos embarcados

- Características dos dispositivos IoT:
- Baixo consumo de energia;
- Baixo poder computacional;
- Baixo custo.

Processamento na borda com dispositivos embarcados

- A energia de processamento depende basicamente do número de ciclos de clock executadas pelo microcontrolador.
- "Os programadores não poderão ignorar o custo de energia dos programas que escrevem ... Você precisa de ferramentas que forneçam feedback e digam o quão boas são suas decisões. Atualmente, as ferramentas não fornecem esse tipo de feedback." (Steve Furber, ARM, 2010)

Processamento tradicional

Machine Learning Workflow



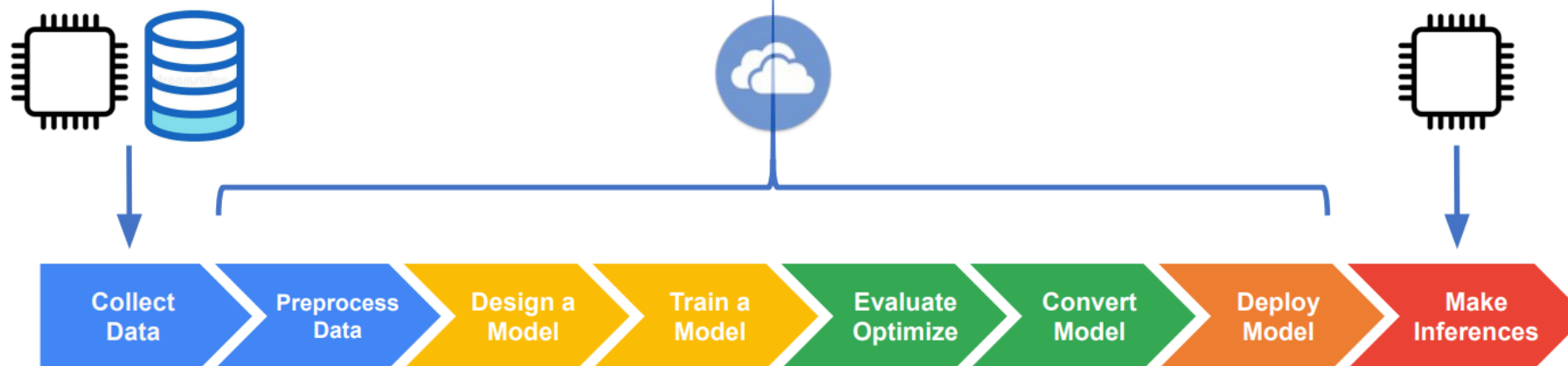
Processamento na borda com dispositivos embarcados

Tiny Machine Learning Workflow (“What”)



Processamento na borda com dispositivos embarcados

Tiny Machine Learning Workflow (“Where”)



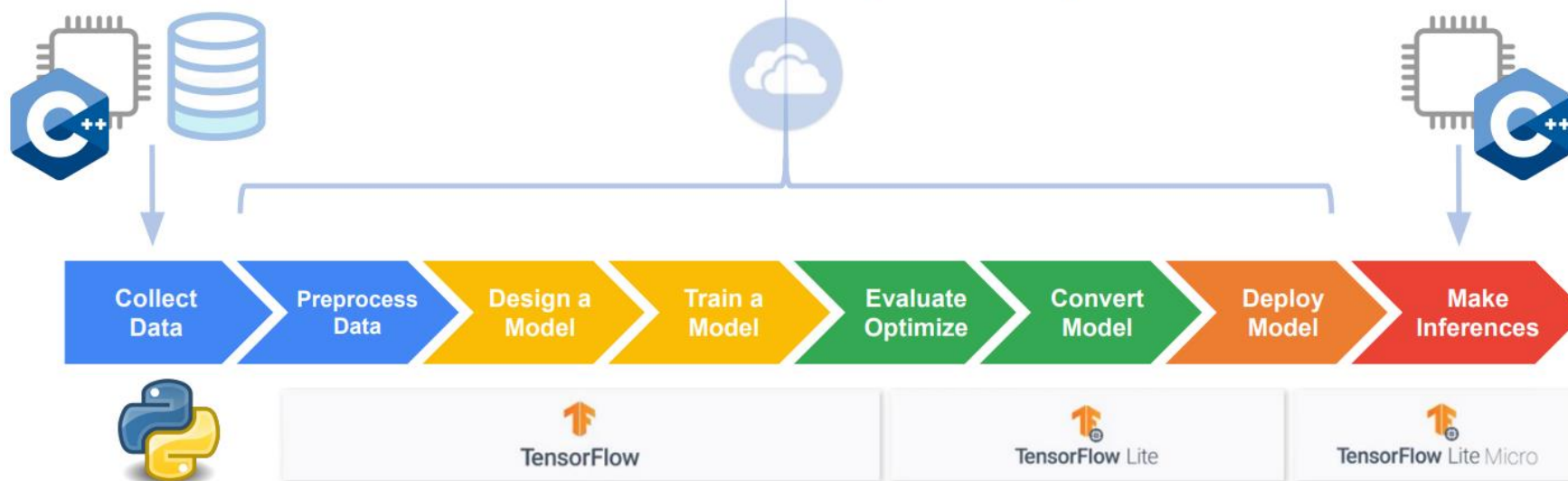
Processamento na borda com dispositivos embarcados

Machine Learning Workflow (“How”)

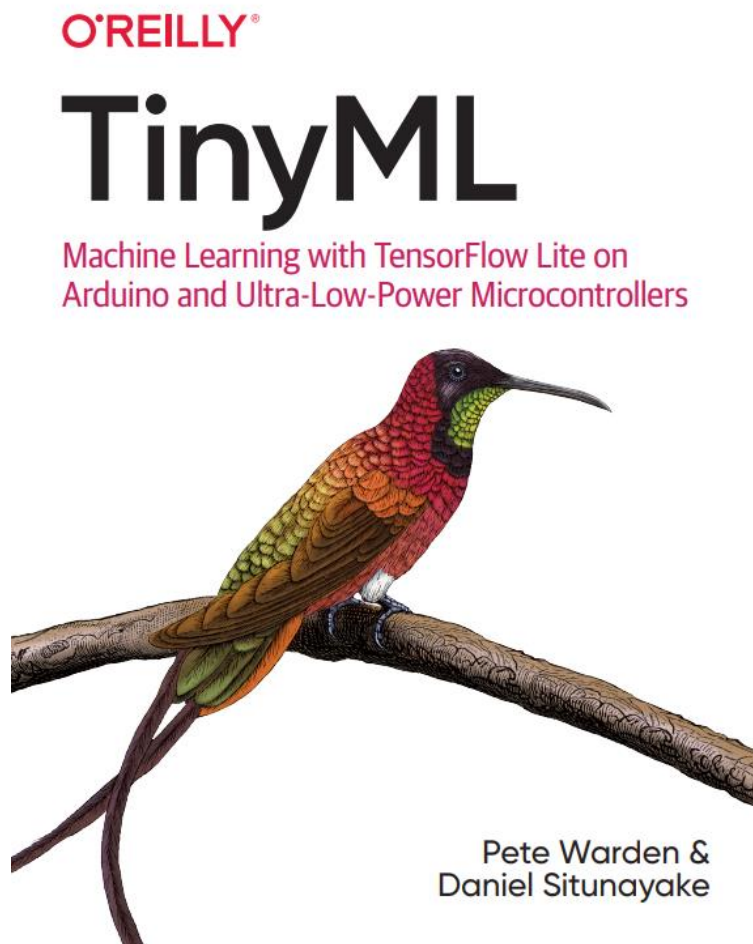


Processamento na borda com dispositivos embarcados

Machine Learning Workflow ("How")



Referência



Tinyml: Machine Learning with Tensorflow Lite on Arduino and Ultra-Low-Power Microcontrollers

Inatel

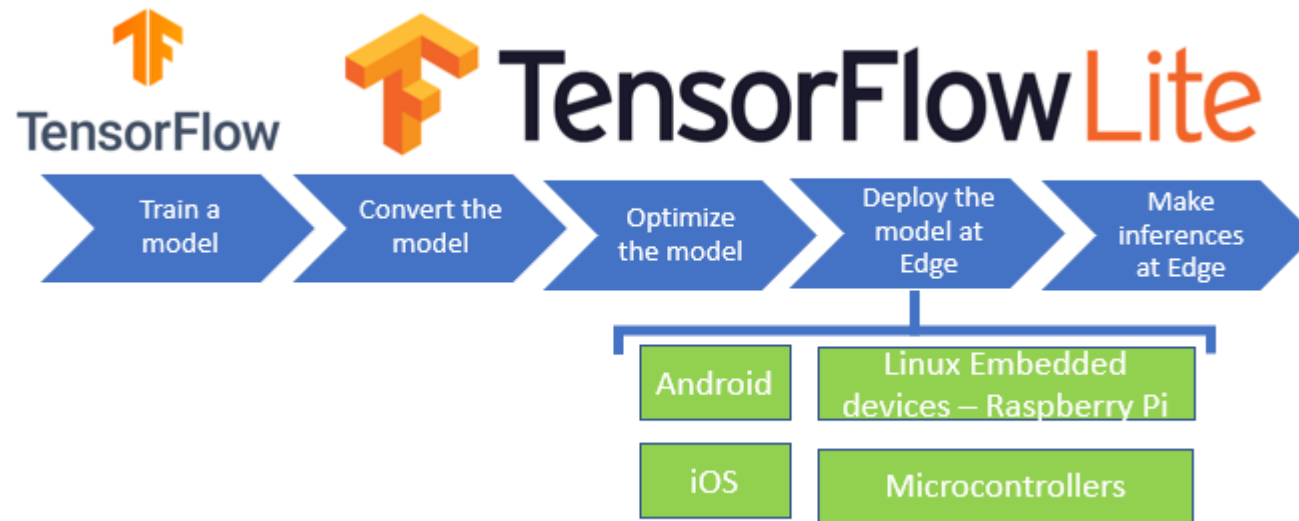
CAMINHOS
QUE CONECTAM
COM O FUTURO

Conteúdo

- Introdução
 - O que é o TensorFlow Lite?
 - Por que usar o TensorFlow Lite?
 - Benefícios de usar o TensorFlow Lite
 - O que o TensorFlow Lite pode fazer?
 - Visão geral do fluxo de trabalho do TensorFlow Lite
 - Convertendo um modelo do TensorFlow em TensorFlow Lite

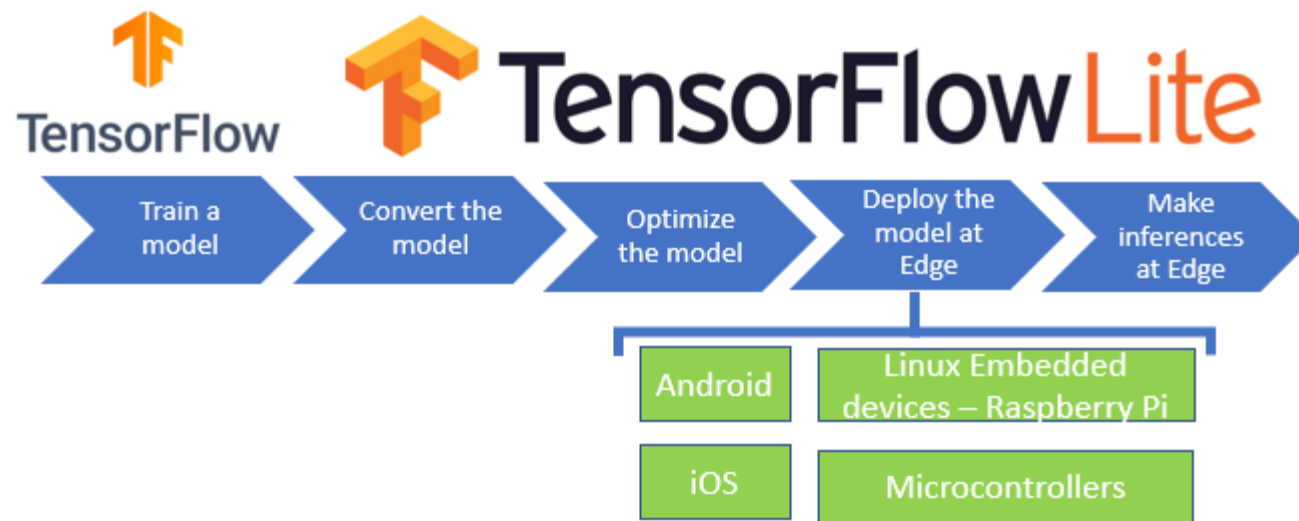
TensorFlow Lite

- O TensorFlow Lite é um conjunto de ferramentas para machine learning no dispositivo que ajuda os desenvolvedores a executar modelos em dispositivos móveis, incorporados e de IoT.



TensorFlow Lite

- O TensorFlow é uma biblioteca de aprendizado de máquina geral que pode ser usada para treinar e implantar modelos de aprendizado de máquina em uma variedade de plataformas, incluindo computadores, dispositivos móveis e dispositivos incorporados.
- O TensorFlow Lite é uma versão reduzida do TensorFlow que é otimizada para desempenho e eficiência em dispositivos móveis e incorporados.



TensorFlow Lite

Por que usar o TensorFlow Lite

- Otimizado para machine learning
 - latência (não há ida e volta para um servidor),
 - privacidade (nenhum dado pessoal sai do dispositivo),
 - conectividade (conectividade com a Internet não é necessária),
 - tamanho (tamanho reduzido do modelo e do binário)
 - consumo de energia (inferência eficiente e falta de conexões de rede)

TensorFlow Lite

Por que usar o TensorFlow Lite

- Suporte a várias plataformas, inclusive dispositivos Android e iOS, Linux incorporado e microcontroladores
- Compatibilidade com diversas linguagens, incluindo Java, Swift, Objective-C, C++ e Python
- Alto desempenho com aceleração de hardware e otimização de modelos

TensorFlow Lite

	 TensorFlow	 TensorFlow Lite
Topology	Variable	Fixed
Weights	Variable	Fixed
Binary Size	Unimportant	High Priority
Distributed Compute	Needed	Not Needed
Developer Background	ML Researcher	Application Developer




TensorFlow Lite

	 TensorFlow	 TensorFlow Lite	 TensorFlow Lite Micro
Training	Yes	No	No
Inference	Yes <i>(but inefficient on edge)</i>	Yes <i>(and efficient)</i>	Yes <i>(and even more efficient)</i>
How Many Ops	~1400	~130	~50
Native Quantization Tooling + Support	No	Yes	Yes

TensorFlow Lite

	 TensorFlow	 TensorFlow Lite	 TensorFlow Lite Micro
Needs an OS	Yes	Yes	No
Memory Mapping of Models	No	Yes	Yes
Delegation to accelerators	Yes	Yes	No

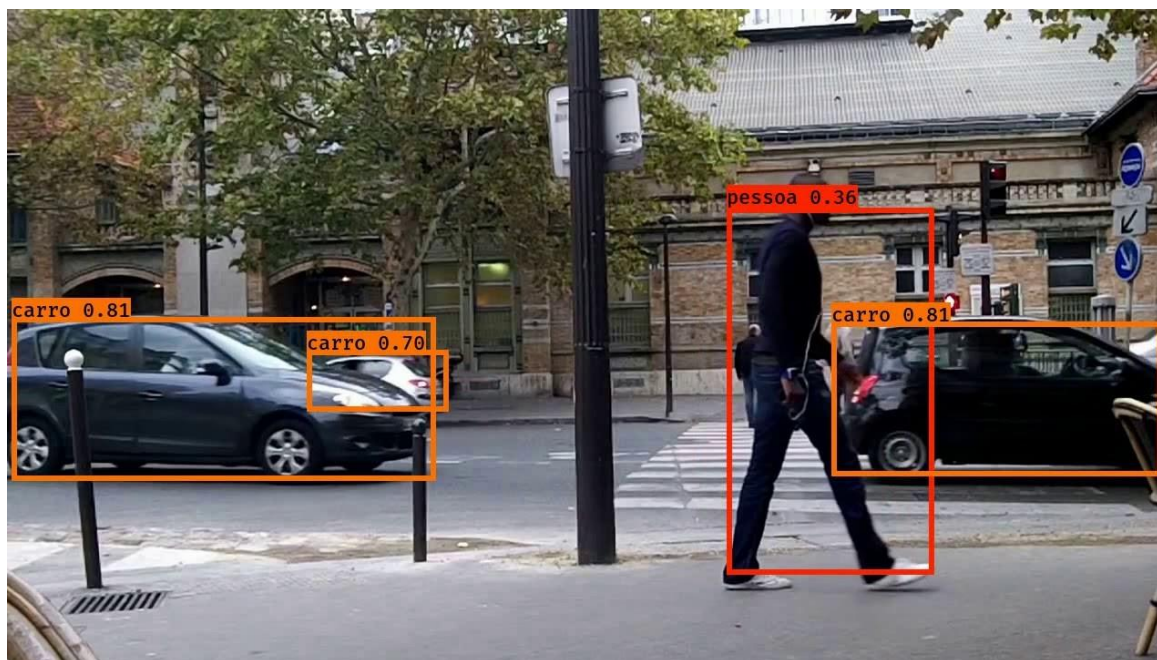
TensorFlow Lite

	 TensorFlow	 TensorFlow Lite	 TensorFlow Lite Micro
Base Binary Size	3MB+	100KB	~10 KB
Base Memory Footprint	~5MB	300KB	20KB
Optimized Architectures	X86, TPUs, GPUs	Arm Cortex A, x86	Arm Cortex M, DSPs, MCUs

TensorFlow Lite

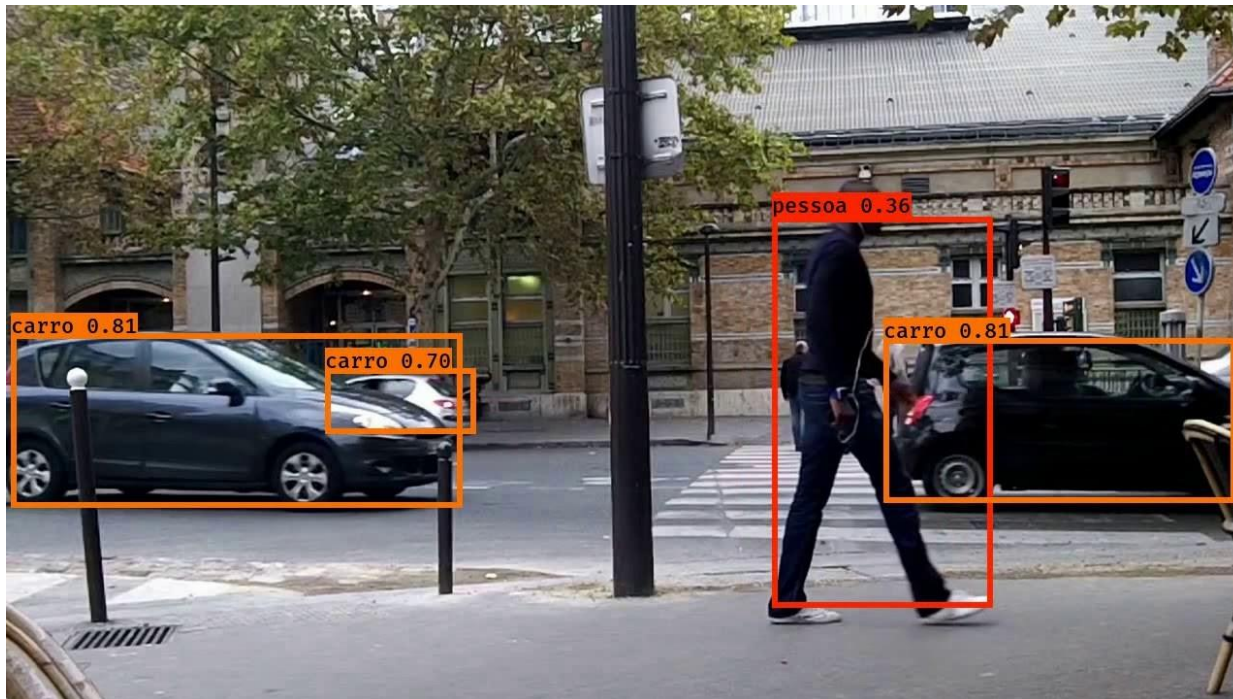
O TensorFlow Lite é um conjunto de ferramentas para machine learning no dispositivo que ajuda os desenvolvedores a executar modelos em dispositivos móveis, incorporados e de IoT. Ele pode ser usado para uma ampla variedade de tarefas, incluindo:

- Reconhecimento de imagem: O TensorFlow Lite pode ser usado para identificar objetos em imagens, como pessoas, animais, objetos e cenas.



TensorFlow Lite

Detecção de objeto: O TensorFlow Lite pode ser usado para identificar e rastrear objetos em tempo real, como carros, pessoas e animais.



TensorFlow Lite

Reconhecimento de voz: O TensorFlow Lite pode ser usado para reconhecer fala humana, como comandos de voz e perguntas.



TensorFlow Lite

Processamento de linguagem natural: O TensorFlow Lite pode ser usado para entender e responder a texto natural, como perguntas e solicitações.



TensorFlow Lite

Análise de vídeo: O TensorFlow Lite pode ser usado para analisar vídeos para detectar objetos, pessoas e comportamentos.



TensorFlow Lite

- Um modelo do TensorFlow Lite é representado em um formato aberto especial e eficiente conhecido como FlatBuffers, identificado pela extensão de arquivo .tflite.
- Desenvolvida pelo Google.
- Flatbuffers é especialmente útil em cenários em que a velocidade e o tamanho compacto dos dados são essenciais, como jogos, aplicativos móveis e sistemas de comunicação de rede.

TensorFlow Lite: Geração de modelos

É possível gerar um modelo do TensorFlow Lite das seguintes maneiras:

Use um modelo existente do TensorFlow Lite: consulte os exemplos de TensorFlow Lite para escolher um modelo existente. Os modelos podem ou não conter metadados.

Apps de exemplo do TensorFlow Lite

Conheça modelos pré-treinados do TensorFlow Lite e aprenda a usá-los em apps de amostra para várias aplicações de ML.



Preenchimento automático

Gere sugestões para entradas de texto usando um modelo de linguagem Keras.

[Visão geral do modelo →](#)

[Teste no Android](#)



Classificação de imagens

Identifique centenas de objetos, incluindo pessoas, atividades, animais, plantas e locais.

[Visão geral do modelo →](#)

[Teste no Android](#)

[Teste no iOS](#)

[Teste no Raspberry Pi](#)



Detecção de objetos

Detecte diversos objetos com caixa delimitadora. Sim, isso inclui cães e gatos.

[Visão geral do modelo →](#)

[Teste no Android](#)

[Teste no iOS](#)

[Teste no Raspberry Pi](#)

TensorFlow Lite

É possível gerar um modelo do TensorFlow Lite das seguintes maneiras:

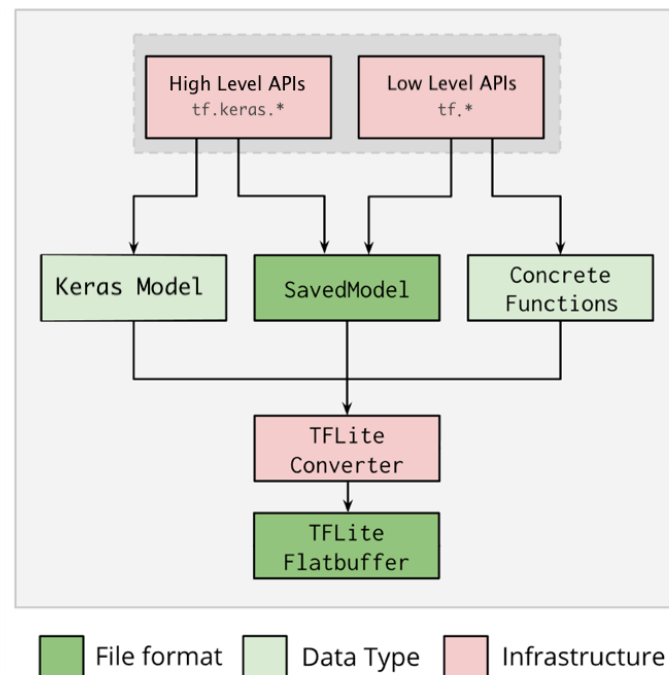
Crie um modelo do TensorFlow Lite: use o TensorFlow Lite Model Maker para criar um modelo com seu próprio conjunto de dados personalizado. Por padrão, todos os modelos contêm metadados. Utiliza transfer learning com modelos pré treinados.

https://www.tensorflow.org/lite/models/modify/model_maker?hl=pt-br

TensorFlow Lite

É possível gerar um modelo do TensorFlow Lite das seguintes maneiras:

Converta um modelo do TensorFlow em um do TensorFlow Lite: use o TensorFlow Lite Converter para converter modelos. Durante a conversão, você pode aplicar otimizações como a quantização para diminuir o tamanho e a latência do modelo com pouca ou nenhuma perda na acurácia. Por padrão, nenhum dos modelos contém metadados.



TensorFlow Lite

Quando utiliza outros frameworks é necessário converter primeiro para o TensorFlow.



TensorFlow Lite

- Como o tensorflow lite roda em dispositivos embarcados?
 - Operações de poda;
 - Operações de quantização.

Poda de redes neurais (Pruning)

- Existem diferentes maneiras de podar uma rede neural.
 - (1) Você pode podar pesos. Isso é feito definindo parâmetros individuais como zero e tornando a rede esparsa. Isso reduziria o número de parâmetros no modelo, mantendo a mesma arquitetura.
 - (2) Você pode remover nós inteiros da rede. Isso tornaria a própria arquitetura da rede menor, ao mesmo tempo que visava manter a precisão da rede inicial maior.

Poda de redes neurais (Pruning)

Remover um peso é essencialmente defini-lo como zero. Você pode minimizar o efeito na rede removendo pesos que já estão próximos de zero, ou seja, de magnitude baixa. Isto pode ser implementado removendo todos os pesos abaixo de um determinado limite.



Poda de redes neurais (Pruning)

- Em vez de apenas pesos, as ativações nos dados de treinamento podem ser usadas como critério para remoção.
- Ao executar um conjunto de dados em uma rede, certas estatísticas das ativações podem ser observadas.
- Você pode observar que alguns neurônios sempre produzem valores próximos de zero.

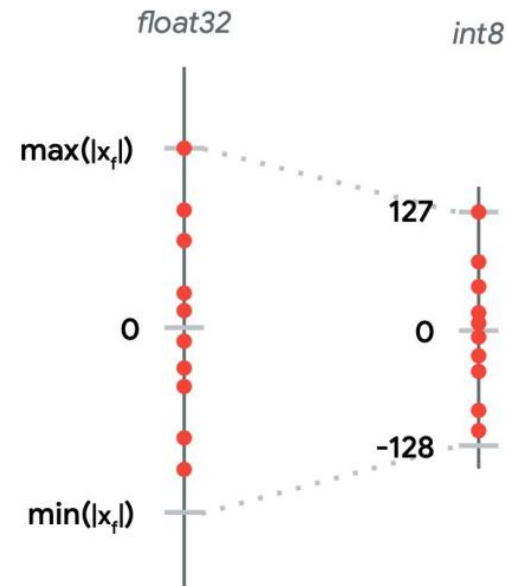
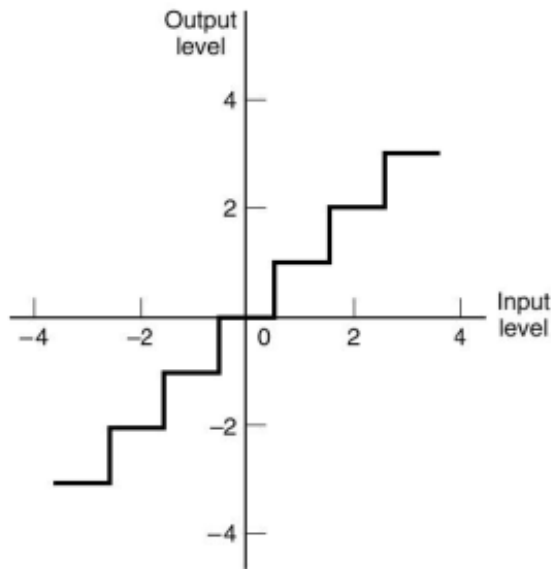
Poda de redes neurais (Pruning)

- A redundância de parâmetros pode significar que um neurônio pode ser removido. Se dois neurônios em uma camada tiverem pesos ou ativações muito semelhantes, isso pode significar que estão fazendo a mesma coisa.



Quantização

- A quantização é uma otimização que funciona reduzindo a precisão dos números usados para representar os parâmetros de um modelo, que por padrão são números de ponto flutuante de 32 bits.



Quantização

- A operação de quantização não é reversível.
- Imagem Lena quantizada em 16, 8, 4 e 2 bits / pixel



Quantização

- Perda de precisão pela quantização

	Floating-point Baseline	Post-training Quantization (PTQ)	Accuracy Drop
MobileNet v1 1.0 224	71.03%	69.57%	▼1.46%
MobileNet v2 1.0 224	70.77%	70.20%	▼0.57%
Resnet v1 50	76.30%	75.95%	▼0.35%

TensorFlow Lite micro

- O TensorFlow Lite for Microcontrollers (TFLM, abreviadamente) foi projetado para executar modelos de aprendizado de máquina em microcontroladores e outros dispositivos com apenas alguns kilobytes de memória.
- Com a finalidade de implantar o modelo de aprendizado de máquina em dispositivos embarcados, ele também é chamado de TinyML.
- O Tensorflow é uma das estruturas de aprendizado profundo comumente usadas e, a partir da versão 2.x, eles oferecem alguns recursos de aprendizado de máquina para sistemas embarcados por meio do tensorflow lite.
- Ao contrário do processador móvel de alto desempenho (como a série Cortex-A), o microcontrolador (série Cortex-M ou ESP32) tem baixo consumo de energia e pode ser implantado de várias formas em produtos do cliente, como geladeira, máquina de lavar e assim por diante.

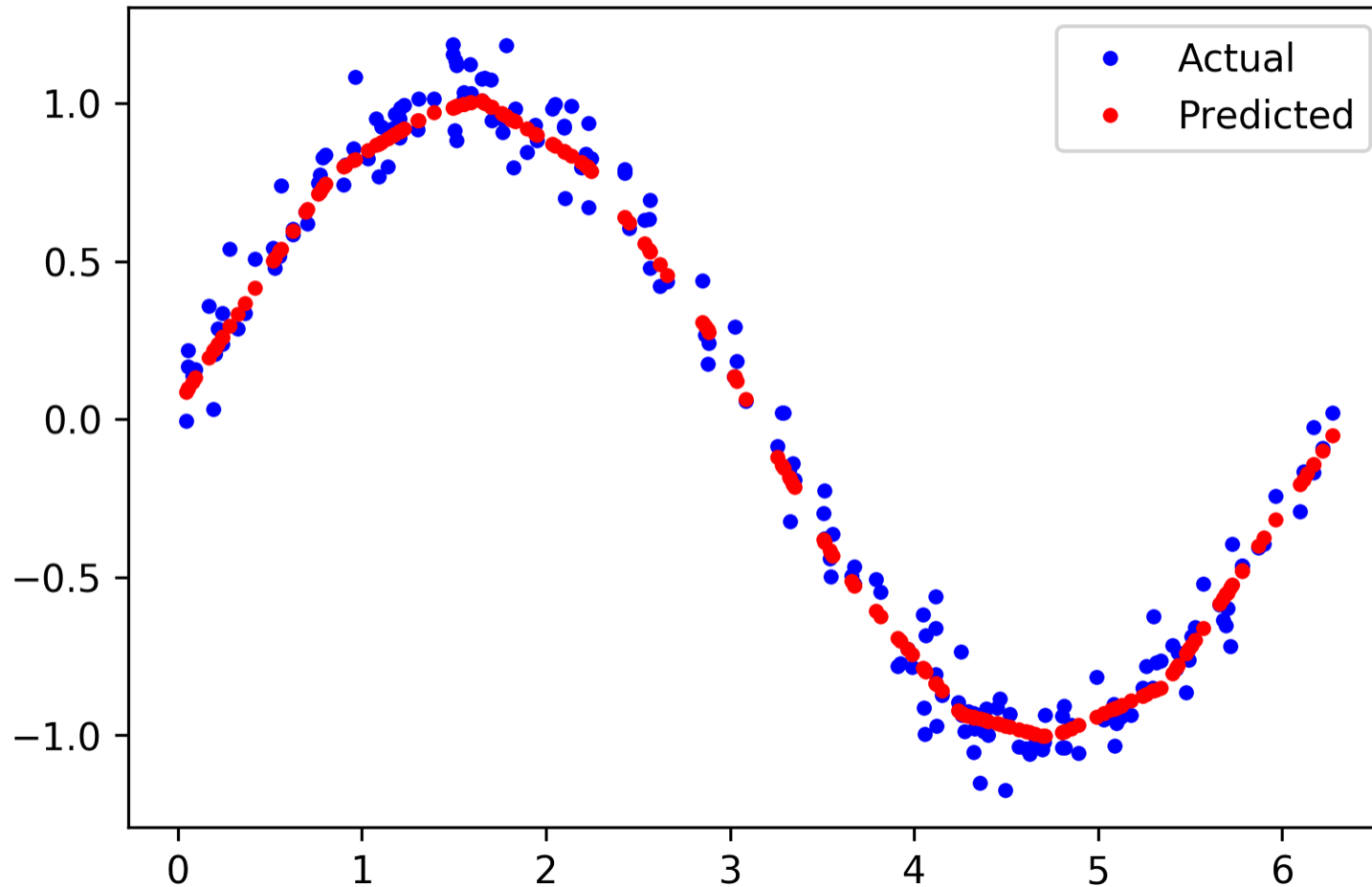
TensorFlow Lite micro

- O TensorFlow Lite para microcontroladores é escrito em C++ 11 e requer uma plataforma de 32 bits.
- Ele foi testado extensivamente com muitos processadores baseados na arquitetura Arm Cortex-M Series e foi portado a outras arquiteturas, incluindo ESP32. O framework está disponível como uma biblioteca Arduino.
- Ele também pode gerar projetos para ambientes de desenvolvimento, como o Mbed. Ele é de código aberto e pode ser incluído em qualquer projeto C++ 11.

Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$

Comparison of predictions and actual values



Exemplo

- Hello world seno de um ângulo $\tilde{H}(x) = \sin(x)$
- Importação de bibliotecas e definição das sementes

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

plt.rcParams['figure.figsize'] = (16, 10)
plt.rc('font', size=15)

# define random seed for reproducibility
np.random.seed(1) # numpy seed
tf.random.set_seed(1) # tensorflow global random seed
```

Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$
- Geração da onda senoidal

```
# Generate a uniformly distributed set of random numbers in the range from
# 0 to 2π, which covers a complete sine wave oscillation
X = np.random.uniform(
    low=0, high=2*math.pi, size=10000).astype(np.float32)

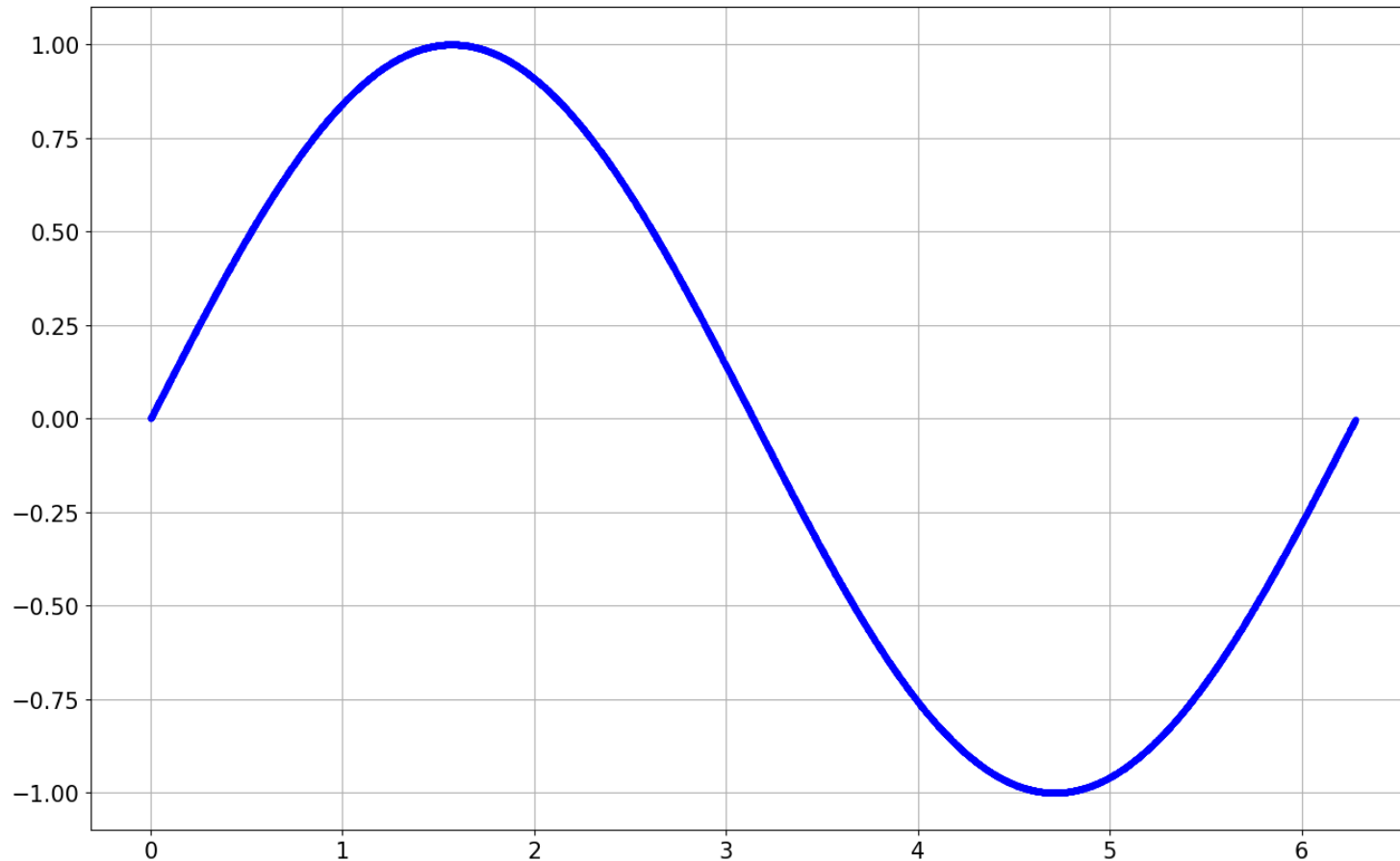
# Shuffle the values to guarantee they're not in order
np.random.shuffle(X)

# Calculate the corresponding sine values
y = np.sin(X).astype(np.float32)

# Plot our data. The 'b.' argument tells the library to print blue dots.
plt.plot(X, y, 'b.')
plt.grid()
plt.show()
```

Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$
- Geração da onda senoidal



Exemplo

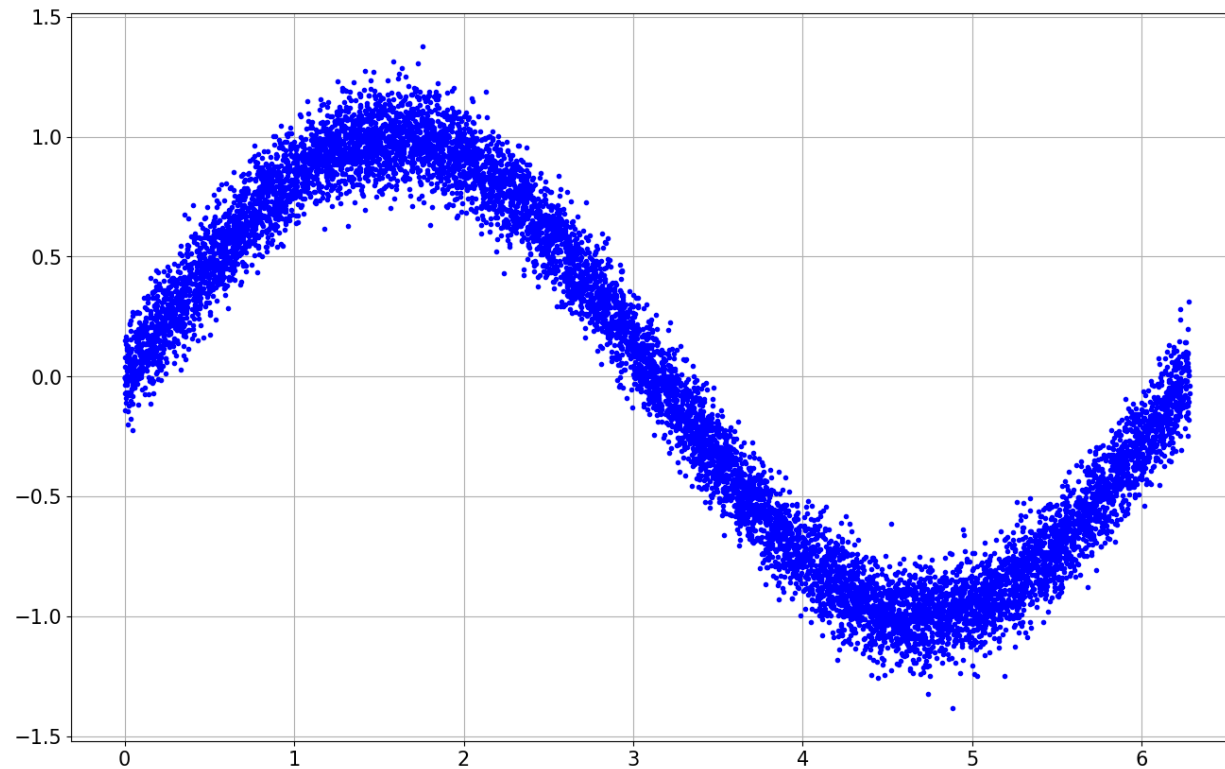
- Hello world seno de um ângulo $\tilde{H}(x) = \sin(x)$
- Adição de ruído

```
[5] # Add a small random number to each y value
    y += 0.1 * np.random.randn(*y.shape)

    # Plot our data
    plt.plot(X, y, 'b.')
    plt.grid()
    plt.show()
```

Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$
- Adição de ruído



Exemplo

- Hello world seno de um ângulo $\tilde{H}(x) = \sin(x)$
- Separação das amostras em treinamento, validação e teste
- Train data: 60%
- Validation data: 20%
- Test data: 20%

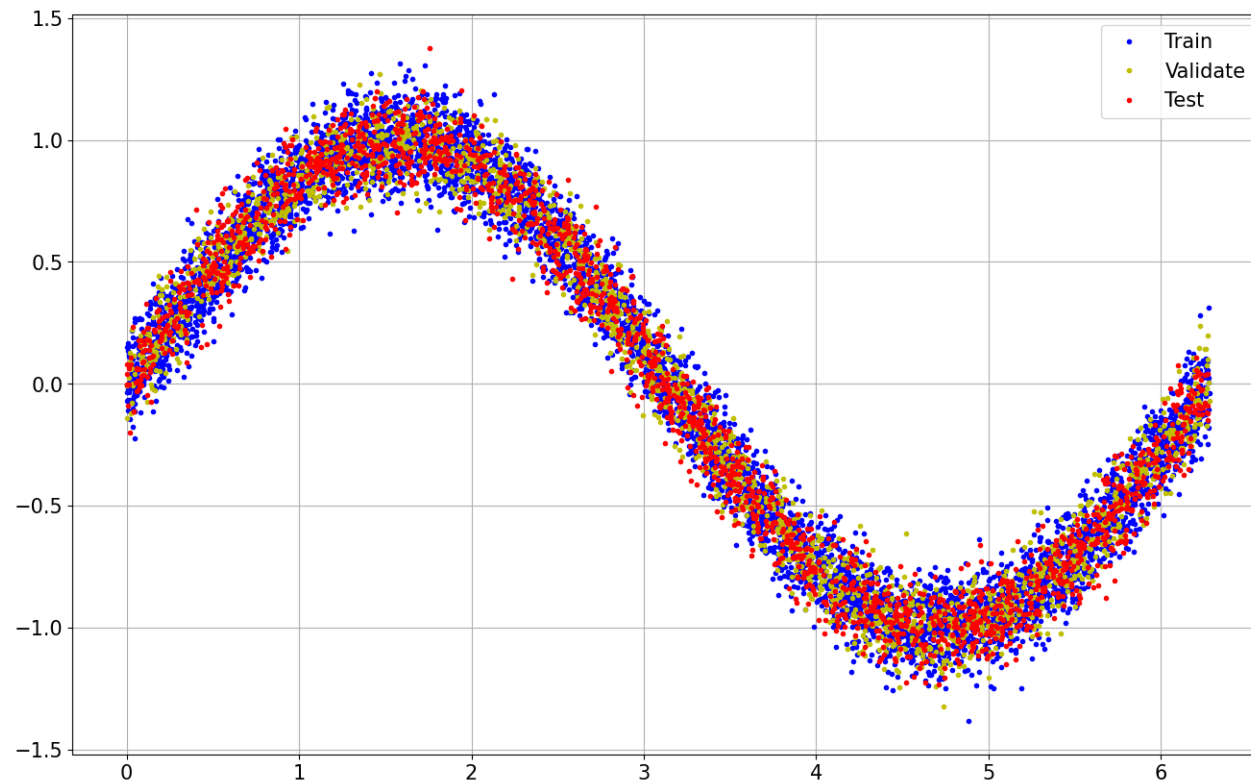
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.25, random_state=1)

# Plot the data in each partition in different colors:
plt.plot(X_train, y_train, 'b.', label="Train")
plt.plot(X_val, y_val, 'y.', label="Validate")
plt.plot(X_test, y_test, 'r.', label="Test")
plt.legend()
plt.grid()
plt.show()
```

Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$
- Separação das amostras em treinamento, validação e teste
- Train data: 60%
- Validation data: 20%
- Test data: 20%



Exemplo

- Hello world seno de um ângulo $H(\tilde{x}) = \sin(x)$
- Modelo inicial

```
model = tf.keras.Sequential(name='sine')
model.add(tf.keras.layers.Dense(16, activation='relu', input_shape=(1, )))
model.add(tf.keras.layers.Dense(1))

model.summary()
```

```
model.compile(optimizer='adam', loss='mse', metrics=['accuracy', 'mae'])
```

Model: "sine"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 16)	32
dense_1 (Dense)	(None, 1)	17

```
=====
Total params: 49 (196.00 Byte)
Trainable params: 49 (196.00 Byte)
Non-trainable params: 0 (0.00 Byte)
```

Exemplo

- Hello world seno de um ângulo
- Treinamento

$$\tilde{H}(x) = \sin(x)$$

```
history = model.fit(X_train, y_train, epochs=1000, batch_size=15, validation_data=(X_val, y_val), verbose=False)
```

Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$
- Gráfico da métrica loss para treinamento e validação

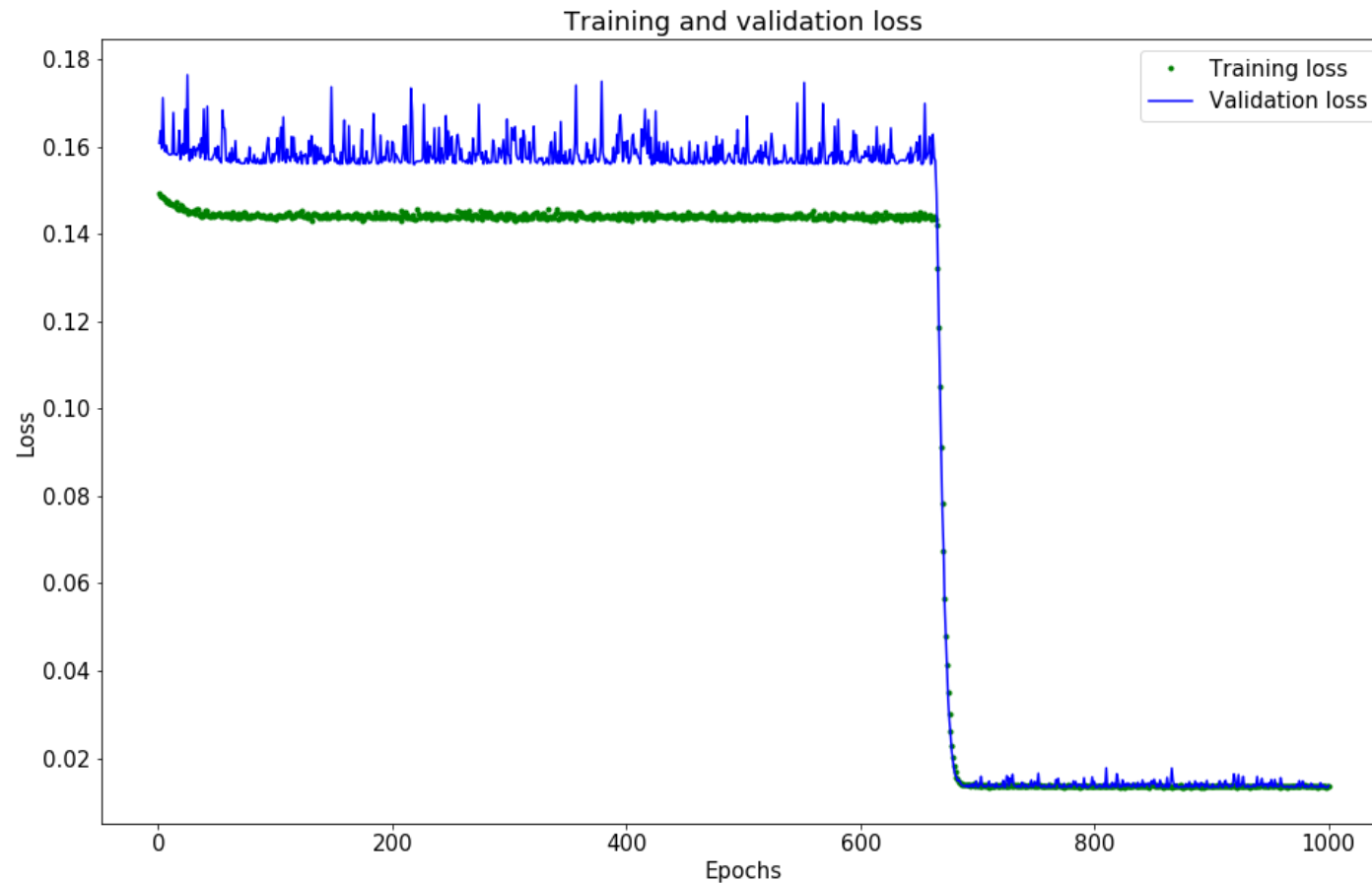
```
# Draw a graph of the loss, which is the distance between
# the predicted and actual values during training and validation.
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(loss) + 1)

plt.plot(epochs, loss, 'g.', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

Exemplo

- Hello world seno de um ângulo $H(\tilde{x}) = \sin(x)$
- Gráfico da métrica loss para treinamento e validação



Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$
- Cálculo do loss e predição do modelo para o dataset de test

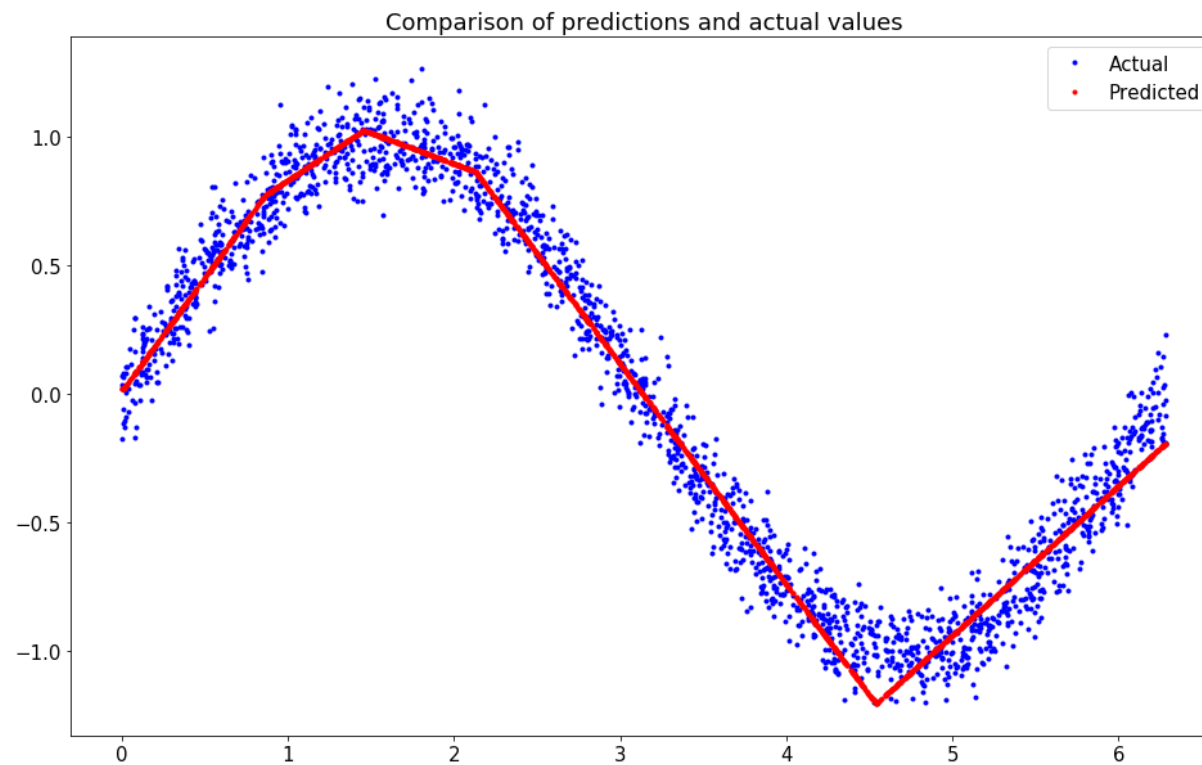
```
# Calculate and print the loss on our test dataset
loss = model.evaluate(X_test, y_test)

# Make predictions based on our test dataset
predictions = model.predict(X_test)

# Graph the predictions against the actual values
plt.clf()
plt.title('Comparison of predictions and actual values')
plt.plot(X_test, y_test, 'b.', label='Actual')
plt.plot(X_test, predictions, 'r.', label='Predicted')
plt.legend()
plt.show()
```


Exemplo

- Hello world seno de um ângulo $H(\tilde{x}) = \sin(x)$
- Predição do modelo para o dataset de teste



Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$
- Segundo modelo com uma camada densa extra

```
model_2 = tf.keras.Sequential(name='sine2')

# First layer takes a scalar input and feeds it through 16 "neurons". The
# neurons decide whether to activate based on the 'relu' activation function.
model_2.add(tf.keras.layers.Dense(16, activation='relu', input_shape=(1,)))

# The new second layer may help the network learn more complex representations
model_2.add(tf.keras.layers.Dense(16, activation='relu'))

# Final layer is a single neuron, since we want to output a single value
model_2.add(tf.keras.layers.Dense(1))

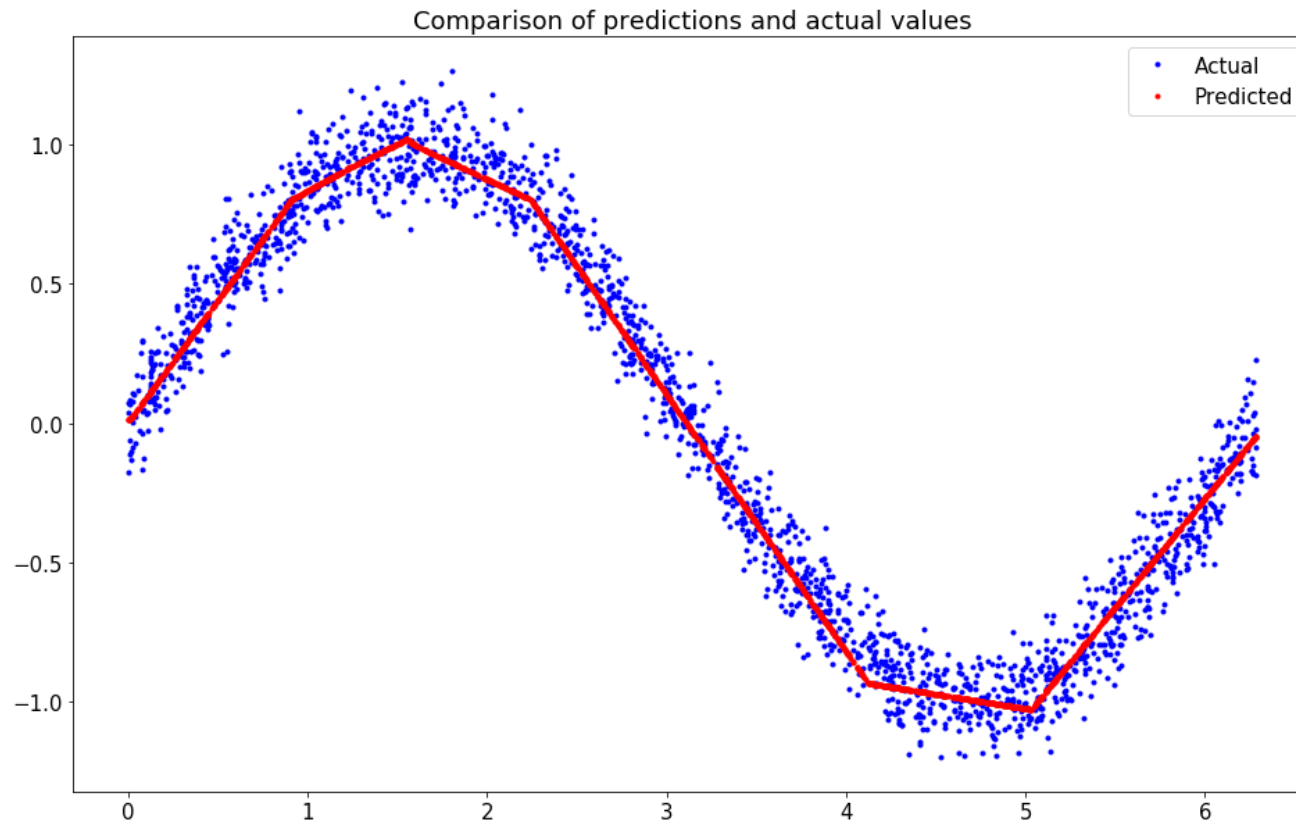
# Compile the model using a standard optimizer and loss function for regression
model_2.compile(optimizer='adam', loss='mse', metrics=['accuracy', 'mae'])

model_2.summary()
```

```
history = model_2.fit(X_train, y_train, epochs=500, batch_size=64,
                      validation_data=(X_val, y_val), verbose=False)
```

Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$
- Predição do modelo para o dataset de teste



Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$

Conversão dos modelos

Criação de um diretório para os modelos

```
import os
MODEL_DIR = './models/'
if not os.path.exists(MODEL_DIR):
    os.mkdir(MODEL_DIR)
```

Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$

Conversão dos modelos

Criação de um diretório para os modelos

```
import os
MODEL_DIR = './models/'
if not os.path.exists(MODEL_DIR):
    os.mkdir(MODEL_DIR)
```

Exemplo

- Hello world seno de um ângulo $H(\tilde{x}) = \sin(x)$

Conversão dos modelos

```
# Convert the model to the TensorFlow Lite format without quantization
converter = tf.lite.TFLiteConverter.from_keras_model(model_2)
model_no_quant_tflite = converter.convert()

# # Save the model to disk
open(MODEL_DIR + 'model_no_quant.tflite', "wb").write(model_no_quant_tflite)

# Convert the model to the TensorFlow Lite format with quantization
def representative_dataset():
    for i in range(500):
        yield([X_train[i].reshape(1, 1)])
# Set the optimization flag.
converter.optimizations = [tf.lite.Optimize.DEFAULT]
# Enforce full-int8 quantization (except inputs/outputs which are always float)
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
# Provide a representative dataset to ensure we quantize correctly.
converter.representative_dataset = representative_dataset
model_tflite = converter.convert()

# Save the model to disk
open(MODEL_DIR + 'model.tflite', "wb").write(model_tflite)
```

Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$

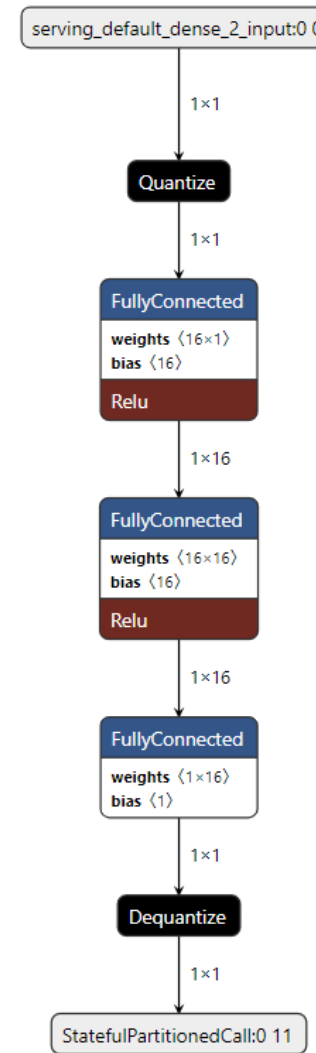
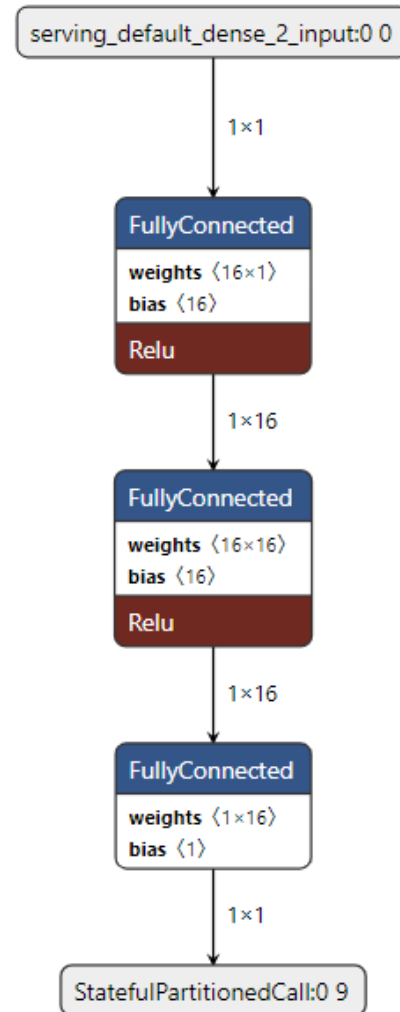
Conversão dos modelos

Comparação dos tamanhos dos modelos

```
model_no_quant_size = os.path.getsize(MODEL_DIR + 'model_no_quant.tflite')
print("Model is %d bytes" % model_no_quant_size)
model_size = os.path.getsize(MODEL_DIR + 'model.tflite')
print("Quantized model is %d bytes" % model_size)
difference = model_no_quant_size - model_size
print("Difference is %d bytes" % difference)
```

```
Model is 3128 bytes
Quantized model is 2976 bytes
Difference is 152 bytes
```


Exemplo



<https://netron.app/>

Exemplo

INPUTS		
input	name: serving_default_dense_2_input:0	+
weights	name: sine2/dense_2/MatMul	-
	tensor: float32[16,1]	
	location: 4	
	<div>[[0.30061519145965576], [-0.1736752986907959], [-0.16740983724594116], [-0.003516256809234619], [-0.49522048234939575], [-0.5830960273742676], [0.35114142298698425], [-0.03327280282974243], [-0.2239295244216919], [-0.49682486057281494], [0.27168044447898865],]</div>	

INPUTS		
input	name: tfl.quantize	+
weights	name: sine2/dense_2/MatMul	-
	tensor: int8[16,1]	
	quantization: 0.004591307137161493 * q	
	location: 6	
	<div>[[65], [-38], [-36], [-1], [-108], [-127], [76], [-7], [-49], [-108], [59],]</div>	

Exemplo

- Hello world seno de um ângulo $H(x) = \sin(x)$

Conversão dos modelos

Fazendo a predição para os modelos criados

```
# Instantiate an interpreter for each model
model_no_quant = tf.lite.Interpreter(MODEL_DIR + 'model_no_quant.tflite')
model = tf.lite.Interpreter(MODEL_DIR + 'model.tflite')

# Allocate memory for each model
model_no_quant.allocate_tensors()
model.allocate_tensors()

# Get the input and output tensors so we can feed in values and get the results
model_no_quant_input = model_no_quant.tensor(model_no_quant.get_input_details()[0]["index"])
model_no_quant_output = model_no_quant.tensor(model_no_quant.get_output_details()[0]["index"])
model_input = model.tensor(model.get_input_details()[0]["index"])
model_output = model.tensor(model.get_output_details()[0]["index"])

# Create arrays to store the results
model_no_quant_predictions = np.empty(X_test.size)
model_predictions = np.empty(X_test.size)

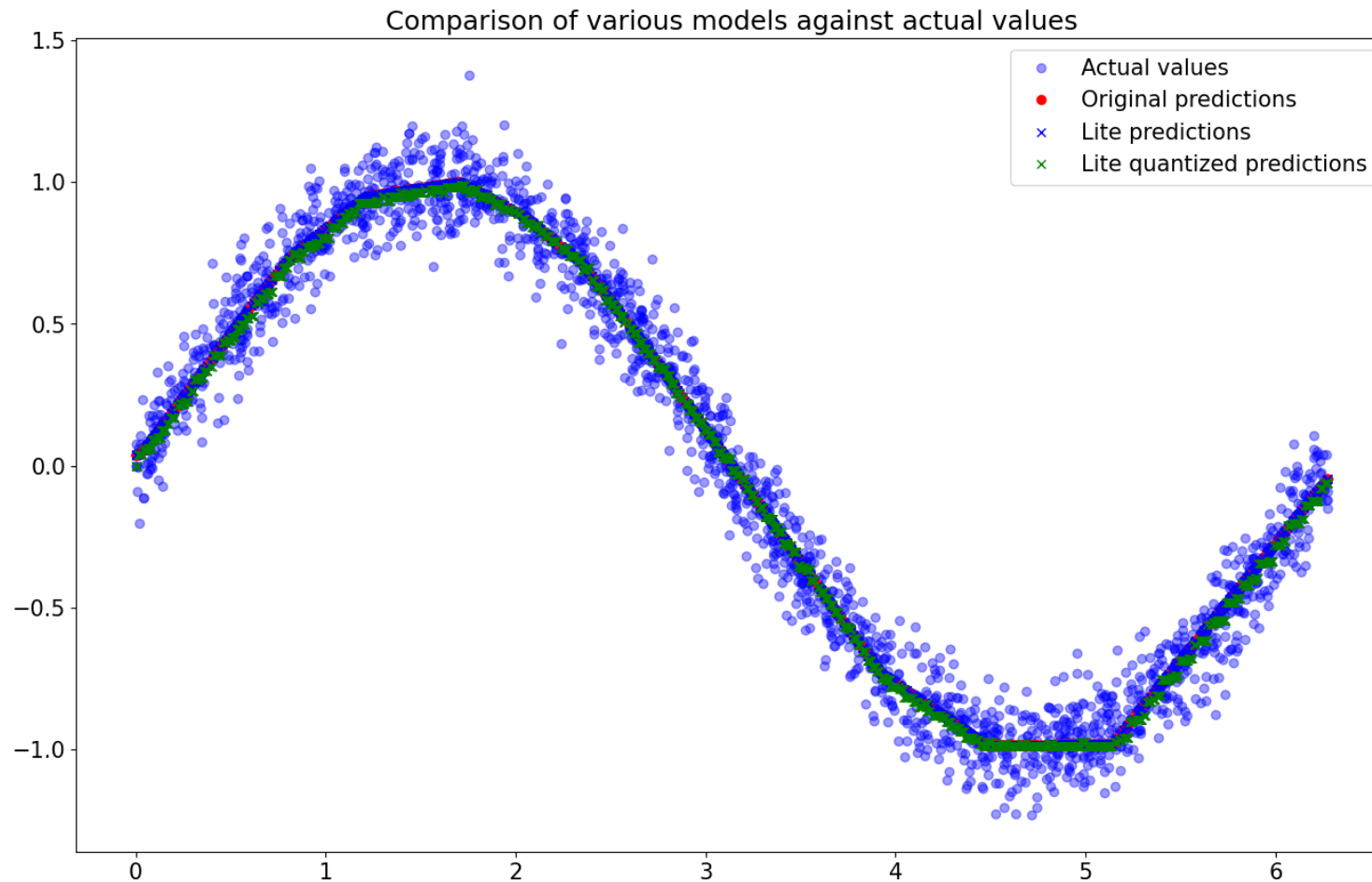
# Run each model's interpreter for each value and store the results in arrays
for i in range(X_test.size):
    model_no_quant_input().fill(X_test[i])
    model_no_quant.invoke()
    model_no_quant_predictions[i] = model_no_quant_output()[0]

    model_input().fill(X_test[i])
    model.invoke()
    model_predictions[i] = model_output()[0]
```

Exemplo

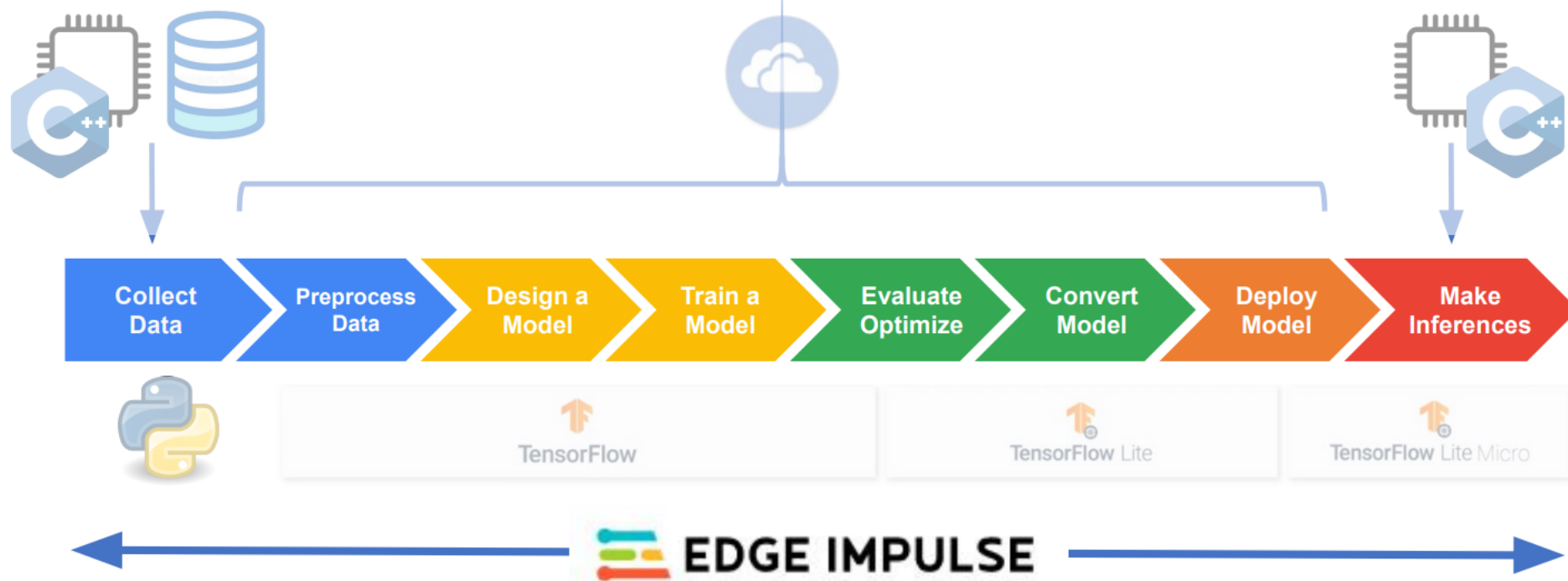
- Hello world seno de um ângulo $H(\tilde{x}) = \sin(x)$

Comparação dos modelos

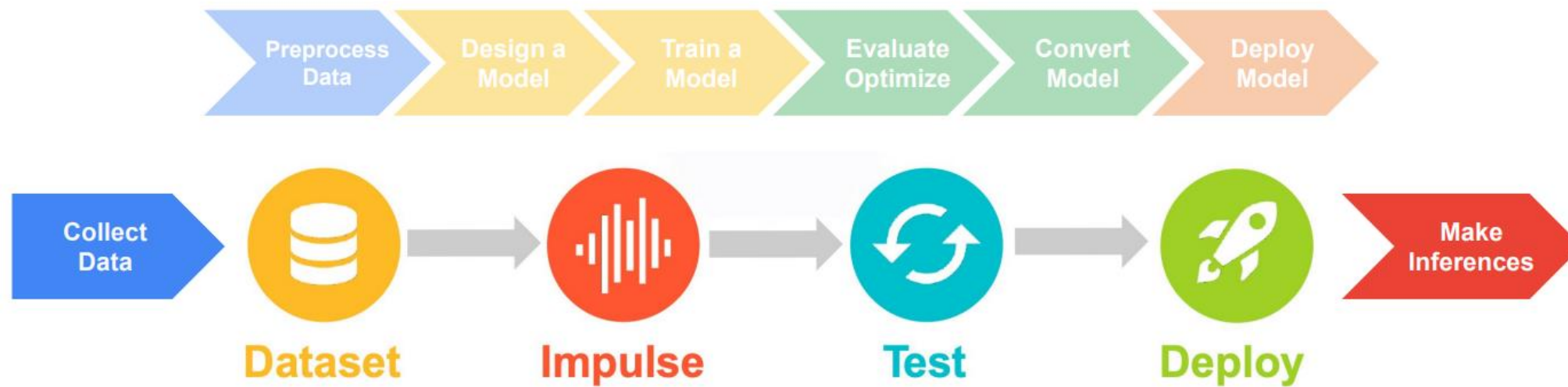


Edge Impulse

Machine Learning Workflow (“How”)



Edge Impulse



Edge Impulse

- Edge Impulse é uma plataforma para desenvolver e implantar modelos de aprendizado de máquina em dispositivos de ponta.
- Ele fornece um conjunto de ferramentas e serviços que facilitam a coleta de dados, o treinamento de modelos e a implantação deles em dispositivos de borda.
- Edge Impulse oferece suporte a uma ampla variedade de dispositivos de ponta, incluindo microcontroladores, sistemas embarcados e dispositivos móveis.

Edge Impulse

- O Edge Impulse é usado por uma ampla variedade de empresas e organizações.

SONY

OURA

TEXAS
INSTRUMENTS

NASA



Lexmark™

Inatel

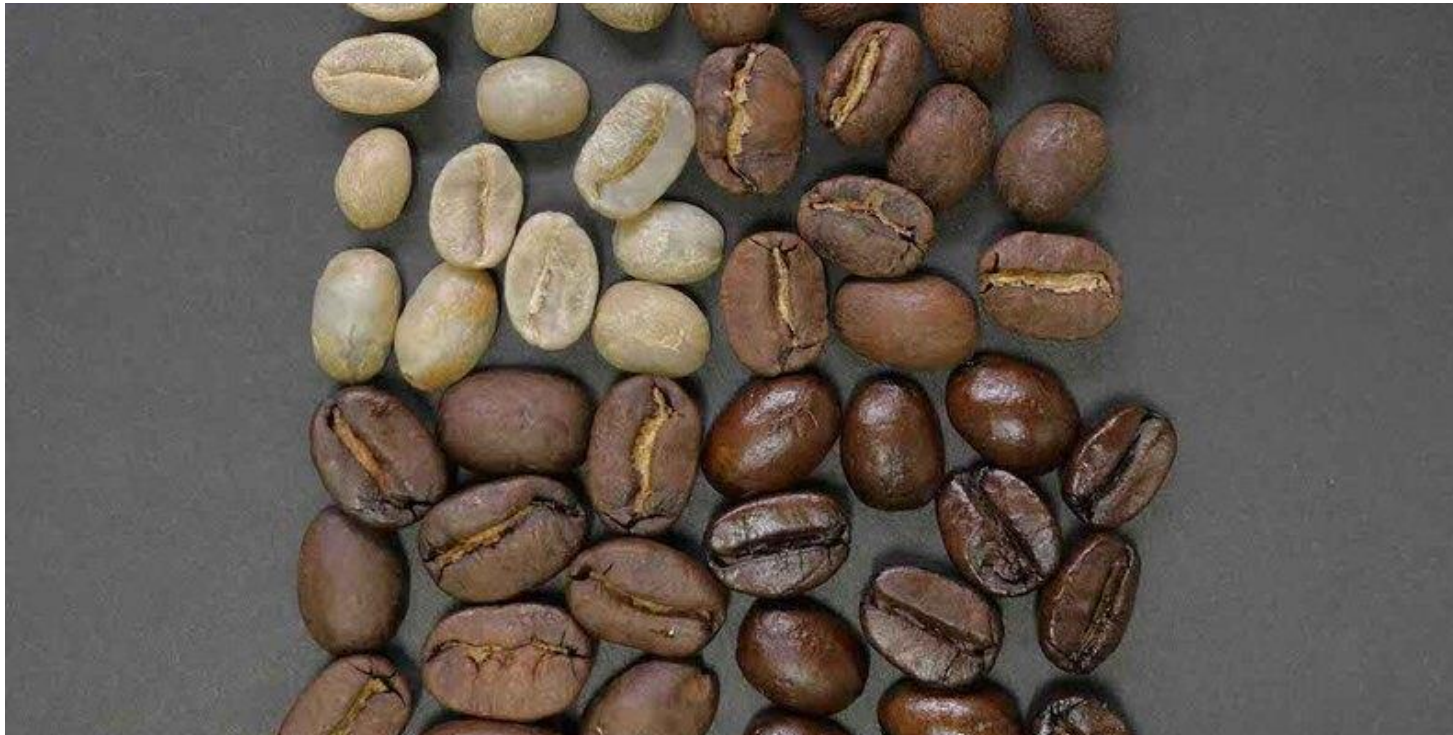
CAMINHOS
QUE CONECTAM
COM O FUTURO

Edge Impulse

- Fácil de usar: O Edge Impulse é fácil de usar, mesmo para desenvolvedores sem experiência anterior em aprendizado de máquina.
- Abrangente: O Edge Impulse fornece um conjunto abrangente de ferramentas e serviços para desenvolver e implantar modelos de aprendizado de máquina em dispositivos de borda.
- Flexível: O Edge Impulse oferece suporte a uma ampla variedade de dispositivos de borda e estruturas de aprendizado de máquina.
- Escalável: O Edge Impulse pode ser usado para desenvolver e implantar modelos de aprendizado de máquina em um grande número de dispositivos de borda.

Edge Impulse

- Exemplo guiado classificação do grão de café.
- <https://edgeimpulse.com/>
- Dataset: <https://www.kaggle.com/datasets/gpiosenka/coffee-bean-dataset-resized-224-x-224>



Edge Impulse

- Tarefa aula
- Escolher um dataset no kaggle e fazer um treinamento no edge impulse

Edge Impulse

- Trabalho
- Criar um dataset de imagens com o celular e fazer um treinamento no edge impulse