

Image Classification using CNN – Edge Impulse Introduction

Cris Ababei



BE THE DIFFERENCE.

1

1

Embedded Machine Learning (TinyML) Workflow Review

2

2

Machine Learning Workflow



3

3

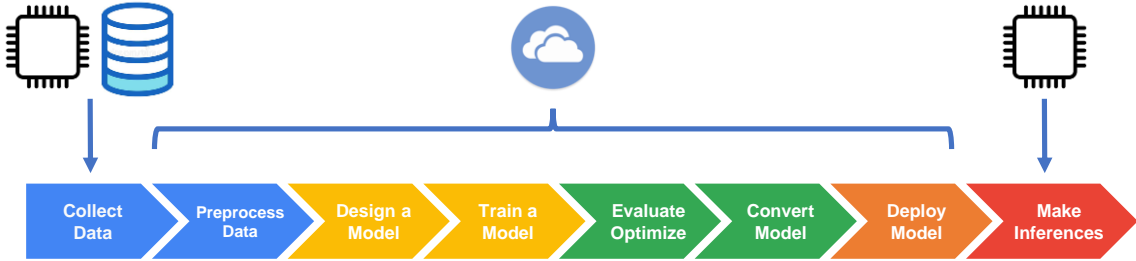
Tiny Machine Learning Workflow (“What”)



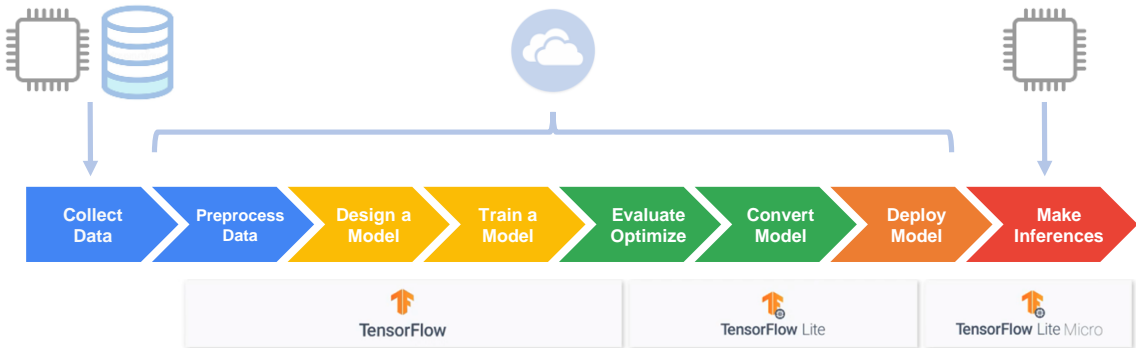
4

4

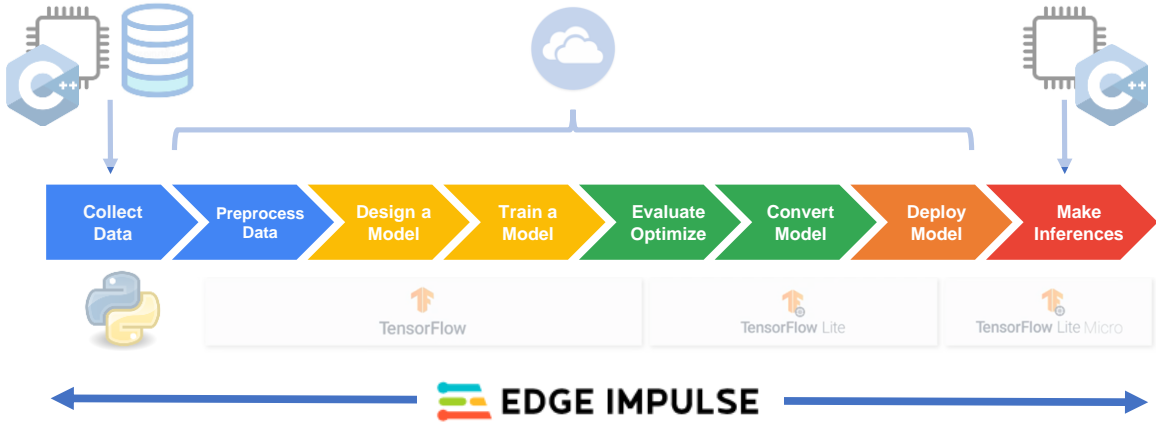
Tiny Machine Learning Workflow (“Where”)



Machine Learning Workflow (“How”)

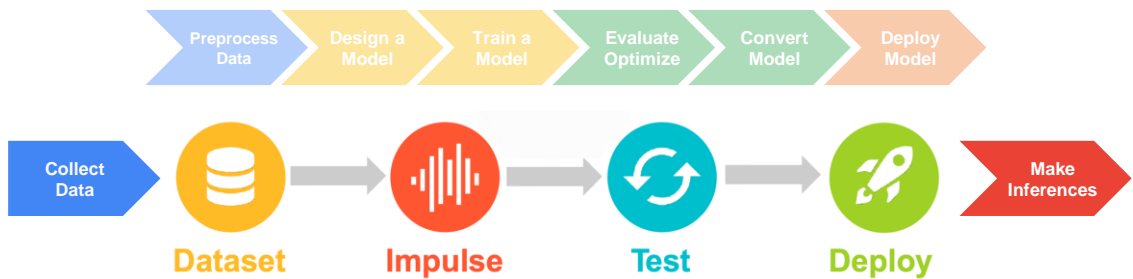


Machine Learning Workflow ("How")



7

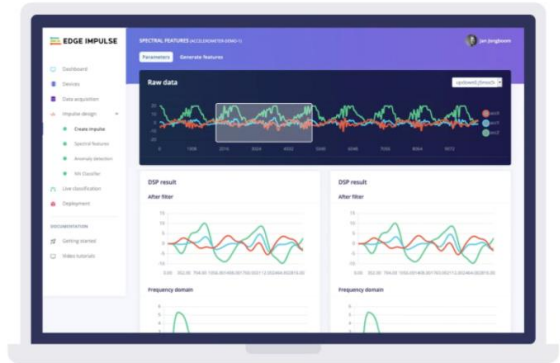
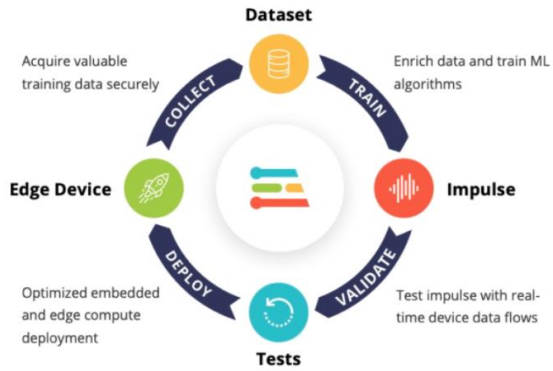
7



8

8

EI Studio - Embedded ML platform (“AutoML”)



Learn more at <http://edgeimpulse.com>



8 Copyright © EdgeImpulse Inc.

9

Cifar10 Edge Impulse Studio



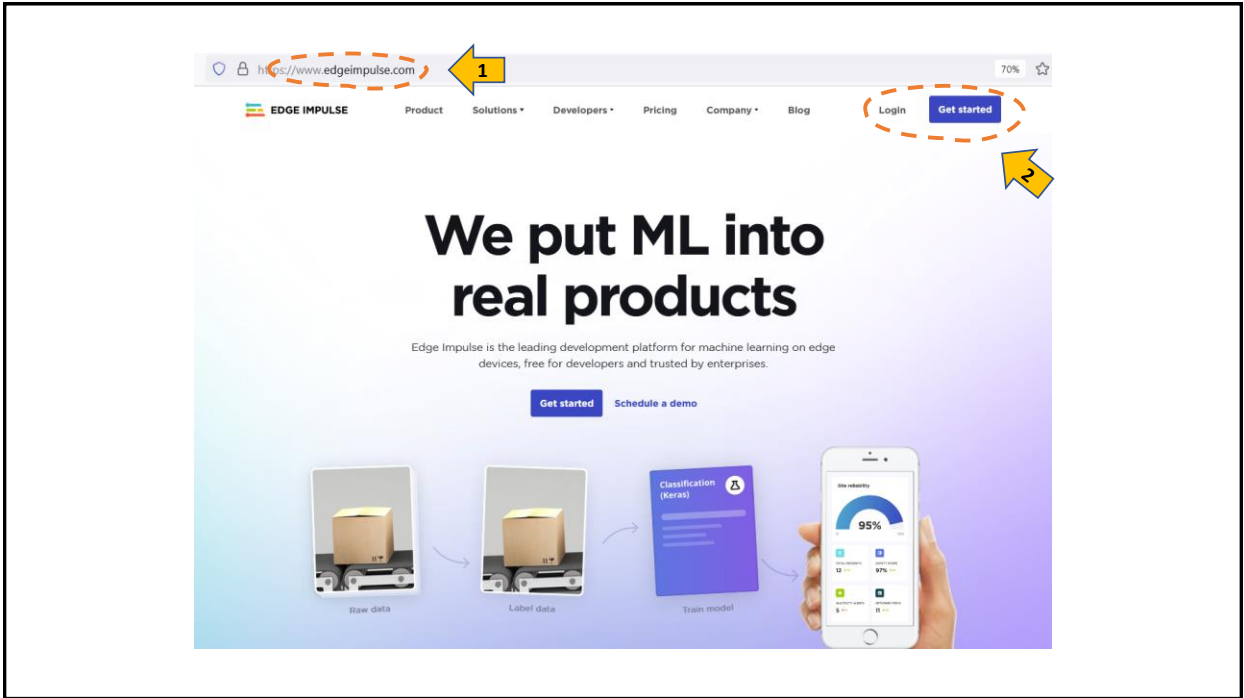
Dataset: <https://github.com/YoongiKim/CIFAR-10-images>

EI Studio Public Projects:

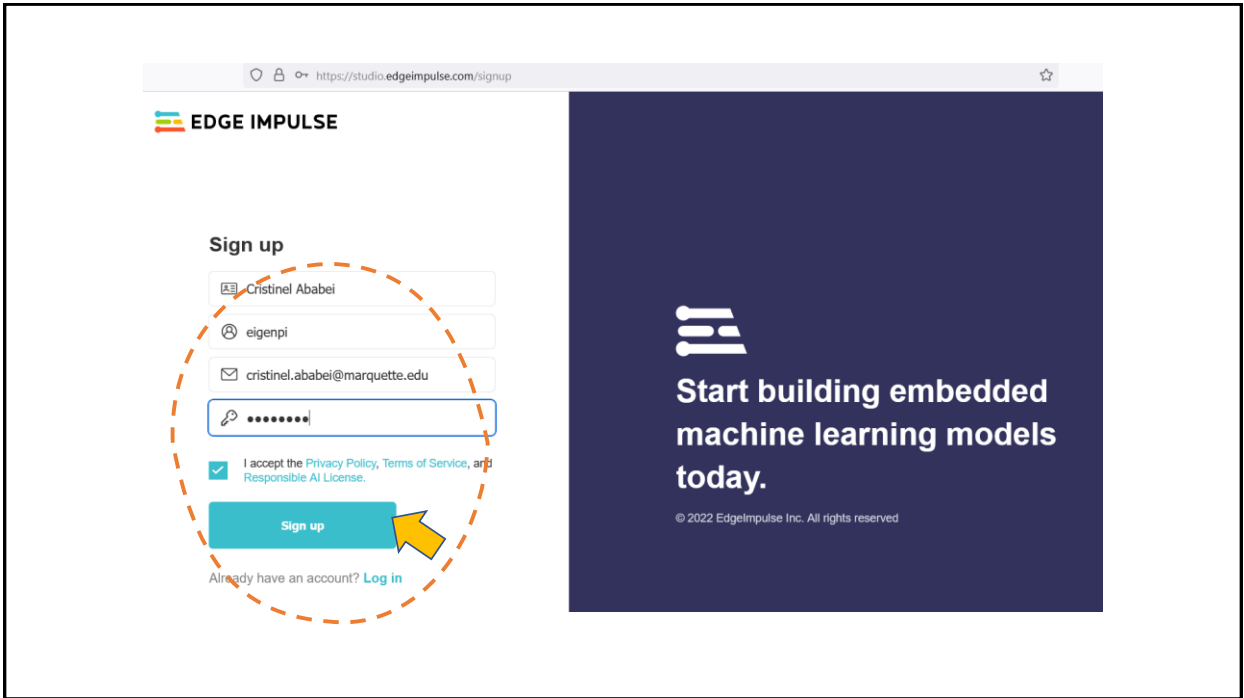
<https://studio.edgeimpulse.com/public/165504/latest>

<https://studio.edgeimpulse.com/public/51070/latest>

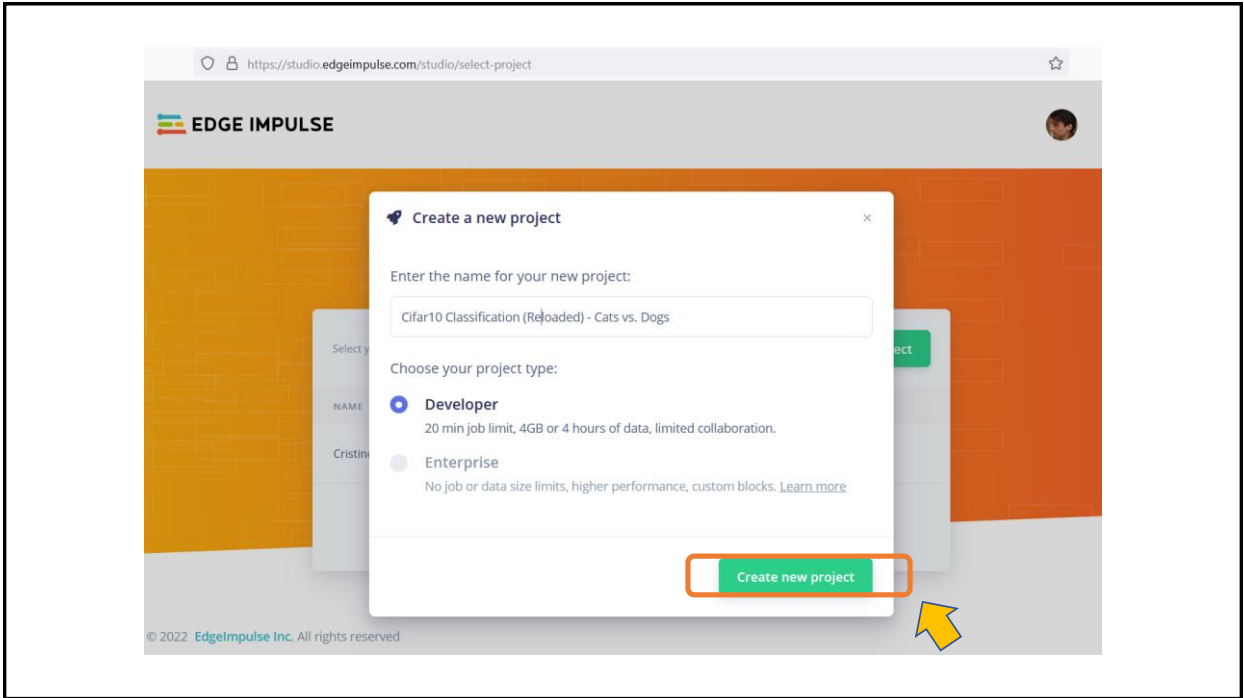
10



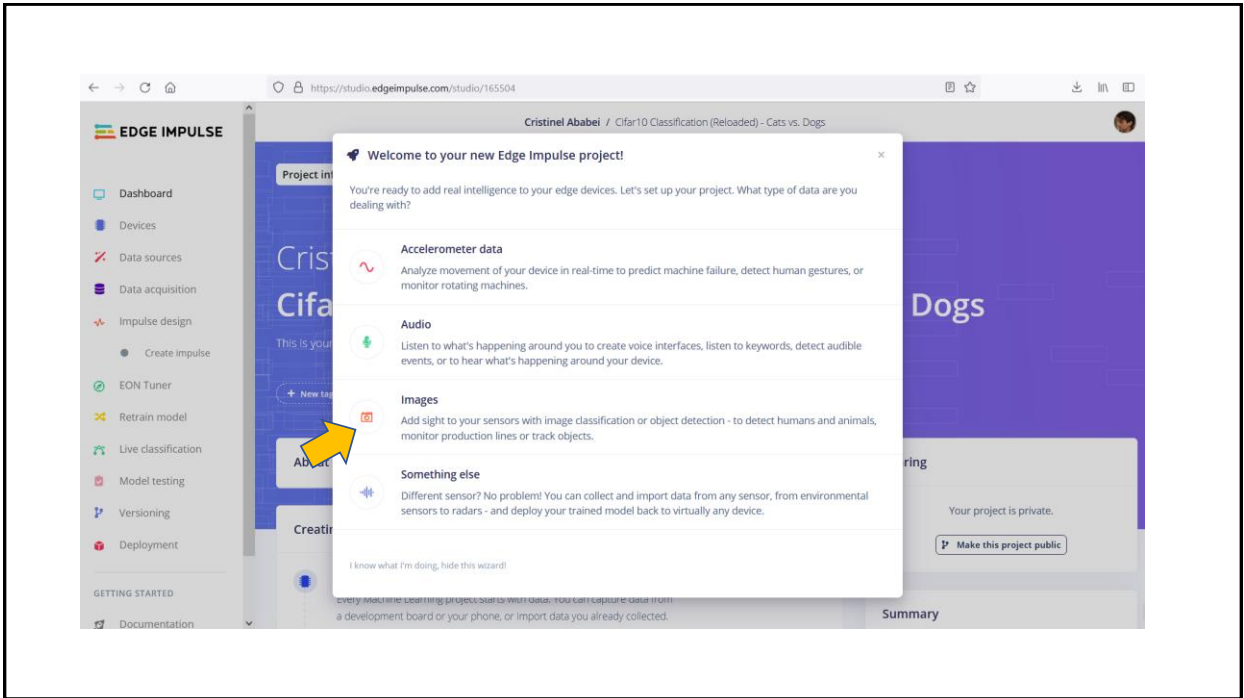
11



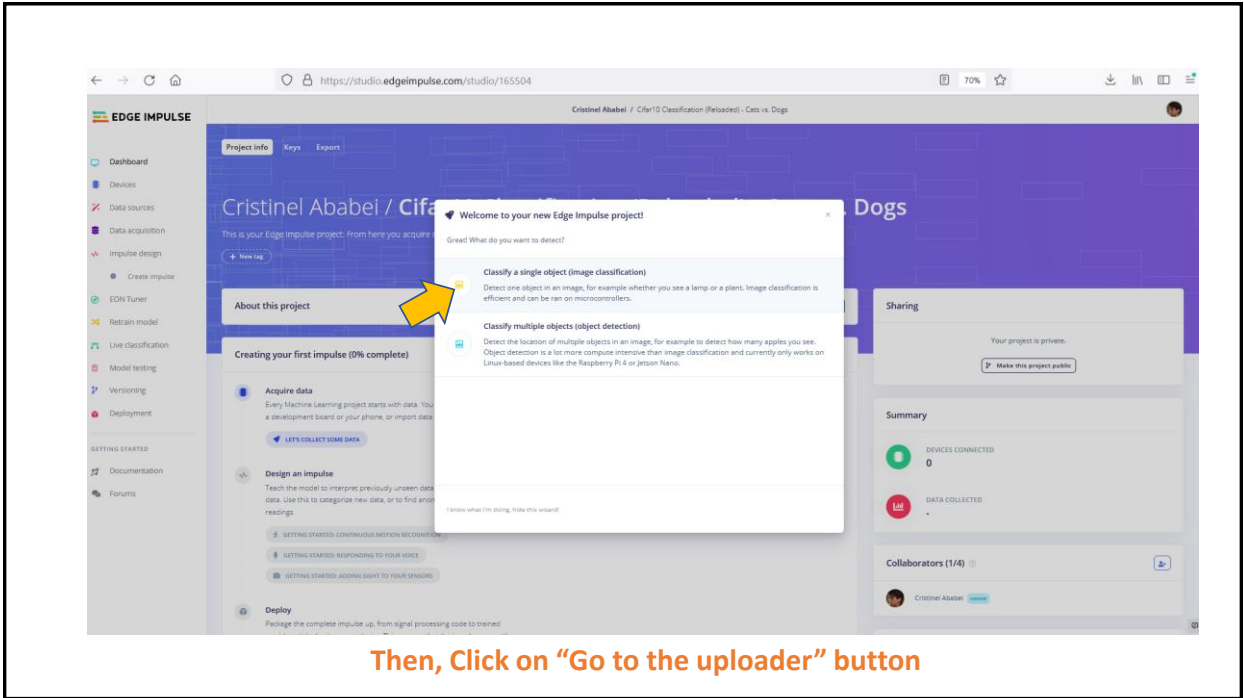
12



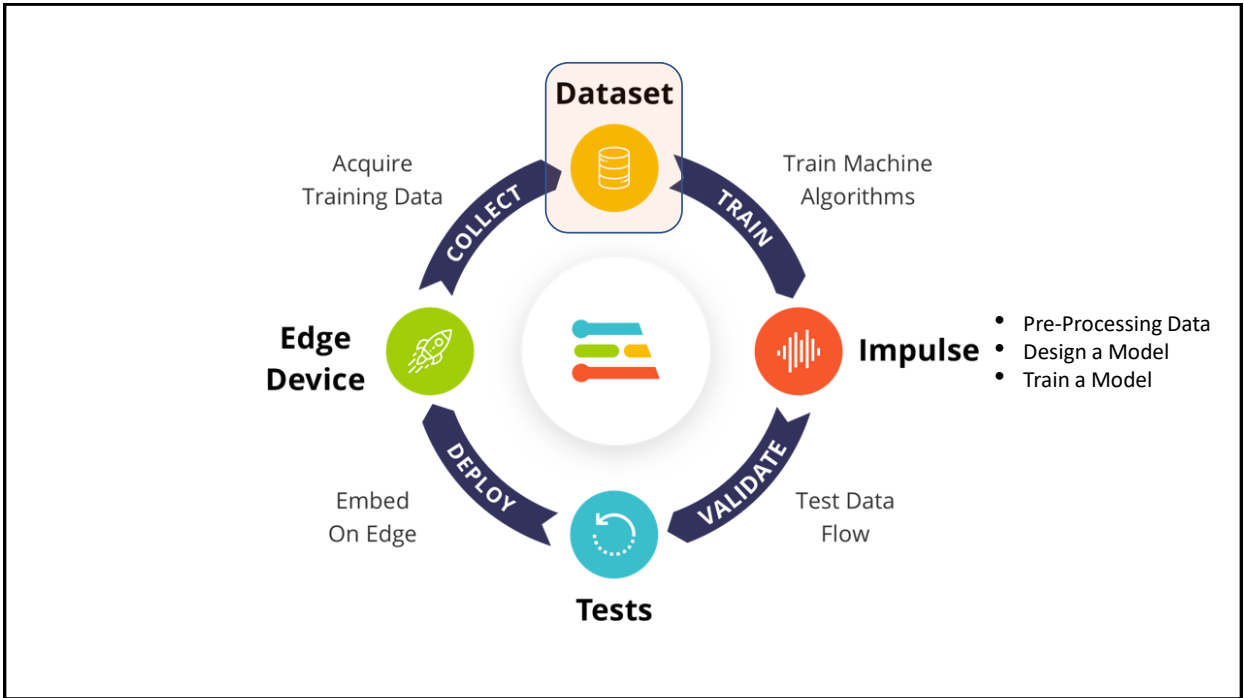
13



14



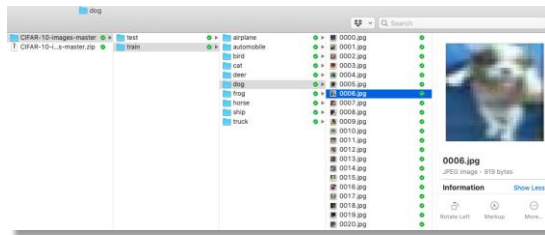
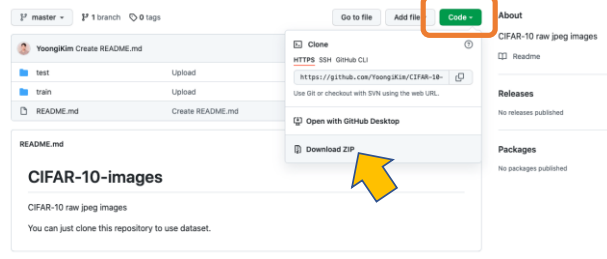
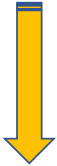
15



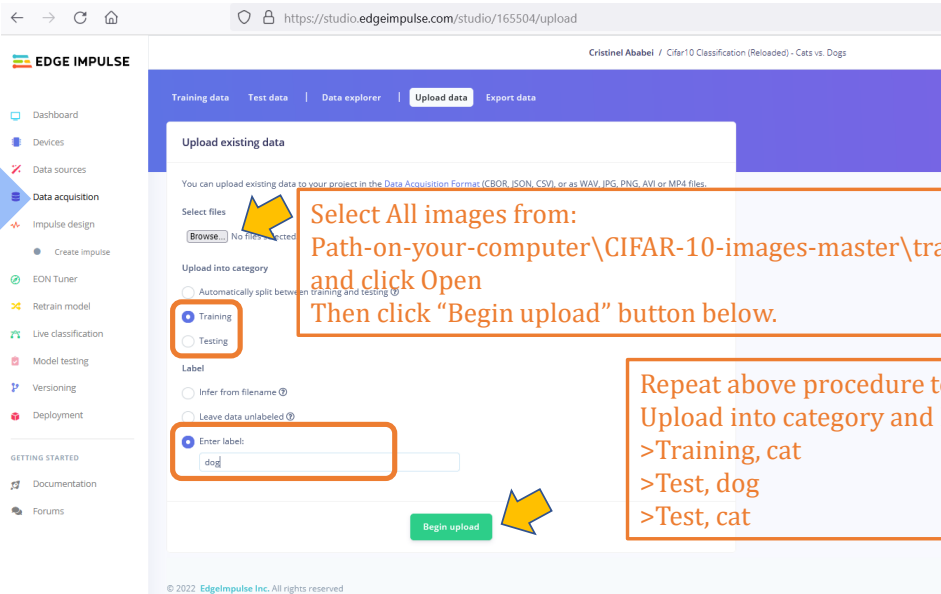
16

Download Dataset

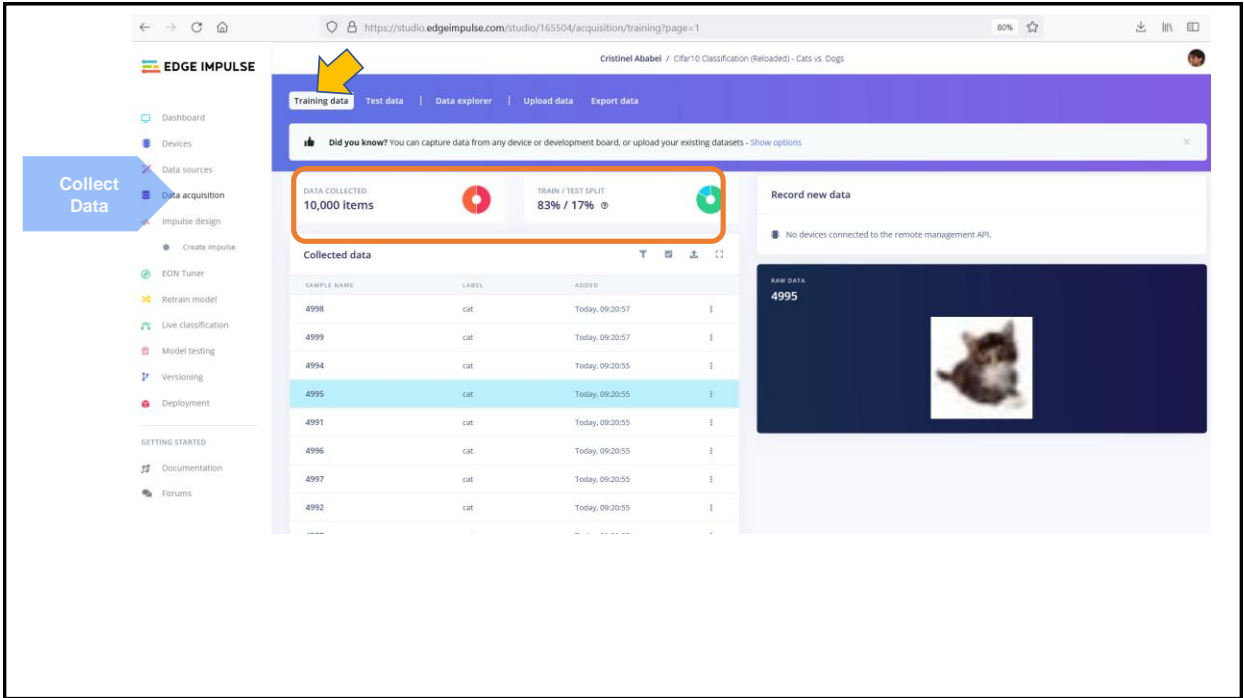
<https://github.com/YoongiKim/CIFAR-10-images>



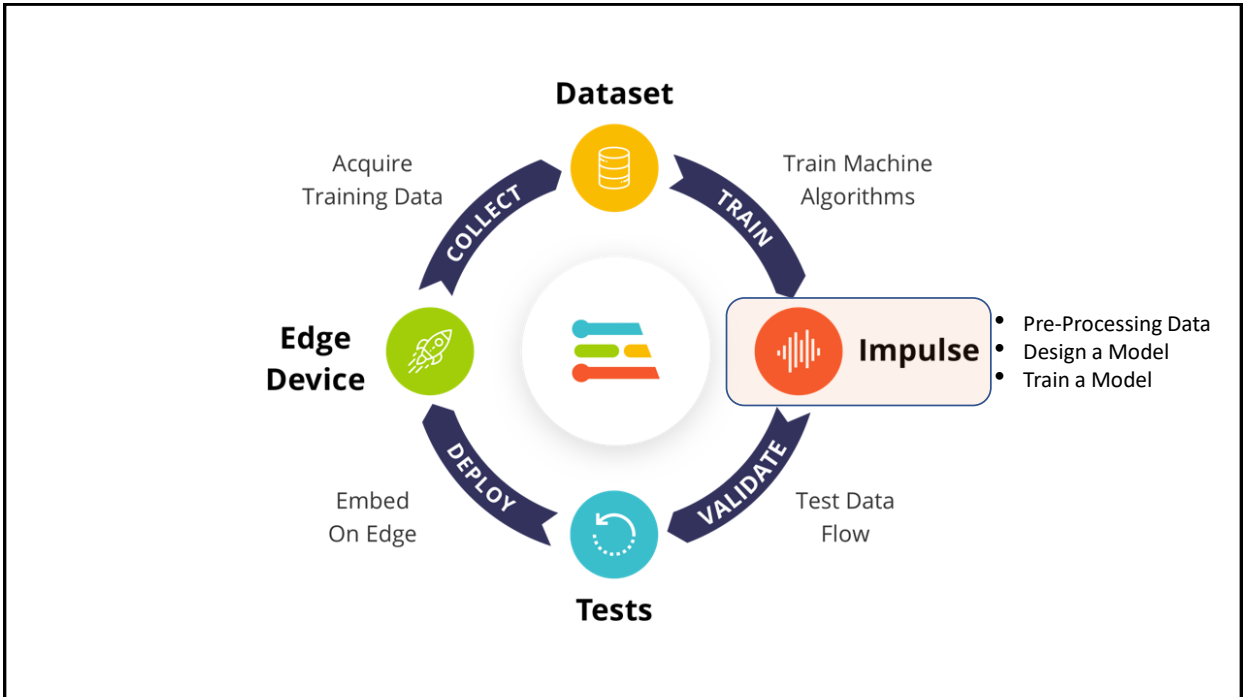
17



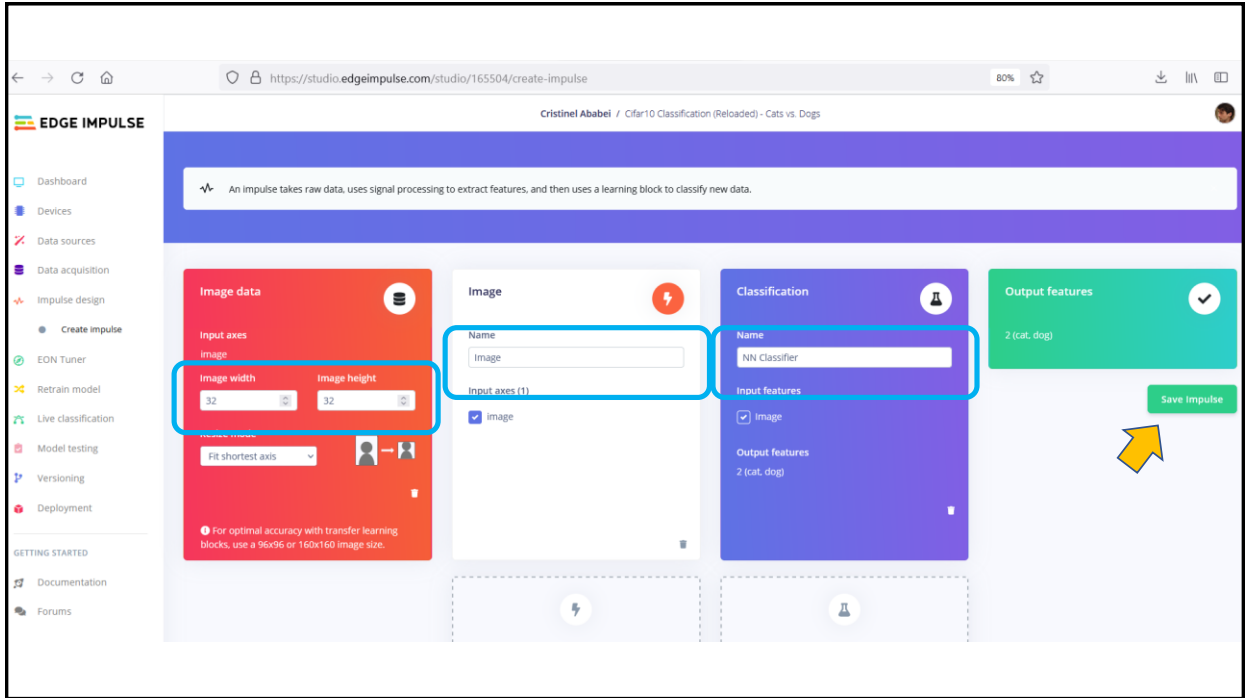
18



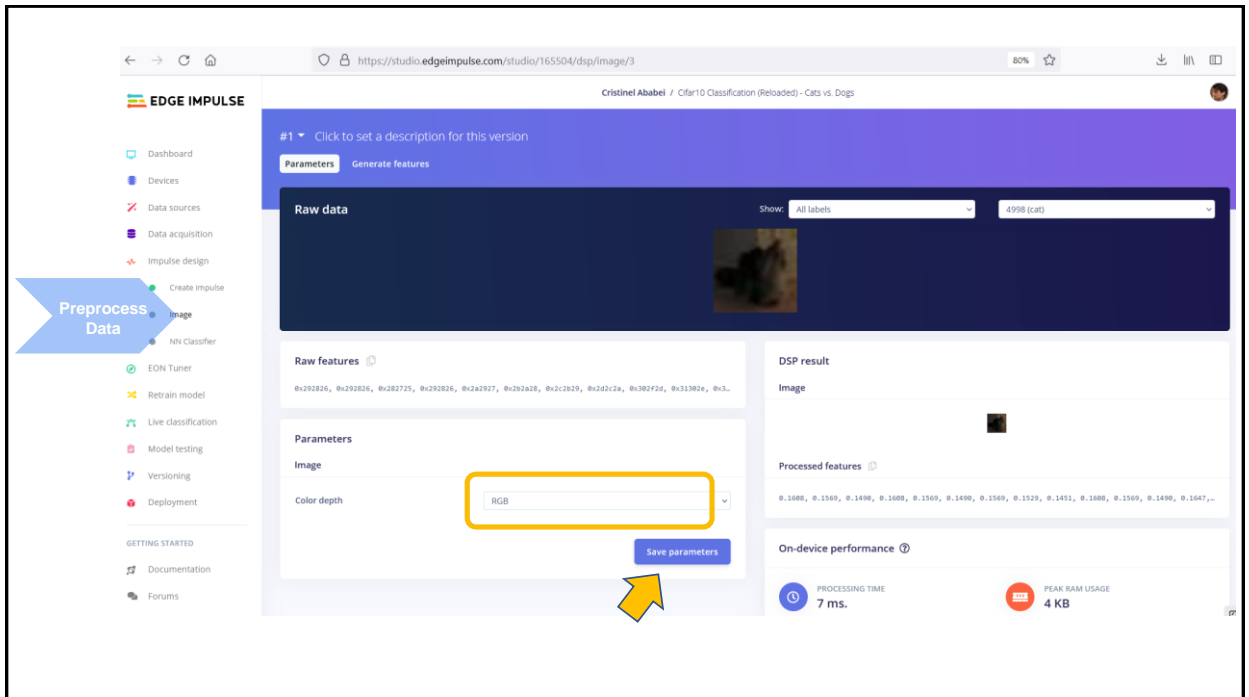
19



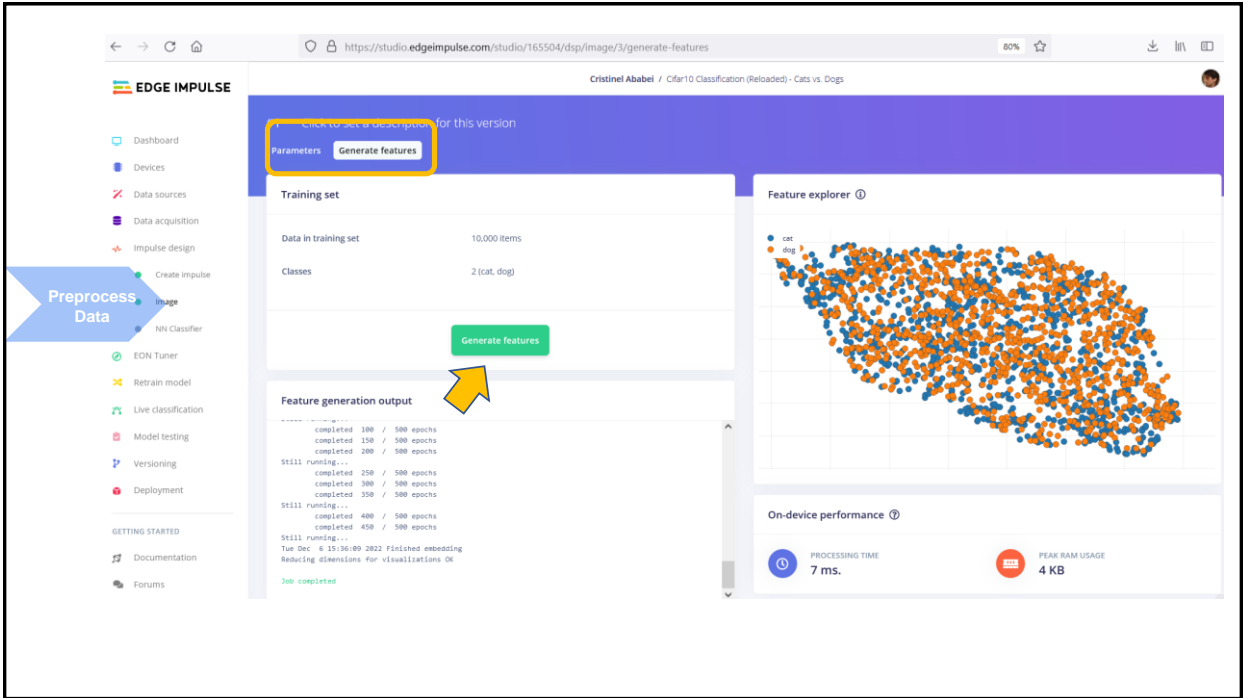
20



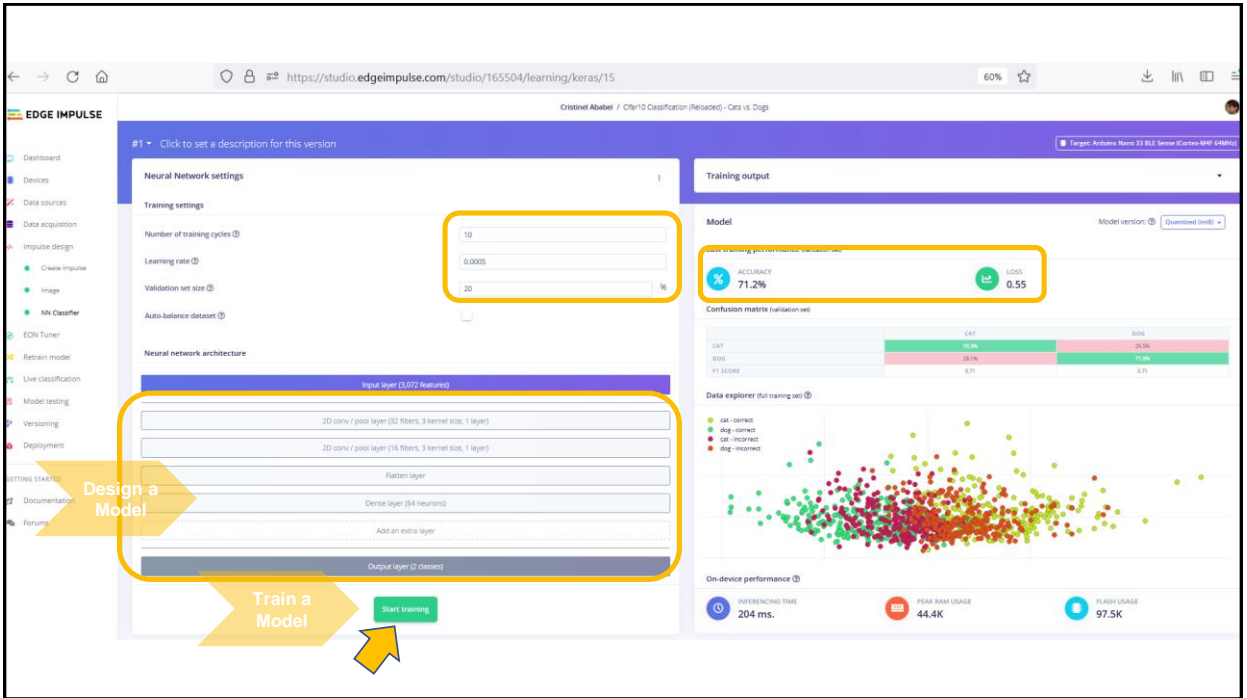
21



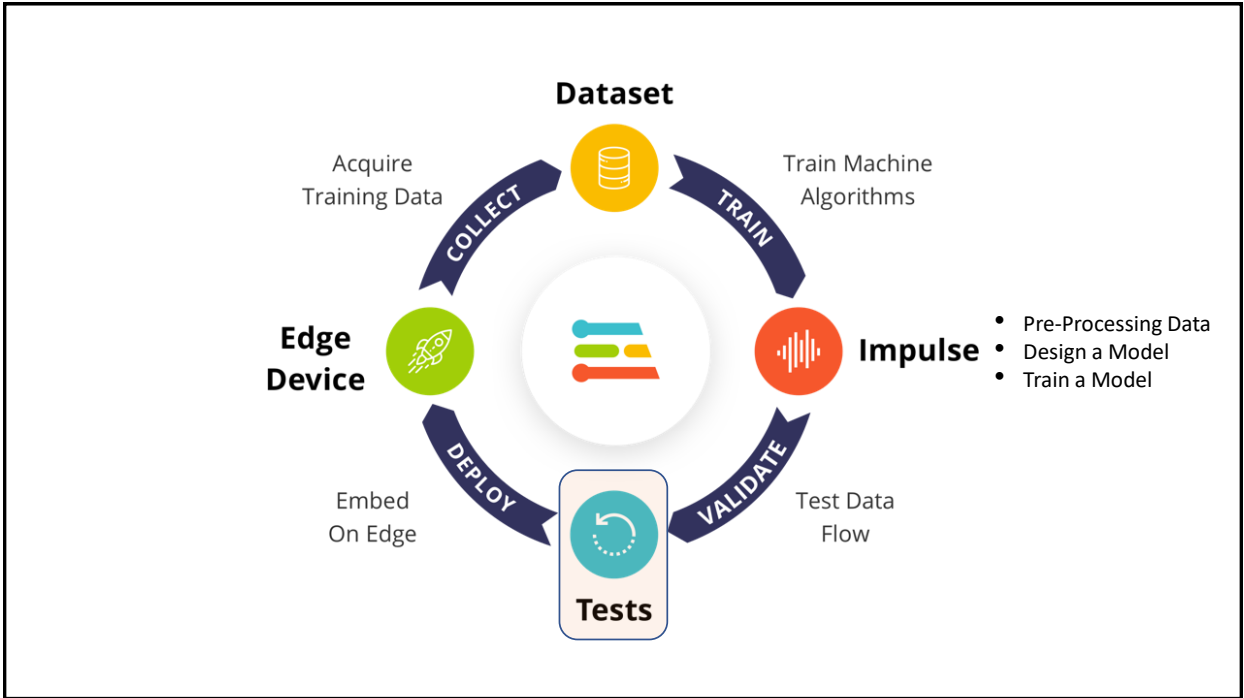
22



23



24



25

EDGE IMPULSE

Dashboard

Devices

Data sources

Data acquisition

Impulse design

Create impulse

Image

NN Classifier

EOIn Tuner

Retrain model

Use classification

Model testing

Deployment

GETTING STARTED

Documentation

Forums

Test data

This lists all test data. You can manage this data through Data acquisition.

Set the "expected outcome" for each sample to the desired outcome to automatically score the impulse.

SAMPLE NAME	EXPECTED OUTCOME	LENGTH	ACCURACY	RESULT
0987	cat	-	100%	1 cat
0996	cat	-	100%	1 cat
0997	cat	-	100%	1 cat
0999	cat	-	0%	1 uncertain
0993	cat	-	100%	1 cat
0994	cat	-	0%	1 uncertain
0995	cat	-	0%	1 dog
0998	cat	-	100%	1 cat
0992	cat	-	100%	1 cat
0990	cat	-	100%	1 cat
0989	cat	-	100%	1 cat
0991	cat	-	100%	1 cat
0988	cat	-	0%	1 uncertain
0985	cat	-	0%	1 dog
0984	cat	-	0%	1 dog
0983	cat	-	0%	1 uncertain
0986	cat	-	0%	1 uncertain

Model testing output

Model testing results

ACCURACY 61.75%

	CAT	DOG	UNCERTAIN
CAT	88.8%	16.1%	14.1%
DOG	21%	85.7%	16.1%
UNCERTAIN	0%	0%	0%

Feature explorer

- Get selected
- Get correct
- Get incorrect
- Get uncertain

DATA

Label: dog

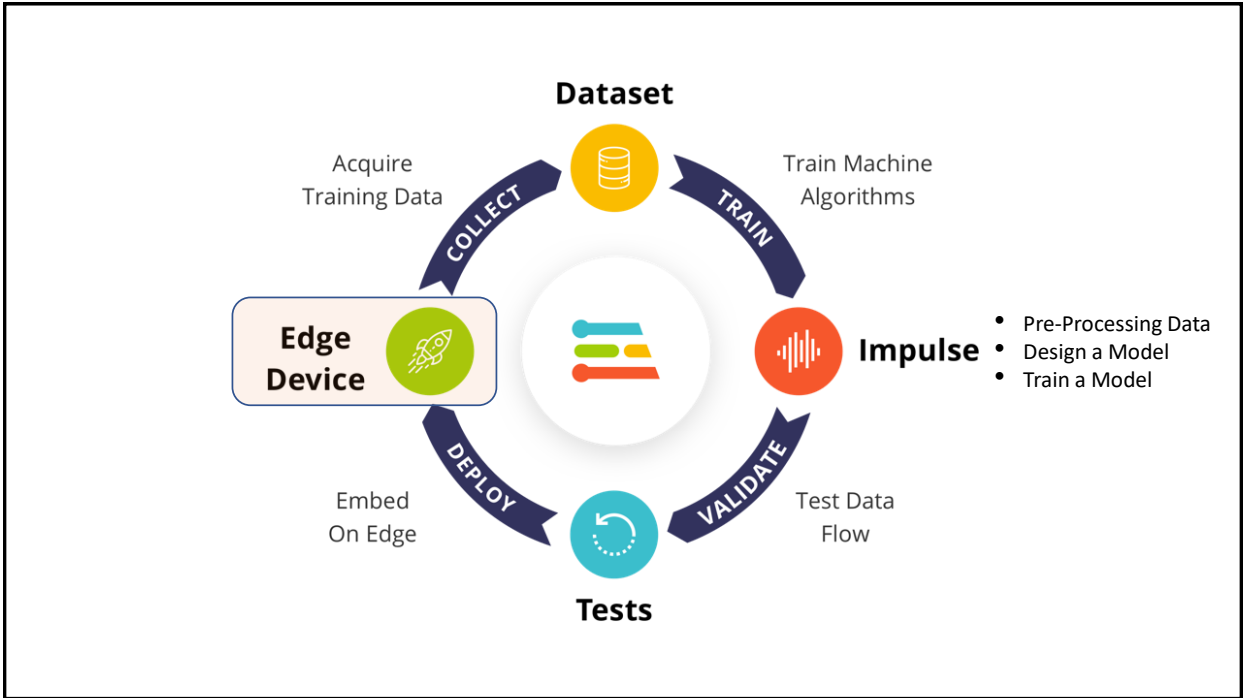
Predicted: cat

View details

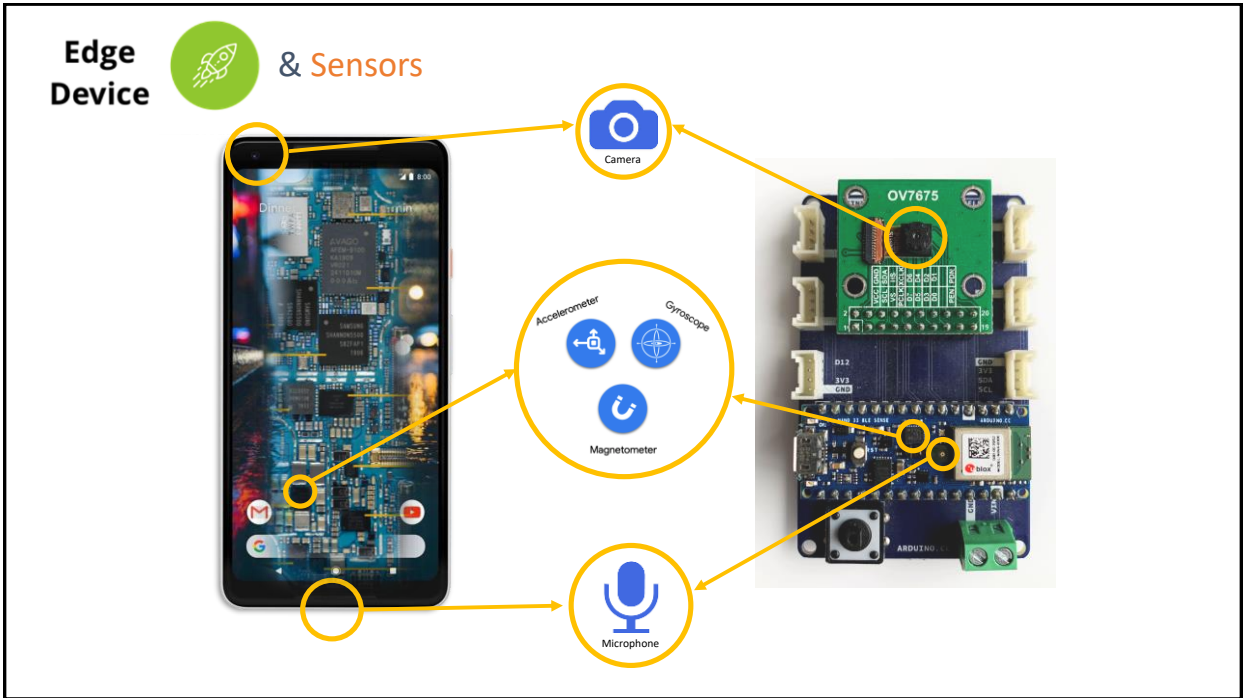
View classification

Evaluate Optimize

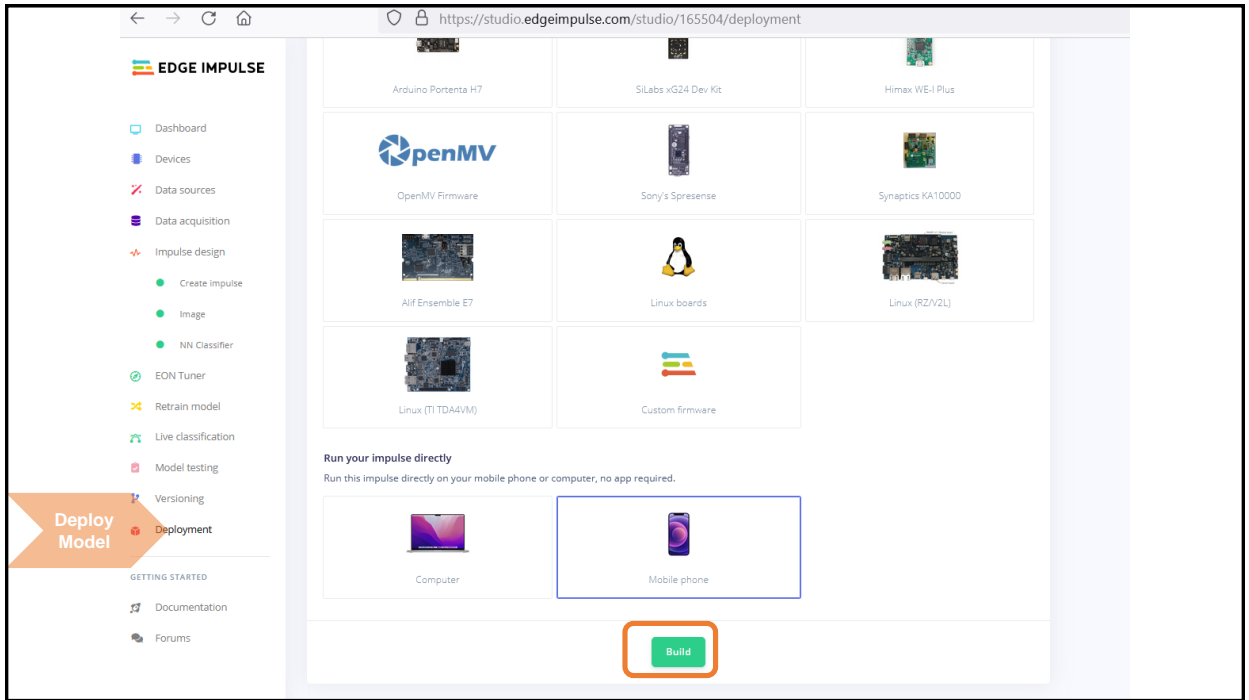
26



27



28

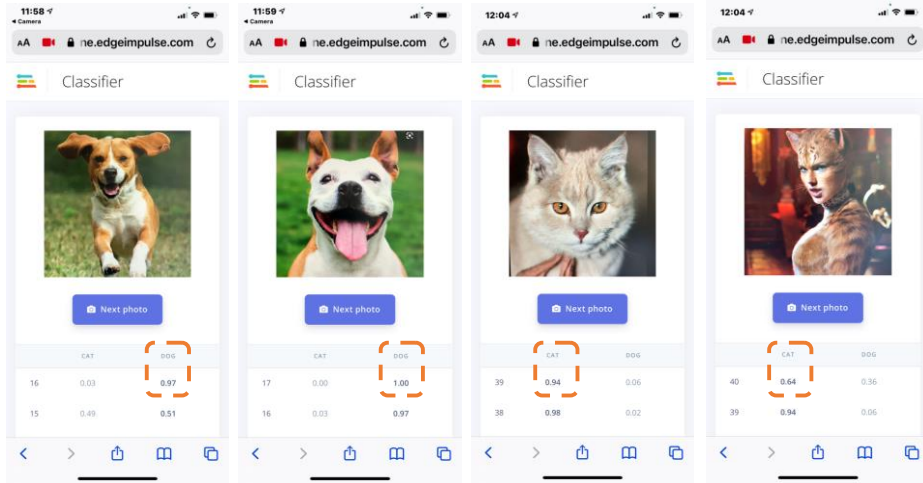


29



30

Make Inferences



31

El Studio “under the hood” Code Time!

[ei-cifar10_image_classification-nn-classifier-explained.ipynb](#)



32

32

EDGE IMPULSE

Crissinel Ababei / CIFAR10 Classification (Revised) - Cats vs Dogs

#1 - Click to set a description for this version

Neural Network settings

Training settings

Validation set size 20

Neural network architecture

```

1 import tensorflow as tf
2 from tensorflow.keras.models import Sequential
3 from tensorflow.keras.layers import Dense, InputLayer, Dropout, Conv2D, Conv1D, Flatten, Reshape, ReLU, MaxPooling2D, MaxPooling1D, BatchNormalization,
4   Lambda, Concatenate, ReLU, Softmax
5 from tensorflow.keras.optimizers import Adam
6 EPOCHS = 100 # number of epochs
7 # This is the path to the data folder, or you can manipulate the tf.data.Dataset objects yourself
8 DATA_SIZE = 10
9 train_dataset = tf.data.Dataset.from_tensor_slices((train_images, train_labels))
10 validation_dataset = tf.data.Dataset.from_tensor_slices((val_images, val_labels))
11 # model architecture
12 model = Sequential()
13 model.add(Conv2D(32, kernel_size=3, kernel_constraint=tf.keras.constraints.MaxNorm(1), padding='same', activation='relu'))
14 model.add(MaxPooling2D(kernel_size=2, activation='padding'))
15 model.add(Conv2D(64, kernel_size=3, kernel_constraint=tf.keras.constraints.MaxNorm(1), padding='same', activation='relu'))
16 model.add(MaxPooling2D(kernel_size=2, activation='padding'))
17 model.add(Flatten())
18 model.add(Dense(100, activation='relu'))
19 model.add(Dense(10, activation='softmax'))
20 # compile the model
21 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
22 # train the model
23 opt = AdamLearningRateScheduler(lambda epoch: 1e-08)
24 validation_data = validation_dataset.as_numpy_iterator()
25 model.fit(train_dataset.as_numpy_iterator(), validation_data, epochs=EPOCHS)
26 # use this flag to disable per-channel quantization for a model
27 # This can reduce the size for quantized models, but may hurt
28 # on-device accuracy
29 disable_per_channel_quantization = False

```

Start training

Training output

Model

Last training performance (validation set)

ACCURACY 71.2%

LOSS 0.56

Confusion matrix (validation set)

	CAT	DOG
CAT	10.0%	30.2%
DOG	1.0%	18.8%
F1 SCORE	0.71	0.71

Data explorer (full training set)

- Get correct
- Get incorrect
- Get incorrect
- Get incorrect

On-device performance

INFERENCE TIME 234 ms

PEAK RAM USAGE 44.4K

FLASH USAGE 97.5K

33

EDGE IMPULSE

Crissinel Ababei / CIFAR10 Classification (Revised) - Cats vs Dogs

#1 - Click to set a description for this version

Neural Network settings

Training settings

Validation set size 20

Neural network architecture

```

1 import tensorflow as tf
2 from tensorflow.keras.models import Sequential
3 from tensorflow.keras.layers import Dense, InputLayer, Dropout, Conv2D, Conv1D, Flatten, Reshape, ReLU, MaxPooling2D, MaxPooling1D, BatchNormalization,
4   Lambda, Concatenate, ReLU, Softmax
5 from tensorflow.keras.optimizers import Adam
6 EPOCHS = 100 # number of epochs
7 # This is the path to the data folder, or you can manipulate the tf.data.Dataset objects yourself
8 DATA_SIZE = 10
9 train_dataset = tf.data.Dataset.from_tensor_slices((train_images, train_labels))
10 validation_dataset = tf.data.Dataset.from_tensor_slices((val_images, val_labels))
11 # model architecture
12 model = Sequential()
13 model.add(Conv2D(32, kernel_size=3, kernel_constraint=tf.keras.constraints.MaxNorm(1), padding='same', activation='relu'))
14 model.add(MaxPooling2D(kernel_size=2, activation='padding'))
15 model.add(Conv2D(64, kernel_size=3, kernel_constraint=tf.keras.constraints.MaxNorm(1), padding='same', activation='relu'))
16 model.add(MaxPooling2D(kernel_size=2, activation='padding'))
17 model.add(Flatten())
18 model.add(Dense(100, activation='relu'))
19 model.add(Dense(10, activation='softmax'))
20 # compile the model
21 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
22 # train the model
23 opt = AdamLearningRateScheduler(lambda epoch: 1e-08)
24 validation_data = validation_dataset.as_numpy_iterator()
25 model.fit(train_dataset.as_numpy_iterator(), validation_data, epochs=EPOCHS)
26 # use this flag to disable per-channel quantization for a model
27 # This can reduce the size for quantized models, but may hurt
28 # on-device accuracy
29 disable_per_channel_quantization = False

```

Start training

Training output

Model

Last training performance (validation set)

ACCURACY 71.2%

LOSS 0.56

Confusion matrix (validation set)

	CAT	DOG
CAT	10.0%	30.2%
DOG	1.0%	18.8%
F1 SCORE	0.71	0.71

Data explorer (full training set)

- Get correct
- Get correct
- Get incorrect
- Get incorrect

On-device performance

INFERENCE TIME 234 ms

PEAK RAM USAGE 44.4K

FLASH USAGE 97.5K

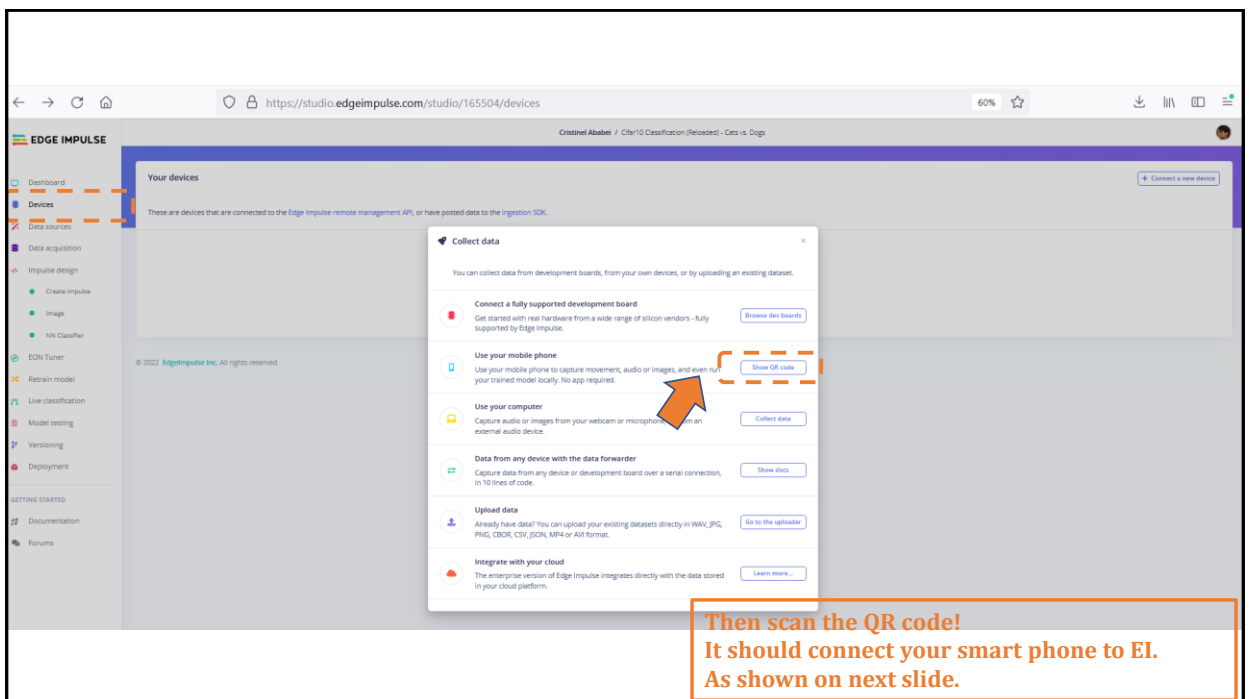
Click on Edit block locally.
A .zip file will be download on your computer.
Open on your computer and study it!!!

34

Optional Connect device for data capture!

35

35



The screenshot shows the Edge Impulse Studio web interface. The main content area is titled "Your devices" and contains a "Collect data" modal. The modal lists several options for data collection:

- Connect a fully supported development board**: Get started with real hardware from a wide range of silicon vendors - fully supported by Edge Impulse. (Button: Browse dev boards)
- Use your mobile phone**: Use your mobile phone to capture movement, audio or images, and even for your trained model locally. No app required. (Button: Show QR code)
- Use your computer**: Capture audio or images from your webcam or microphone, or an external audio device. (Button: Collect data)
- Data from any device with the data forwarder**: Capture data from any device or development board over a serial connection, in 10 lines of code. (Button: Show docs)
- Upload data**: Already have data? You can upload your existing datasets directly in WAV, JPG, PNG, CBOR, CSV, JSON, MP4 or AVI format. (Button: Go to the uploader)
- Integrate with your cloud**: The enterprise version of Edge Impulse integrates directly with the data stored in your cloud platform. (Button: Learn more...)

A red dashed box highlights the "Use your mobile phone" option, and an orange arrow points to the "Show QR code" button. A text box at the bottom right of the slide contains the following text:

**Then scan the QR code!
It should connect your smart phone to EI.
As shown on next slide.**

36

The screenshot shows the Edge Impulse Studio web interface. A dialog box titled "Collect data" is displayed, indicating that the device "phone_la5wlu6t" is now connected. A yellow arrow points to the "Get started!" button in the dialog. To the right, a mobile app interface is shown with a "Data collection" screen, featuring a green checkmark and options for "Collecting images?", "Collecting audio?", and "Collecting motion?".

Click on Get started in EI project.
On your phone click "Collecting images?"
Take snapshot of images of dogs and cats with phone.
Photos will be added to your dataset within EI project: to label "dog" or "cat" into category train or test, as you set it within the phone app interface!
As shown on next slide...

37

The three screenshots illustrate the mobile app's data collection workflow. The first screenshot shows the "Data collection" screen with a green checkmark and the text "Connected as phone_l2rkzyb0". The second screenshot shows a "Permission required" dialog with a camera icon and a "Give access to the camera" button. The third screenshot shows the "Data collection" screen with a "Label: cat" and "Category: Training" dropdown, a photo of a cat, and a "Capture" button.

38

Credits

- A previous edition of this course was developed in collaboration with Dr. Susan C. Schneider of Marquette University.
- We are very grateful and thank all the following professors, researchers, and practitioners for jump-starting courses on TinyML and for sharing their teaching materials:
 - Prof. Marcelo Rovai - TinyML - Machine Learning for Embedding Devices, UNIFEI
 - <https://github.com/Mjrovai/UNIFEI-IESTI01-TinyML-2022.1>
 - Prof. Vijay Janapa Reddi - CS249r: Tiny Machine Learning, Applied Machine Learning on Embedded IoT Devices, Harvard
 - <https://sites.google.com/g.harvard.edu/tinyml/home>
 - Prof. Rahul Mangharam – ESE3600: Tiny Machine Learning, Univ. of Pennsylvania
 - <https://tinyml.seas.upenn.edu/#>
 - Prof. Brian Plancher - Harvard CS249r: Tiny Machine Learning (TinyML), Barnard College, Columbia University
 - https://a2r-lab.org/courses/cs249r_tinyml/

39

39

References

- Additional references from where information and other teaching materials were gathered include:
 - Applications & Deploy textbook: “TinyML” by Pete Warden, Daniel Situnayake
 - <https://www.oreilly.com/library/view/tinyml/9781492052036/>
 - Deploy textbook “TinyML Cookbook” by Gian Marco Iodice
 - <https://github.com/PacktPublishing/TinyML-Cookbook>
 - Jason Brownlee
 - <https://machinelearningmastery.com/>
 - TinyMLedu
 - <https://tinyml.seas.harvard.edu/>
 - Professional Certificate in Tiny Machine Learning (TinyML) – edX/Harvard
 - <https://www.edx.org/professional-certificate/harvardx-tiny-machine-learning>
 - Introduction to Embedded Machine Learning - Coursera/Edge Impulse
 - <https://www.coursera.org/learn/introduction-to-embedded-machine-learning>
 - Computer Vision with Embedded Machine Learning - Coursera/Edge Impulse
 - <https://www.coursera.org/learn/computer-vision-with-embedded-machine-learning>

40

40