Convolutions for Hands-on Computer Vision

Brian Plancher
Harvard John A. Paulson School of Engineering and Applied Sciences
Barnard College, Columbia University
brianplancher.com
Quick Disclaimer:
Today will be both too fast and too slow!
Quick Disclaimer:

Today will be both too fast and too slow!
We will explore the science behind computer vision and collect data and train our own custom model to recognize objects using Edge Impulse.
Today’s Agenda

- Introduction to Computer Vision
- Hands-on Computer Vision: Thing Translator
- Building an Object Detection Dataset
- Training our Model using Transfer Learning
- Deploying our Model onto our Arduino
- Summary
Today’s Agenda

- Introduction to Computer Vision
- Hands-on Computer Vision: Thing Translator
- Building an Object Detection Dataset
- Training our Model using Transfer Learning
- Deploying our Model onto our Arduino
- Summary
Machine Learning

Inputs

We provide Answers
Aka the Labels of the Data

The Computer Learns

Rules
Let’s try to figure out **what** she’s doing?

- **walking**
  - 01010101001000110101
  - 01010100101001001010
  - 10101011010100100000

- **running**
  - 11110101001001010101
  - 01010010100101010100
  - 11010110010101001111

- **biking**
  - 00001111010110010010
  - 01010111101011010101
  - 11010111111001001111

- **golfing**
  - 01111110101110101010
  - 01011110101011010101
  - 11111111110010001110
Let’s try to figure out what she’s doing?

- **walking**: 0101010100100110101 010100101001001010 1010110101001001111
- **running**: 1111010100100101011 010010010010100 11010110010101001111
- **biking**: 00001110101110101101 01011110101101010101 11010111111010010111
- **golfing**: 01111110101110101010 10101111010101101011 01111110101110110110010001110
What is a neural network?
What is a neural network?
What is a neural network?
Training the machine

For a set of Input Data
Training the machine

For a set of Input Data → Guess the Answer and count mistakes
Training the machine

For a set of Input Data → Guess the Answer and count mistakes → Improve the model to be more correct
Training the machine

For a set of Input Data → Guess the Answer and count mistakes → Improve the model to be more correct

16
After it’s **learned** use it for **inference**:
What is a neural network?

To learn more about the math behind neural network training there is a nice series of videos here: 3Blue1Brown Neural Networks Playlist
Computer Vision is Hard
Computer Vision is Hard

What color are the pants and the shirt?

Slide Credit: Hamilton Chong
Computer Vision is Hard
Computer Vision is Hard

Slide Credit: Hamilton Chong
Computer Vision is Hard

Is square A or B darker in color?
Computer Vision is Hard

Areas of the image A and B are the same color

A rectangle of the same color has been drawn connecting the two areas of the image
What **Features** of the image might be important for self driving cars?
What **Features** of the image might be important for self driving cars?

Maybe straight lines to see the lanes of the road?
How might we find these features?
How might we find these features?
How might we find these features?

Black: 0
White: 255
How might we find these features?

Black: 0
White: 255
How might we find these features?

Black: 0
White: 255

Look for a Big Change!
How might we find these features?

Convolutions
How might we find these features?

Convolutions

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>255</th>
<th>255</th>
<th>255</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
</tbody>
</table>

Original Image
How might we find these features?

**Convolutions**

**Original Image**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>255</th>
<th>255</th>
<th>255</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
</tbody>
</table>

**Filter**

<table>
<thead>
<tr>
<th>-1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

The filter is applied to the original image to detect edges or other features.
How might we find these features?

Convolutions

Original Image

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>255</th>
<th>255</th>
<th>255</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
</tbody>
</table>

Filter

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

How might we find these features?

Convolutions

Original Image

Filter

Output Feature Map

-1 0 1
-1 0 1
-1 0 1

765
How might we find these features?

**Convolutions**

![Original Image](image)

![Filter](image)

![Output Feature Map](image)
How might we find these features?

**Convolutions**

<table>
<thead>
<tr>
<th>-2</th>
<th>0</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Colab Link
What features are needed for Object Detection?
What features are needed for Object Detection?

The ImageNet Challenge provided 1.2 million labeled examples of 1,000 labeled items and challenged algorithms to learn from the data and then was tested on another 100,000 images.
What features are needed for Object Detection?

Vertical Lines, Horizontal Lines, Changes in Color, Changes in Focus, etc.

Regression, Clustering, etc.
What features are needed for Object Detection?

In 2010 teams had 75-50% error

In 2011 teams had 75-25% error
What features are needed for Object Detection?

In 2012 still no team had less than 25% error barrier except AlexNet at 15%
What features are needed for Object Detection?

Let the computer figure out its own features and how to combine them!
AlexNet

Use convolutions to find features and the summarize them into higher level features

Combine the features to classify the various objects in the dataset
How might we find these features?

Convolutions

Features
How might we find these features?

**Convolutions**

\[
\begin{array}{ccc}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{array}
\]
How might we find these features?

Convolutions
How might we find these features?

**Convolutions**
How might we find these features?

Convolutions
How might we find these features?

**Convolutions**

First Layer Filters
Learned by AlexNet
AlexNet

Use convolutions to find features and the summarize them into higher level features

Combine the features to classify the various objects in the dataset
What features are needed for Object Detection?

https://www.researchgate.net/figure/Historical-top5-error-rate-of-the-annual-winner-of-the-ImageNet-image-classification_fig7_303992986
A word of caution...

Ackerman “Hacking the Brain With Adversarial Images”

There is no model of the world semantically just mathematically
A word of caution...

There is no model of the world semantically just mathematically

Ackerman “Hacking the Brain With Adversarial Images”

Today’s Agenda

- Introduction to Computer Vision
- Hands-on Computer Vision: Thing Translator
- Building an Object Detection Dataset
- Training our Model using Transfer Learning
- Deploying our Model onto our Arduino
- Summary
The **Thing Translator**

https://thing-translator.appspot.com/

Open On Your Phone
The **Thing Translator**

[https://thing-translator.appspot.com/](https://thing-translator.appspot.com/)

Open On Your Phone
Today’s Agenda

- Introduction to Computer Vision
- Hands-on Computer Vision: Thing Translator
- Building an Object Detection Dataset
- Training our Model using Transfer Learning
- Deploying our Model onto our Arduino
- Summary
The TinyML Workflow using Edge Impulse

1. Collect Data
2. Preprocess Data
3. Design a Model
4. Train a Model
5. Evaluate
6. Optimize
7. Convert Model
8. Deploy Model
9. Make Inferences

Dataset -> Impulse -> Test -> Deploy
Create an Edge Impulse Account

1. Create an Edge Impulse account: https://studio.edgeimpulse.com/signup

2. Validate your email by clicking the link in the email sent to your account’s email address
Select project

Select your Edge Impulse project, or create a new one.
Welcome to your new Edge Impulse project!

You're ready to add real intelligence to your edge devices. Let's set up your project. What type of data are you dealing with?

**Accelerometer data**
Analyze movement of your device in real-time to predict machine failure, detect human gestures, or monitor rotating machines.

**Audio**
Listen to what's happening around you to create voice interfaces, listen to keywords, detect audible objects, or to have what's happening around your device.

**Images**
Add sight to your sensors with image classification or object detection - to detect humans and animals, monitor production lines or track objects.

**Something else**
Different sensor? No problem! You can collect and import data from any sensor, from environmental sensors to radars - and deploy your trained model back to virtually any device.
Welcome to your new Edge Impulse project!

Great! What do you want to detect?

Classify a single object (image classification)
Detect one object in an image, for example whether you see a lamp or a plant. Image classification is efficient and can be run on microcontrollers.

Classify multiple objects (object detection)
Detect the location of multiple objects in an image, for example to detect how many apples you see. Object detection is a lot more compute intensive than image classification and currently only works on Linux-based devices like the Raspberry Pi 4 or Jetson Nano.
Welcome to your new Edge Impulse project!

Great! Here’s how you can get started with image classification:

Connect a development board
Get started with real hardware from a wide range of silicon vendors to quickly build a custom image dataset.

- Connect your development board

Import existing data
If you already have images in JPG or PNG file format, you can upload it to Edge Impulse through the web interface or using the Edge Impulse CLI.

- Go to the uploader

Tutorial: adding sight to your sensors
Follow our end-to-end tutorial to collect data, train a model, and deploy it back to your device to analyze images in realtime.

- Read the tutorial

I know what I’m doing, hide this wizard!

Let’s get started!
Edge Impulse Project Dashboard

[Diagram showing a pipeline for data collection, preprocessing, model design, training, evaluation, optimization, conversion, deployment, and making inferences.]
Activity: Create an Object Classification Dataset

Collect ~30 samples each of the following classes of data:

- Target Object #1
- Target Object #2
- (Optional) Target Object #3
Creating your first impulse (0% complete)

Acquire data
Every Machine Learning project starts with data. You can capture data from sensors or import data you already collected.

Design an impulse
Teach the model to interpret previously unseen data, based on historical data. Use this to categorize new data, or to find anomalies in sensor readings.
You can collect data from development boards, from your own devices, or by uploading an existing dataset.

**Connect a fully supported development board**
Get started with real hardware from a wide range of silicon vendors - fully supported by Edge Impulse.

**Use your mobile phone**
Use your mobile phone to capture movement, audio or images, and even run your trained model locally. No app required.

**Use your computer**
Capture audio or images from your webcam or microphone, or from an external audio device.

**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.

**Upload data**
Already have data? You can upload your existing datasets directly in WAV, JPG, PNG, CBOR, CSV or JSON format.
EDGE IMPULSE

Dashboard

Data acquisition

Project info  Keys  Export

Brian_plancher

This is your Edge Impulse project. From here you can:

About this project

Creating your first impulse (0% complete)

Acquire data
Every Machine Learning project starts with a dataset. This can be either a development board or your phone.

Let's collect some data

Design an impulse
Teach the model to interpret previous data. Use this to categorize new data readings.

Acquire data

Design an impulse
Did you know? You can capture data from any device or development board, or upload your existing datasets - Show options

Upload existing data
...dgeimpulse.com wants to connect to a serial port

Nano 33 BLE (ttyACM0) - Paired

- ttyS0
- ttyS1
- ttyS10
- ttyS11
- ttyS12
- ttyS13
- ttyS14

[Connect]
You may need to re-flash the EI Firmware!

1. Double tap RESET to enter bootloader mode


3. Run the flash script for your operating system (flash_windows.bat, flash_mac.command or flash_linux.sh).

4. Wait until flashing is complete, and press the RESET button once to launch the new firmware.
Did you know? You can capture data from any device or development board, or upload your existing datasets - Show options

Record new data

Device
6F:3E:4B:F3:11:23

Label
truck

Sensor
Camera (160x120)

Start sampling
<table>
<thead>
<tr>
<th>String</th>
<th>Type</th>
<th>Date</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>truck.30rop8b</td>
<td>truck</td>
<td>Today, 16:05:45</td>
<td>-</td>
</tr>
<tr>
<td>truck.30ropdr8</td>
<td>truck</td>
<td>Today, 16:05:45</td>
<td>-</td>
</tr>
<tr>
<td>truck.30rop4ea</td>
<td>truck</td>
<td>Today, 16:05:45</td>
<td>-</td>
</tr>
<tr>
<td>truck.30roohr0</td>
<td>truck</td>
<td>Today, 16:05:45</td>
<td>-</td>
</tr>
<tr>
<td>truck.30rp422j</td>
<td>truck</td>
<td>Today, 16:05:32</td>
<td>-</td>
</tr>
<tr>
<td>truck.30rp3gr4</td>
<td>truck</td>
<td>Today, 16:05:14</td>
<td>-</td>
</tr>
<tr>
<td>truck.30rp349b</td>
<td>truck</td>
<td>Today, 16:05:02</td>
<td>-</td>
</tr>
<tr>
<td>SAMPLE NAME</td>
<td>LABEL</td>
<td>ADDED</td>
<td>LENGTH</td>
</tr>
<tr>
<td>----------------</td>
<td>-------</td>
<td>---------------------</td>
<td>--------</td>
</tr>
<tr>
<td>truck.30sf60s</td>
<td>truck</td>
<td>Yesterday, 22:42:38</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sf2va</td>
<td>truck</td>
<td>Yesterday, 22:42:35</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfqmn</td>
<td>truck</td>
<td>Yesterday, 22:42:32</td>
<td></td>
</tr>
<tr>
<td>truck.30sfq45</td>
<td>truck</td>
<td>Yesterday, 22:42:27</td>
<td></td>
</tr>
<tr>
<td>truck.30sfqkg</td>
<td>truck</td>
<td>Yesterday, 22:42:21</td>
<td></td>
</tr>
<tr>
<td>truck.30sfq538</td>
<td>truck</td>
<td>Yesterday, 22:42:05</td>
<td></td>
</tr>
<tr>
<td>truck.30sfq0k</td>
<td>truck</td>
<td>Yesterday, 22:42:00</td>
<td></td>
</tr>
</tbody>
</table>

RAW DATA

truck.30sf2va

Options:
- Rename
- Edit label
- Move to test set
- Disable
- Download
- Delete
Activity: Create an Object Classification Dataset

Collect ~30 samples each of the following classes of data:

- Target Object #1
- Target Object #2
- (Optional) Target Object #3

Download the firmware:
bit.ly/El-Nano33-Firmware

flash_windows.bat
flash_mac.command
flash_linux.sh
### DATA COLLECTED

- **60 items**

### TRAIN / TEST SPLIT

- **100% / 0%**

#### Collected data

<table>
<thead>
<tr>
<th>SAMPLE NAME</th>
<th>LABEL</th>
<th>ADDED</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>truck.30sfr605</td>
<td>truck</td>
<td>Today, 22:42:38</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfr2va</td>
<td>truck</td>
<td>Today, 22:42:35</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfqmn</td>
<td>truck</td>
<td>Today, 22:42:32</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfqr45</td>
<td>truck</td>
<td>Today, 22:42:27</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfq45g</td>
<td>truck</td>
<td>Today, 22:42:21</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfq538</td>
<td>truck</td>
<td>Today, 22:42:05</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfq0f4</td>
<td>truck</td>
<td>Today, 22:42:00</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfpi8a</td>
<td>truck</td>
<td>Today, 22:41:45</td>
<td>-</td>
</tr>
</tbody>
</table>
DATA ACQUISITION (TEST IMAGE 2)

Training data  |  Test data  |  Export data

Did you know? You can capture data from any device or development board, or upload your existing datasets - Show options

Record new data

Device

Label

Sensor

Camera feed

Start sampling

Collected data

<table>
<thead>
<tr>
<th>SAMPLE NAME</th>
<th>LABEL</th>
<th>ADDED</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>truck.30sf605</td>
<td>truck</td>
<td>Today, 22:42:38</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sf2va</td>
<td>truck</td>
<td>Today, 22:42:35</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfqmn</td>
<td>truck</td>
<td>Today, 22:42:32</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfq45</td>
<td>truck</td>
<td>Today, 22:42:27</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfqsg</td>
<td>truck</td>
<td>Today, 22:42:21</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfq38</td>
<td>truck</td>
<td>Today, 22:42:05</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfq0f</td>
<td>truck</td>
<td>Today, 22:42:00</td>
<td>-</td>
</tr>
<tr>
<td>truck.30sfpi8a</td>
<td>truck</td>
<td>Today, 22:41:45</td>
<td>-</td>
</tr>
</tbody>
</table>
Danger zone

- Perform train / test split
- Launch getting started wizard
- Transfer ownership
- Delete this project
- Delete all data in this project
Danger zone

Perform train / test split

Launch getting started

Transfer ownership

Delete this project

Delete all data in this project

Perform train / test split

Are you sure you want to rebalance your dataset? This splits all your data automatically between the training and testing set, and resets the categories for all data. This is irrevocable!

Cancel  Yes, perform the train / test split
Confirm

Enter "perform split" to continue

perform split

Cancel  Perform train / test split
**DATA ACQUISITION - TESTING**

- **Training data**
- **Test data**
- **Export data**

**DATA COLLECTED**
- 12 items

**TRAIN / TEST SPLIT**
- 80% / 20%

---

**Collected data**

<table>
<thead>
<tr>
<th>SAMPLE NAME</th>
<th>LABEL</th>
<th>ADDED</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>car.30sfn dar</td>
<td>car</td>
<td>Yesterday, 22:40:35</td>
<td></td>
</tr>
<tr>
<td>car.30sfmdvi</td>
<td>car</td>
<td>Yesterday, 22:40:03</td>
<td></td>
</tr>
<tr>
<td>car.jpg.30pr4p5.ingestion-7...</td>
<td>car</td>
<td>Yesterday, 22:37:46</td>
<td></td>
</tr>
<tr>
<td>truck.jpg.30rv8kr.ingestion-7...</td>
<td>truck</td>
<td>Yesterday, 22:37:45</td>
<td></td>
</tr>
<tr>
<td>truck.jpg.30rv9gs9.ingestion-7...</td>
<td>truck</td>
<td>Yesterday, 22:37:44</td>
<td></td>
</tr>
<tr>
<td>car.jpg.30prnc2.ingestion-7...</td>
<td>car</td>
<td>Yesterday, 22:37:44</td>
<td></td>
</tr>
<tr>
<td>car.jpg.30pradun.ingestion-7...</td>
<td>car</td>
<td>Yesterday, 22:37:44</td>
<td></td>
</tr>
<tr>
<td>truck.jpg.30rv9q9f.ingestion-7...</td>
<td>truck</td>
<td>Yesterday, 22:37:44</td>
<td></td>
</tr>
</tbody>
</table>
Today’s Agenda

- Introduction to Computer Vision
- Hands-on Computer Vision: Thing Translator
- Building an Object Detection Dataset
- Training our Model using Transfer Learning
- Deploying our Model onto our Arduino
- Summary
Edge Impulse Project Dashboard
Edge Impulse Project Dashboard

- Collect Data
- Preprocess Data
- Design a Model
- Train a Model
- Evaluate
- Optimize
- Convert Model
- Deploy Model
- Make Inferences

- Dataset
- Impulse
- Test
- Deploy

- Dashboard
- Devices
- Data acquisition
- Impulse design
- Create impulse
- EON Tuner
- Retrain model
- Live classification
- Model testing
- Versioning
- Deployment
Transfer Learning: Saving time and computational resources
Transfer Learning: Saving time and computational resources

Learns \textit{general features} irrespective of task
Transfer Learning: Saving time and computational resources

Learns **general features** irrespective of task
Transfer Learning: Saving time and computational resources

Task-specific features
Transfer Learning: Saving time and computational resources

Learns \textit{general features} irrespective of task

\textbf{Reuse} (freeze general feature extraction)
Transfer Learning: Saving time and computational resources

Input A

W_{A1} W_{A2} W_{A3} W_{A4}

W_{A5} W_{A6} W_{A7}

Labels A

Task-specific features

Train only last few layers
So what model should we transfer from?
Model Evolution
Model Evolution

![Diagram showing the evolution of different models with top-1 accuracy and operations as axes. The MobileNet v1 model is highlighted.]
### MobileNet v1

<table>
<thead>
<tr>
<th>Model</th>
<th>Size</th>
<th>Top-1 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNet v1</td>
<td>16 MB</td>
<td>0.713</td>
</tr>
</tbody>
</table>

Fine for mobile phones with GB of RAM, but 64X microcontroller RAM

Our board [Course 3 Kit] only has 256KB of RAM (memory)
Further **Optimizations**

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Image Size</th>
<th>MACs (millions)</th>
<th>Params (millions)</th>
<th>Top-1 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>224</td>
<td>569</td>
<td>4.24</td>
<td>70.7</td>
</tr>
<tr>
<td>1</td>
<td>128</td>
<td>186</td>
<td>4.14</td>
<td>64.1</td>
</tr>
<tr>
<td>0.75</td>
<td>224</td>
<td>317</td>
<td>2.59</td>
<td>68.4</td>
</tr>
<tr>
<td>0.75</td>
<td>128</td>
<td>104</td>
<td>2.59</td>
<td>61.8</td>
</tr>
<tr>
<td>0.5</td>
<td>224</td>
<td>150</td>
<td>1.34</td>
<td>64.0</td>
</tr>
<tr>
<td>0.5</td>
<td>128</td>
<td>49</td>
<td>1.34</td>
<td>56.2</td>
</tr>
<tr>
<td>0.25</td>
<td>224</td>
<td>41</td>
<td>0.47</td>
<td>50.6</td>
</tr>
<tr>
<td>0.25</td>
<td>128</td>
<td>14</td>
<td>0.47</td>
<td>41.2</td>
</tr>
</tbody>
</table>
Further **Optimizations**

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Image Size</th>
<th>MACs (millions)</th>
<th>Params (millions)</th>
<th>Top-1 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>224</td>
<td>569</td>
<td>4.24</td>
<td>70.7</td>
</tr>
<tr>
<td>1</td>
<td>128</td>
<td>186</td>
<td>4.14</td>
<td>64.1</td>
</tr>
<tr>
<td>0.75</td>
<td>224</td>
<td>317</td>
<td>2.59</td>
<td>68.4</td>
</tr>
<tr>
<td>0.75</td>
<td>128</td>
<td>104</td>
<td>2.59</td>
<td>61.8</td>
</tr>
<tr>
<td>0.5</td>
<td>224</td>
<td>150</td>
<td>1.34</td>
<td>64.0</td>
</tr>
<tr>
<td>0.5</td>
<td>128</td>
<td>49</td>
<td>1.34</td>
<td>56.2</td>
</tr>
<tr>
<td>0.25</td>
<td>224</td>
<td>41</td>
<td>0.47</td>
<td>50.6</td>
</tr>
<tr>
<td>0.25</td>
<td>128</td>
<td>14</td>
<td>0.47</td>
<td>41.2</td>
</tr>
</tbody>
</table>
## Further Optimizations

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Image Size</th>
<th>MACs (millions)</th>
<th>Params (millions)</th>
<th>Top-1 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>224</td>
<td>569</td>
<td>4.24</td>
<td>70.7</td>
</tr>
<tr>
<td>1</td>
<td>128</td>
<td>186</td>
<td>4.14</td>
<td>64.1</td>
</tr>
<tr>
<td>0.75</td>
<td>224</td>
<td>317</td>
<td>2.59</td>
<td>68.4</td>
</tr>
<tr>
<td>0.75</td>
<td>128</td>
<td>104</td>
<td>2.59</td>
<td>61.8</td>
</tr>
<tr>
<td>0.5</td>
<td>224</td>
<td>150</td>
<td>1.34</td>
<td>64.0</td>
</tr>
<tr>
<td>0.5</td>
<td>128</td>
<td>49</td>
<td>1.34</td>
<td>56.2</td>
</tr>
<tr>
<td>0.25</td>
<td>224</td>
<td>41</td>
<td>0.47</td>
<td>50.6</td>
</tr>
<tr>
<td>0.25</td>
<td>128</td>
<td>14</td>
<td>0.47</td>
<td>41.2</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>Image Size</td>
<td>MACs (millions)</td>
<td>Params (millions)</td>
<td>Top-1 Accuracy</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
<td>-----------------</td>
<td>-------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>1</td>
<td>224</td>
<td>569</td>
<td>4.24</td>
<td>70.7</td>
</tr>
<tr>
<td>1</td>
<td>128</td>
<td>186</td>
<td>4.14</td>
<td>64.1</td>
</tr>
<tr>
<td>0.75</td>
<td>224</td>
<td>317</td>
<td>2.59</td>
<td>68.4</td>
</tr>
<tr>
<td>0.75</td>
<td>128</td>
<td>104</td>
<td>2.59</td>
<td>61.8</td>
</tr>
<tr>
<td>0.5</td>
<td>224</td>
<td>150</td>
<td>1.34</td>
<td>64.0</td>
</tr>
<tr>
<td>0.5</td>
<td>128</td>
<td>49</td>
<td>1.34</td>
<td>56.2</td>
</tr>
<tr>
<td>0.25</td>
<td>224</td>
<td>41</td>
<td>0.47</td>
<td>50.6</td>
</tr>
<tr>
<td>0.25</td>
<td>128</td>
<td>14</td>
<td>0.47</td>
<td>41.2</td>
</tr>
</tbody>
</table>
## Further Optimizations

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>Image Size</th>
<th>MACs (millions)</th>
<th>Params (millions)</th>
<th>Top-1 Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>224</td>
<td>569</td>
<td>4.24</td>
<td>70.7</td>
</tr>
<tr>
<td>1</td>
<td>128</td>
<td>186</td>
<td>4.14</td>
<td>64.1</td>
</tr>
<tr>
<td>0.75</td>
<td>224</td>
<td>317</td>
<td>2.59</td>
<td>68.4</td>
</tr>
<tr>
<td>0.75</td>
<td>128</td>
<td>104</td>
<td>2.59</td>
<td>61.8</td>
</tr>
<tr>
<td>0.5</td>
<td>224</td>
<td>150</td>
<td>1.34</td>
<td>64.0</td>
</tr>
<tr>
<td>0.5</td>
<td>128</td>
<td>49</td>
<td>1.34</td>
<td>56.2</td>
</tr>
<tr>
<td>0.25</td>
<td>224</td>
<td>41</td>
<td>0.47</td>
<td>50.6</td>
</tr>
<tr>
<td>0.25</td>
<td>128</td>
<td>14</td>
<td>0.47</td>
<td>41.2</td>
</tr>
</tbody>
</table>

We will need to both reduce alpha and the image size!
An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.
MobileNet is trained on square images!
An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.
We are just going to use the suggested standard processing block and not do anything sophisticated.
An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.
An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.
An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.
Successfully stored impulse. Configure the signal processing and learning blocks in the navigation bar.

For optimal accuracy with transfer learning blocks, use a 96x96 or 160x160 image size.
# Training set

<table>
<thead>
<tr>
<th>Data in training set</th>
<th>48 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td>2 (Car, Truck)</td>
</tr>
</tbody>
</table>

## Feature explorer

No features generated yet.
Neural Network settings

Training settings

Number of training cycles: 20
Learning rate: 0.0005
Validation set size: 20

Auto-balance dataset: 
Data augmentation: 

Neural network architecture

Input layer (27,648 features)

MobileNetV2 (52,496 features) final layer: 16 neurons, 0.1 dropout

Choose a different model

Output layer (2 classes)
Choose a different model

Did you know? You can customize your model using Keras through the Expert view (click on to switch).

**Layer Type**

**MobileNetV1 96x96 0.25**
A pre-trained multi-layer convolutional network designed to efficiently classify images. Uses around 105.9K RAM and 301.6K ROM with default settings and optimizations.

**MobileNetV1 96x96 0.2**
Uses around 83.1K RAM and 218.3K ROM with default settings and optimizations. Works best with 96x96 input size. Supports both RGB and grayscale.

**MobileNetV1 96x96 0.1**
Uses around 53.2K RAM and 101K ROM with default settings and optimizations. Works best with 96x96 input size. Supports both RGB and grayscale.

**MobileNetV2 96x96 0.35**
Uses around 296.8K RAM and 575.2K ROM with default settings and optimizations. Works best with 96x96 input size. Supports both RGB and grayscale.

**MobileNetV2 96x96 0.1**
Uses around 270.2K RAM and 212.3K ROM with default settings and optimizations. Works best with 96x96 input size. Supports both RGB and grayscale.

**MobileNetV2 96x96 0.05**
Uses around 265.3K RAM and 162.4K ROM with default settings and optimizations. Works best with 96x96 input size. Supports both RGB and grayscale.

**MobileNetV2 160x160 1.0**
Uses around 1.3M RAM and 2.6M ROM with default settings and optimizations. Works best with 160x160 input size. Supports RGB only.

**MobileNetV2 160x160 0.75**
Limited Memory
## Neural Network settings

### Training settings

- **Number of training cycles**: 20
- **Learning rate**: 0.0005
- **Validation set size**: 20%
- **Auto-balance dataset**: 
- **Data augmentation**: 

### Neural network architecture

- **Input layer**: 22,648 features

- **MobileNetV1 96x96 0.1 (no final dense layer, 0.1 dropout)**

- **Output layer (2 classes)**

[Start training button]
<table>
<thead>
<tr>
<th>Epoch</th>
<th>8s</th>
<th>loss: 0.1044</th>
<th>accuracy: 0.9500</th>
<th>val_loss: 0.2934</th>
<th>val_accuracy: 0.9231</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8s</td>
<td>loss: 0.0256</td>
<td>accuracy: 1.0000</td>
<td>val_loss: 0.3830</td>
<td>val_accuracy: 0.8846</td>
</tr>
<tr>
<td></td>
<td>8s</td>
<td>loss: 0.0523</td>
<td>accuracy: 0.9800</td>
<td>val_loss: 0.4366</td>
<td>val_accuracy: 0.8462</td>
</tr>
<tr>
<td></td>
<td>8s</td>
<td>loss: 0.0451</td>
<td>accuracy: 0.9800</td>
<td>val_loss: 0.4265</td>
<td>val_accuracy: 0.8846</td>
</tr>
<tr>
<td></td>
<td>8s</td>
<td>loss: 0.0514</td>
<td>accuracy: 0.9900</td>
<td>val_loss: 0.3926</td>
<td>val_accuracy: 0.8846</td>
</tr>
<tr>
<td></td>
<td>8s</td>
<td>loss: 0.0348</td>
<td>accuracy: 0.9900</td>
<td>val_loss: 0.3571</td>
<td>val_accuracy: 0.9231</td>
</tr>
</tbody>
</table>

Finished training.
Final Test Accuracy

Model

Last training performance (validation set)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>70.0%</td>
</tr>
<tr>
<td>Loss</td>
<td>0.33</td>
</tr>
</tbody>
</table>

Confusion matrix (validation set)

<table>
<thead>
<tr>
<th></th>
<th>Car</th>
<th>Truck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car</td>
<td>0%</td>
<td>60%</td>
</tr>
<tr>
<td>Truck</td>
<td>20%</td>
<td>0%</td>
</tr>
<tr>
<td>F1 Score</td>
<td>0.67</td>
<td>0.23</td>
</tr>
</tbody>
</table>

Feature explorer (full training set)

On-device performance

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference Time</td>
<td>58 ms</td>
</tr>
<tr>
<td>Peak RAM Usage</td>
<td>66.1K</td>
</tr>
<tr>
<td>Flash Usage</td>
<td>108.1K</td>
</tr>
</tbody>
</table>
Final Test Accuracy

Accuracy Breakdown
## Confusion Matrix

<table>
<thead>
<tr>
<th></th>
<th>Actually Object 1</th>
<th>Actually Object 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predicted Object 1</td>
<td># of Correct Object 1</td>
<td># of Error</td>
</tr>
<tr>
<td>Predicted Object 2</td>
<td># of Error</td>
<td># of Correct Object 2</td>
</tr>
</tbody>
</table>
Final Test Accuracy

Accuracy Breakdown
Feature explorer (full training set)

- car - correct
- truck - correct
- car - incorrect
- truck - incorrect

car.30sfl3kp
Label: car
Predicted: truck
View sample
View features
Final Test Accuracy

Accuracy Breakdown

Memory and Time
Edge Impulse Project Dashboard

- Collect Data
  - Preprocess Data
  - Design a Model
  - Train a Model
  - Evaluate Optimize
  - Convert Model
  - Deploy Model
  - Make Inferences

- Dataset
- Impulse
- Test
- Deploy

- Dashboard
- Devices
- Data acquisition
- Impulse design
- Create impulse
- MFCC
- NN Classifier
- EON Tuner
- Retrain model
- Live classification
- Model testing
- Versioning
- Deployment
Today’s Agenda

- Introduction to Computer Vision
- Hands-on Computer Vision: Thing Translator
- Building an Object Detection Dataset
- Training our Model using Transfer Learning
- **Deploying our Model onto our Arduino**
- Summary
Edge Impulse Project Dashboard

Edge Impulse Project Dashboard

- Collect Data
- Preprocess Data
- Design a Model
- Train a Model
- Evaluate
- Optimize
- Convert Model
- Deploy Model
- Make Inferences

Dataset → Impulse → Test → Deploy

- Dashboard
- Devices
- Data acquisition
- Impulse design
- Create impulse
- MFCC
- NN Classifier
- EON Tuner
- Retrain model
- Live classification
- Model testing
- Versioning
- Deployment
Deploy your impulse

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. Read more.

Create library

Turn your impulse into optimized source code for Arduino.

Build firmware

Get a ready-to-go binary for your development board that includes your impulse.
Select optimizations (optional)

Model optimizations can increase on-device performance but may reduce accuracy. Click below to analyze optimizations and see the recommended choices for your target. Or, just click Build to use the currently selected options.

Enable EON™ Compiler
Same accuracy, up to 50% less memory. Open source.

Available optimizations for Transfer learning

<table>
<thead>
<tr>
<th></th>
<th>RAM USAGE</th>
<th>LATENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantized (int8)</td>
<td>66.1K</td>
<td>58 ms</td>
</tr>
<tr>
<td></td>
<td>108.1K</td>
<td></td>
</tr>
<tr>
<td>Unoptimized (float32)</td>
<td>155.6K</td>
<td>43 ms</td>
</tr>
<tr>
<td></td>
<td>193.8K</td>
<td></td>
</tr>
</tbody>
</table>

Estimate for Arduino Portenta H7 (Cortex-M7 480MHz)

[Build button]
Reduces the precision of numbers used in a model which results in:
- smaller model size
- faster computation
Reducing the Precision

**float32**
- max: $3.40282 \times 10^{38}$
- min: $1.17549 \times 10^{-38}$

**int8**
- max: 127
- min: -128

4 bytes per model parameter

1 byte per model parameter
## Tradeoff

<table>
<thead>
<tr>
<th>Model</th>
<th>Floating-point Baseline</th>
<th>After Quantization</th>
<th>Accuracy Drop</th>
</tr>
</thead>
<tbody>
<tr>
<td>MobileNet v1 1.0 224</td>
<td>71.03%</td>
<td>69.57%</td>
<td>▼1.46%</td>
</tr>
<tr>
<td>MobileNet v2 1.0 224</td>
<td>70.77%</td>
<td>70.20%</td>
<td>▼0.57%</td>
</tr>
<tr>
<td>Resnet v1 50</td>
<td>76.30%</td>
<td>75.95%</td>
<td>▼0.35%</td>
</tr>
</tbody>
</table>
Select optimizations *(optional)*

Model optimizations can increase on-device performance but may reduce accuracy. Click below to analyze optimizations and see the recommended choices for your target. Or, just click Build to use the currently selected options.

**Enable EON™ Compiler**
Same accuracy, up to 50% less memory. Open source.

**Available optimizations for Transfer learning**

<table>
<thead>
<tr>
<th>Quantized (int8)</th>
<th>RAM USAGE</th>
<th>LATENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Currently selected</strong></td>
<td>66.1K</td>
<td>58 ms</td>
</tr>
<tr>
<td><strong>Unoptimized (float32)</strong></td>
<td>155.6K</td>
<td>43 ms</td>
</tr>
</tbody>
</table>

**Unoptimized (float32)**

<table>
<thead>
<tr>
<th>RAM USAGE</th>
<th>LATENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>193.8K</td>
<td>-</td>
</tr>
</tbody>
</table>

Estimate for Arduino Portenta H7 (Cortex M7 480MHz)

[Build]
Edge Impulse Project Dashboard

- Collect Data
- Preprocess Data
- Design a Model
- Train a Model
- Evaluate
- Optimize
- Convert Model
- Deploy Model
- Make Inferences

Dataset → Impulse → Test → Deploy

- Dashboard
- Devices
- Data acquisition
- Impulse design
- Create impulse
- MFCC
- NN Classifier
- EON Tuner
- Retrain model
- Live classification
- Model testing
- Versioning
- Deployment
Built Arduino library

Add this library through the Arduino IDE via:

Sketch > Include Library > Add .ZIP Library...

Examples can then be found under:

File > Examples > test_image_2_inferencing
Built Arduino library

Add this library through the Arduino IDE via:

Sketch > Include Library > Add .ZIP Library...

Examples can then be found under:

File > Examples > test_image_2_inferencing
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*/

/* Includes --------------------------------------------- */
#include <test_image_inferencing.h>
#include <Arduino_OV767X.h>

#include <stdio.h>
#include <stdlib.h>

Arduino_OV767X.h: No such file or directory
Arduino_OV767X
by Arduino  Version 0.0.2 INSTALLED
Capture images from your OmniVision OV7670 camera in your Arduino sketches.
More info
Select version  Install

Harvard_TinyMLx
by Brian Plancher  Version 1.1.0-Alpha INSTALLED
Supports the TinyML edX Course and TinyML Shield. This library supports the TinyML Shield and provides examples that support the TinyML edX course. The examples work best with the Arduino Nano 33 BLE Sense board and the Tiny Machine Learning Kit from Arduino. It also includes a modified version of the Arduino_OV767X library version 0.0.2 and a fork of the TensorFlow_Lite version 2.4.0-Alpha Arduino examples.
More info
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
* SOFTWARE.
*/

#include <test_image_inferencing.h>
#include <Arduino_OV767X.h>

#include <stdio.h>
#include <stdlib.h>

Arduino_OV767X.h: No such file or directory

nano_ble33_sense_camera:25:10: fatal error: Arduino_OV767X.h: No such file or directory
  #include <Arduino_OV767X.h>
  ~~~~~~~~~~~
compilation terminated.
exit status 1
Arduino_OV767X.h: No such file or directory
Double Tap RESET for Bootloader Mode!
Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 9 ms., Classification: 322 ms., Anomaly: 0 ms.):
car: 0.07812
truck: 0.92188

Confidence that the picture is the given class (0-1 scale)
Today’s Agenda

- Introduction to Computer Vision
- Hands-on Computer Vision: Thing Translator
- Building an Object Detection Dataset
- Training our Model using Transfer Learning
- Deploying our Model onto our Arduino

Summary
Machine Learning

We provide answers aka the labels of the data. The computer learns.

Inputs -> Open box

Answers

Rules
Deep Learning with Neural Networks
Features can be found with **Convolutions**
Convolutional Neural Networks
The TinyML Workflow

- Collect Data
- Preprocess Data
- Design a Model
- Train a Model
- Evaluate
- Optimize
- Convert Model
- Deploy Model
- Make Inferences

Dataset → Impulse → Test → Deploy

Camera feed

Starting inferencing in 2 seconds...
Taking photo...
Predictions (DSP: 9 ms., Classification:
car: 0.07812
truck: 0.92188
Convolutions for Hands-on Computer Vision

Brian Plancher
Harvard John A. Paulson School of Engineering and Applied Sciences
Barnard College, Columbia University
brianplancher.com
Edge Impulse CLI Notes:

1. Install the Arduino CLI
   a. On linux:
      
   b. On mac:
      
      brew update
      brew install arduino-cli
   c. Or view the link for binaries
2. Add to your .bashrc:

   # Arduino (CLI)
   export PATH="ARDUINO_INSTALL_LOCATION/bin:$PATH"

   Where ARDUINO_INSTALL_LOCATION is e.g.,: $HOME/Documents/arduino-1.8.19
1. **Install the Edge Impulse CLI**
   a. **Install Node.js** by following the link or on Linux:
      
      ```
curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
sudo apt-get install -y nodejs
      ```
   b. **Run**: `npm install -g edge-impulse-cli --force`
   c. **Add to your .bashrc**:
      
      ```
      # EI (CLI)
      export PATH="$HOME/.npm-global/bin:$PATH"
      ```

2. **Run** `edge-impulse-daemon --clean` to start the daemon and then follow the instructions in the terminal to add it to your current project using your Edge Impulse account!
It should then appear on your “Devices” tab in your project!

And then if you go to “Data Acquisition” you should be able to proceed as you would with the standard instructions!

https://docs.edgeimpulse.com/docs/edge-impulse-cli/cli-installation