Yá’át’éeéh👋

EASI-22

Edge AI Summer Institute 2022

with Navajo Tech
Hi! I’m Brian!

I’m an Assistant Professor of Computer Science at Barnard College, Columbia University
Our team!

with help from many more
Our website!
tinyMLedu.org/EASI-22

home base for all information!
Make Sure to Pick Up an Arduino Kit!

Question? Contact: Monsuru Ramoni
mramoni@navajotech.edu
Teachers Sign up for Buy2Pay

Question? Contact: Molly Marshall
mmmarshall@seas.harvard.edu
Workshop Agenda

Day 1
Introduction to AI and (Tiny)ML

Day 2
Keyword Spotting for the Navajo Language

Day 3
Bringing AI/ML from the Cloud to the Edge

Cloud ML
Mobile ML
Embedded ML
Keyword Spotting in One Slide

If we pick a simple task to only identifying a few key words we can then use a small model and train it with little data and fit it onto an embedded device.
We will explore the science behind KWS and collect data and train our own custom model to recognize “yes” vs. “no” using Edge Impulse.
Today’s Agenda

- Preprocessing for Keyword Spotting
- Convolutional Neural Networks for Image Classification
- Hands-on: KWS Data Collection with Edge Impulse
- Hands-on: Training our Model with Edge Impulse
- Hands-on: Testing our Model in the Real World
- Summary
But before we dive into all of that – a little quick review!
Machine Learning

We provide answers, aka the labels of the data.

The computer learns input rules.
Machine Learning with neural networks?
Training the machine

For a set of Input Data

Guess the Answer and count mistakes

Improve the model to be more correct
After it’s learned use it for inference:

NEW UNLABELED

INPUTS

Model
RULES

NEW

ANSWERS (LABELS)

DOG

CAT
If ML is going to be everywhere we need to consider how to best collect GOOD data RESPONSIBLY.
Today’s Agenda

- Preprocessing for Keyword Spotting
  - Convolutional Neural Networks for Image Classification
  - Hands-on: KWS Data Collection with Edge Impulse
  - Hands-on: Training our Model with Edge Impulse
  - Hands-on: Testing our Model in the Real World
- Summary
Machine Learning Workflow

1. Collect & Transform Data
2. Design & Train a Model
3. Deploy Model
4. Make Inferences
This is an audio signal

“Yes” (spoken loudly)
Can you tell these two signals apart?

“**Yes**” *(spoken loudly)*

“**No**” *(spoken loudly)*
Signal **Components**?
Signal **Components**?

? + ? = “No” (spoken loudly)
Fast Fourier Transform:
extract the frequencies from a signal
Fast Fourier Transform

FFT

No Loud

Frequency

Time
Building a Spectrogram using FFTs
Building a **Spectrogram** using FFTs
Building a **Spectrogram** using FFTs
Building a **Spectrogram** using FFTs

Essentially if you *stack up all the FFTs in a row* then you get the **Spectrogram** (time vs. frequency with color indicating intensity)
Spectrograms help differentiate the data
Spectrograms help differentiate the data
Spectrograms help differentiate the data
Data Preprocessing: Spectrograms

A spectrogram is also effectively an **image** that we can use as an input to a Neural Network!
Can we do better than a spectrogram?

Can we take domain knowledge into account?
Mel Filterbanks
Spectrograms v. MFCCs

Yes Loud

No Loud

Yes Loud

No Loud
Spectrograms v. MFCCs
Today’s Agenda

- Preprocessing for Keyword Spotting
  - Convolutional Neural Networks for Image Classification
  - Hands-on: KWS Data Collection with Edge Impulse
  - Hands-on: Training our Model with Edge Impulse
  - Hands-on: Testing our Model in the Real World
- Summary
Today’s Agenda

- Preprocessing for Keyword Spotting
- Convolutional Neural Networks for Image Classification
  - Hands-on: KWS Data Collection with Edge Impulse
  - Hands-on: Training our Model with Edge Impulse
  - Hands-on: Testing our Model in the Real World
- Summary
Machine Learning Workflow

- Collect & Transform Data
- Design & Train a Model
- Deploy Model
- Make Inferences
Computer Vision is Hard
Computer Vision is Hard

What color are the pants and the shirt?

Slide Credit: Hamilton Chong
Computer Vision is Hard

Slide Credit: Hamilton Chong
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></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>
<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

| 0 | 0 | 0 | 255 | 255 | 255 |
| 0 | 0 | 0 | 255 | 255 | 255 |
| 0 | 0 | 0 | 255 | 255 | 255 |
| 0 | 0 | 0 | 255 | 255 | 255 |
| 0 | 0 | 0 | 255 | 255 | 255 |
| 0 | 0 | 0 | 255 | 255 | 255 |

Filter

| -1 | 0 | 1 |
| -1 | 0 | 1 |
| -1 | 0 | 1 |
How might we find these features?

Convolutions

<table>
<thead>
<tr>
<th>Original Image</th>
<th>Filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
</tr>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
</tr>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
</tr>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
</tr>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
</tr>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
</tr>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
</tr>
</tbody>
</table>
How might we find these features?

Convolutions

Original Image

<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>
</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>

Output Feature Map

765
How might we find these features?

Convolutions

<table>
<thead>
<tr>
<th>Original Image</th>
<th>Filter</th>
<th>Output Feature Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
<td>0 765 765 0</td>
</tr>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
<td>0 765 765 0</td>
</tr>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
<td>0 765 765 0</td>
</tr>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
<td>0 765 765 0</td>
</tr>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
<td>0 765 765 0</td>
</tr>
<tr>
<td>0 0 0 255 255 255</td>
<td>-1 0 1</td>
<td>0 765 765 0</td>
</tr>
</tbody>
</table>
How might we find these features?

**Convolutions**
How might we combine these features to classify an object?
The ImageNet Challenge and the birth of CNNs

The ImageNet Challenge provided 1.2 million examples of 1,000 labeled items and challenged algorithms to learn from the data and then was tested on another 100,000 images.
The ImageNet Challenge and the birth of CNNs

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

Regression, Clustering, etc.

Traditional Machine Learning Flow
The ImageNet Challenge and the birth of CNNs

In 2010 teams had 75-50% error

In 2011 teams had 75-25% error
The ImageNet Challenge and the birth of CNNs

In 2012 still no team had less than 25% error barrier except AlexNet at 15%
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
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
How might we find these features?

Convolutions
AlexNet Paper

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
The ImageNet Challenge and the birth of CNNs
A word of caution...

Ackerman “Hacking the Brain With Adversarial Images”

There is no model of the world semantically just mathematically

“panda” 57.7% confidence

+ \epsilon

= "gibbon" 99.3% confidence
A word of caution...

Ackerman “Hacking the Brain With Adversarial Images”

There is no model of the world semantically just mathematically
Today’s Agenda

- Preprocessing for Keyword Spotting
- Convolutional Neural Networks for Image Classification
  - Hands-on: KWS Data Collection with Edge Impulse
  - Hands-on: Training our Model with Edge Impulse
  - Hands-on: Testing our Model in the Real World
- Summary
Today’s Agenda

- Preprocessing for Keyword Spotting
- Convolutional Neural Networks for Image Classification
- Hands-on: KWS Data Collection with Edge Impulse
  - Hands-on: Training our Model with Edge Impulse
  - Hands-on: Testing our Model in the Real World
- Summary
Machine Learning Workflow

- Collect & Transform Data
- Design & Train a Model
- Deploy Model
- Make Inferences
Machine Learning Workflow

1. Collect & Transform Data
2. Design & Train a Model
3. Deploy Model
4. Make Inferences
Edge Impulse Project Dashboard

1. Collect Data
2. Transform Data
3. Design & Train a Model
4. Deploy Model
5. Make Inferences

Dataset → Impulse → Test → Deploy

- Dashboard
- Devices
- Data acquisition
- Impulse design
- Create impulse
- EON Tuner
- Retrain model
- Live classification
- Model testing
- Versioning
- Deployment
Edge Impulse Project Dashboard

- Collect Data
- Transform Data
- Design & Train a 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
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
Activity: Create a Keyword Spotting Dataset

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

- **Keyword #1**  Yá’át’ééh 👋
- **Keyword #2**  hágoónee’ 👋
- “Unknown” words that are not the keyword and background noise
**Activity**: Create a Keyword Spotting Dataset

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

- **Keyword #1**: Yá’át’ééh 🌟
- **Keyword #2**: hágoónee’ 🌟
- “*Unknown*” words that are not the and background noise

I’ve pre-loaded in a bunch of background noise and unknown words!

https://docs.edgeimpulse.com/docs/pre-built-datasets/keyword-spotting
Clone my starter KWS project: https://bit.ly/EASI22-KWS
Clone succeeded
You're now ready to build your next embedded Machine Learning project!
I’ve pre-loaded in a bunch of noise and unknown words!
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.
Point your phone camera at the QR code and open the link!
Connected as phone_kunh8zjd

You can collect data from this device from the **Data acquisition** page in the Edge Impulse studio.

- Collecting images?
- Collecting audio?
- Collecting motion?
Connected as phone_kunh8zjd

You can collect data from this device from the Data acquisition page in the Edge Impulse studio.

Collecting audio?
Connected as phone_kunh8zjd

You can collect data from this device from the Data acquisition page in the Edge Impulse studio.

- Collecting images?
- Collecting audio?
- Collecting motion?
Connected as phone_kunh8zjd

You can collect data from this device from the Data acquisition page in the Edge Impulse studio.

- Collecting images?
- Collecting audio?
- Collecting motion?
Connected as phone_kunh8zjd

You can collect data from this device from the Data acquisition page in the Edge Impulse studio.

- Collecting images?
- Collecting audio?
- Collecting motion?
Connected as phone_kunh8zjd

You can collect data from this device from the Data acquisition page in the Edge Impulse studio.

https://docs.edgeimpulse.com/docs/using-your-mobile-phone
**Activity**: Create a Keyword Spotting Dataset

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

- **Keyword #1**: Yá’át’ééh ✋ label: yaateeh  length: 10 seconds
- **Keyword #2**: hágoónee’ ✋ label: hagoonee

We’ll resume in 10 minutes!
<table>
<thead>
<tr>
<th>SAMPLE NAME</th>
<th>LABEL</th>
<th>ADDED</th>
<th>LENGTH</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes.30u5okgq</td>
<td>yes</td>
<td>Today, 14:24:58</td>
<td>10s</td>
<td>Rename, Edit label, Move to test set, Disable, Crop sample, Download, Delete</td>
</tr>
<tr>
<td>noise.running_tap.wav.29000</td>
<td>noise</td>
<td>Today, 11:22:57</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Yours may say 100% / 0%
One or more of the labels in your dataset have a poor train / test split. Click to learn how to rebalance your dataset.
Dataset train / test split ratio

Training data is used to train your model, and testing data is used to test your model's accuracy after training. We recommend an approximate 80/20 train/test split ratio for your data for every class (or label) in your dataset, although especially large datasets may require less testing data.

Suggested train / test split
80% / 20%

Labels in your dataset

The 'no' class has a poor train/test split ratio. To fix this, add or move samples to the training or testing data.

NO
100% / 0% (27s / 0s)

NOISE
80% / 20% (20m 22s / 5m 13s)

UNKNOWN
80% / 20% (19m 52s / 5m 7s)

YES
81% / 19% (22s / 5s)

Perform train / test split

Use this option to rebalance your data, automatically splitting items between training and testing datasets.

Warning: this action cannot be undone.
## Collected data

<table>
<thead>
<tr>
<th>SAMPLE NAME</th>
<th>LABEL</th>
<th>ADDED</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>no.30u8qcvh.s1</td>
<td>no</td>
<td>Today, 15:22:58</td>
<td>1 s</td>
</tr>
<tr>
<td>no.30u6k9u9.s5</td>
<td>no</td>
<td>Today, 15:22:58</td>
<td></td>
</tr>
<tr>
<td>no.30u6k9u9.s1</td>
<td>no</td>
<td>Today, 15:22:58</td>
<td></td>
</tr>
<tr>
<td>no.30u8qcvh.s9</td>
<td>no</td>
<td>Today, 15:22:43</td>
<td></td>
</tr>
<tr>
<td>no.30u8qcvh.s7</td>
<td>no</td>
<td>Today, 15:22:43</td>
<td></td>
</tr>
<tr>
<td>yes.30u8rq7l.s8</td>
<td>yes</td>
<td>Today, 15:20:14</td>
<td></td>
</tr>
<tr>
<td>yes.30u8rq7l.s7</td>
<td>yes</td>
<td>Today, 15:20:14</td>
<td></td>
</tr>
</tbody>
</table>

- **Rename**
- **Edit label**
- **Move to test set**
- **Disable**
- **Crop sample**
- **Split sample**
- **Download**
- **Delete**
Dataset train / test split ratio

**Training data** is used to train your model, and **testing data** is used to test your model's accuracy after training. We recommend an approximate 80/20 train/test split ratio for your data for every class (or label) in your dataset, although especially large datasets may require less testing data.

### Suggested Train / Test Split

<table>
<thead>
<tr>
<th>Class</th>
<th>Suggested Split</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>80% / 20%</td>
</tr>
</tbody>
</table>

### Labels in your dataset

The 'no' class has a poor train/test split ratio. To fix this, add or move samples to the training or testing data.

- **NO**
  - 81% / 19% (22s / 5s)
- **NOISE**
  - 80% / 20% (20m 22s / 5m 13s)
- **UNKNOWN**
  - 80% / 20% (19m 52s / 5m 7s)
- **YES**
  - 81% / 19% (22s / 5s)
Today’s Agenda

- Preprocessing for Keyword Spotting
- Convolutional Neural Networks for Image Classification
- Hands-on: KWS Data Collection with Edge Impulse
  - Hands-on: Training our Model with Edge Impulse
  - Hands-on: Testing our Model in the Real World
- Summary
Today’s Agenda

- Preprocessing for Keyword Spotting
- Convolutional Neural Networks for Image Classification
- Hands-on: KWS Data Collection with Edge Impulse
- **Hands-on: Training our Model with Edge Impulse**
- Hands-on: Testing our Model in the Real World
- Summary
Edge Impulse Project Dashboard

- Collect Data
- Transform Data
- Design & Train a 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
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.
We’ll keep things simple today and just add an MFCC but/and in future projects you can:

- create your own blocks
- use multiple blocks

https://docs.edgeimpulse.com/docs/custom-blocks
An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.
## Add a learning block

Some learning blocks have been hidden based on the data in your project.

<table>
<thead>
<tr>
<th>DESCRIPTION</th>
<th>AUTHOR</th>
<th>RECOMMENDED</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Classification (Keras)</strong></td>
<td>EdgelImpulse Inc.</td>
<td></td>
</tr>
<tr>
<td>Learns patterns from data, and can apply these to new data. Great for categorizing movement or recognizing audio.</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Regression (Keras)</strong></td>
<td>EdgelImpulse Inc.</td>
<td></td>
</tr>
<tr>
<td>Learns patterns from data, and can apply these to new data. Great for predicting numeric continuous values.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# MFC (DCT/NYUML2_KWS_TEST.Clone)

**#1 Click to set a description for this version**

**Parameters**

- **Generate features**

**Training set**

- **Data in training set:** 40m 29s
- **Classes:** 4 (no, noise, unknown, yes)
- **Training windows:** 2,429

**Feature explorer**

- **No features generated yet.**
MFCC (BTCITYML22-NIVS-TESTCLONE)

#1 ▽ Click to set a description for this version

Parameters  Generate features

Training set

Data in training set  40m 29s
Classes  4 (no, noise, unknown, yes)
Training windows  2,429

Generating features...

Feature explorer

No features generated yet.

Feature generation output  Cancel

Creating job... OK (ID: 2598741)
Scheduling job in cluster...
Job started
Creating windows from 2429 files...
[2/3] Pre-caching files...
[3/3] Pre-caching files...
Pre-caching files OK
[ 1/2429] Creating windows from files...
If you can visually see the clustering of the data then it is easier for the ML model to learn! (But its not required and provides no guarantees)
Neural Network settings

Training settings
Number of training cycles
Learning rate
Validation set size
Auto-balance dataset

Audio training options
Data augmentation

Neural network architecture

Architecture presets
1D Convolutional (Default) 2D Convolutional

Input layer (650 features)
Reshape layer (13 columns)
1D conv / pool layer (8 neurons, 3 kernel size, 1 layer)
Model Design with Edge Impulse

Pre-made neural network “blocks” that you can add!
Model Design with Edge Impulse

“Expert” mode to write your own TensorFlow code

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, InputLayer, Dropout, Conv1D, Conv2D, Flatten, Reshape, MaxPooling1D, MaxPooling2D, BatchNormalization
from tensorflow.keras.optimizers import Adam
sys.path.append('./resources/libraries')
import ei_tensorflow.training

# model architecture
model = Sequential()
channels = 1
columns = 13
rows = int(input_length / (columns * channels))
model.add(Reshape((rows, columns, channels)), input_shape=(input_length, ))
model.add(Conv2D(8, kernel_size=3, activation='relu',
kernel_constraint=tf.keras.constraints.MaxNorm(1),
padding='same'))
model.add(MaxPooling2D(pool_size=2, strides=2, padding='same'))
model.add(Dropout(0.25))
model.add(Conv2D(16, kernel_size=3, activation='relu',
kernel_constraint=tf.keras.constraints.MaxNorm(1),
padding='same'))
model.add(MaxPooling2D(pool_size=2, strides=2, padding='same'))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(classes, activation='softmax', name='y_pred'))
```

Start training
Neural network architecture

**Architecture presets** 1D Convolutional (Default) 2D Convolutional

**Input layer (650 features)**

**Reshape layer (13 columns)**

1D conv / pool layer (8 neurons, 3 kernel size, 1 layer)

**Dropout (rate 0.25)**

1D conv / pool layer (16 neurons, 3 kernel size, 1 layer)

**Dropout (rate 0.25)**

**Flatten layer**

**Add an extra layer**

**Output layer (3 features)**

---

```python
# model architecture
model.add(Conv1D(16, kernel_size=3, activation='relu', padding='same'))
model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(Dropout(0.25))
model.add(Conv1D(32, kernel_size=3, activation='relu', padding='same'))
model.add(MaxPooling1D(pool_size=2, strides=2, padding='same'))
model.add(Flatten())
model.add(Dense(32, activation='softmax', name='y_pred'))

# this controls the learning rate
opt = Adam(lr=0.005, beta_1=0.9, beta_2=0.999)

# this controls the batch size, or you can manipulate the tf.data.Dataset objects yourself
BATCH_SIZE = 32
train_dataset = train_dataset.batch(BATCH_SIZE, drop_remainder=False)
validation_dataset = validation_dataset.batch(BATCH_SIZE, drop_remainder=False)
callbacks.append(ValidationLoggerCallback(BATCH_SIZE, train_sample_count))

# train the neural network
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
model.fit(train_dataset, epochs=100, validation_data=validation_dataset, verbose=2, callbacks=callbacks)
```
For now just stick with the defaults but/and you can easily design any model you want and use any optimizer you want using TensorFlow!
### Training output

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Info</th>
<th>Training Loss</th>
<th>Validation Loss</th>
<th>Training Accuracy</th>
<th>Validation Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>95/100</td>
<td>4/4 - 0s</td>
<td>0.1044</td>
<td>0.2934</td>
<td>0.9500</td>
<td>0.9231</td>
</tr>
<tr>
<td>96/100</td>
<td>4/4 - 0s</td>
<td>0.0256</td>
<td>0.3830</td>
<td>1.0000</td>
<td>0.8846</td>
</tr>
<tr>
<td>97/100</td>
<td>4/4 - 0s</td>
<td>0.0523</td>
<td>0.4366</td>
<td>0.9800</td>
<td>0.8462</td>
</tr>
<tr>
<td>98/100</td>
<td>4/4 - 0s</td>
<td>0.0451</td>
<td>0.4265</td>
<td>0.9800</td>
<td>0.8846</td>
</tr>
<tr>
<td>99/100</td>
<td>4/4 - 0s</td>
<td>0.0514</td>
<td>0.3926</td>
<td>0.9900</td>
<td>0.8846</td>
</tr>
<tr>
<td>100/100</td>
<td>4/4 - 0s</td>
<td>0.0348</td>
<td>0.3571</td>
<td>0.9900</td>
<td>0.9231</td>
</tr>
</tbody>
</table>

Finished training.

---

**Training Set**

**Validation Set**
Final Accuracy
## Confusion Matrix

<table>
<thead>
<tr>
<th>Predicted Output = Yes</th>
<th>Actual Output = Yes</th>
<th>Actual Output = No</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of True Positive</td>
<td># of False Positive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Type 1 Error</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Predicted Output = No</th>
<th># of False Negative</th>
<th># of True Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Type 2 Error</td>
<td></td>
</tr>
</tbody>
</table>
Final Accuracy

Accuracy Breakdown

Feature Explorer

Individual Data Points
Final Accuracy
96.6%

Accuracy Breakdown

Feature Explorer

Individual Data Points

Expected runtime/memory

On-device performance
11 ms.
5.0K
34.8K
Edge Impulse Project Dashboard

- Collect Data
- Transform Data
- Design & Train a 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

- Preprocessing for Keyword Spotting
- Convolutional Neural Networks for Image Classification
- Hands-on: KWS Data Collection with Edge Impulse
- Hands-on: Training our Model with Edge Impulse
- Hands-on: Testing our Model in the Real World
- Summary
Today’s Agenda

- Preprocessing for Keyword Spotting
- Convolutional Neural Networks for Image Classification
- Hands-on: KWS Data Collection with Edge Impulse
- Hands-on: Training our Model with Edge Impulse
- Hands-on: Testing our Model in the Real World
- Summary
Edge Impulse Project Dashboard
Connected as phone_kunh8zjd

You can collect data from this device from the Data acquisition page in the Edge Impulse studio.

- Collecting images?
- Collecting audio?
- Collecting motion?

Switch to classification mode

Collect data

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.

Show QR code

Connect a new device
Connected as phone_kunh8zjd

You can collect data from this device from the Data acquisition page in the Edge Impulse studio.

- Collecting images?
- Collecting audio?
- Collecting motion?

Switch to classification mode

Building project...

Job started
Deploy and Test your Model

Shows the score for (confidence that the current sounds is) each of the various keywords and unknown and bolds the highest score.
Today’s Agenda

- Preprocessing for Keyword Spotting
- Convolutional Neural Networks for Image Classification
- Hands-on: KWS Data Collection with Edge Impulse
- Hands-on: Training our Model with Edge Impulse
- Hands-on: Testing our Model in the Real World
- Summary
Today’s Agenda

- Preprocessing for Keyword Spotting
- Convolutional Neural Networks for Image Classification
- Hands-on: KWS Data Collection with Edge Impulse
- Hands-on: Training our Model with Edge Impulse
- Hands-on: Testing our Model in the Real World

Summary
Machine Learning

WE PROVIDE
ANSWERS
Aka the Labels of the Data

INPUTS

THE COMPUTER LEARNS
RULES
Deep Learning with Neural Networks
Features can be found with Convolutions
The (Tiny) Machine Learning Workflow

Collect & Transform Data

Design & Train a Model

Deploy Model

Make Inferences

If ML is going to be everywhere we need to consider how to best collect GOOD data RESPONSIBLY
The (Tiny) Machine Learning Workflow

Collect & Transform Data

Design & Train a Model

Deploy Model

Make Inferences

No Loud

No Loud

No Loud
The (Tiny) Machine Learning **Workflow**

1. Collect & Transform Data
2. Design & Train a Model
3. Deploy Model
4. Make Inferences
The (Tiny) Machine Learning Workflow

1. Collect Data
2. Transform Data
3. Design & Train a Model
4. Deploy Model
5. Make Inferences

Dataset → Impulse → Test → Deploy

Edge Impulse Simplifies Training and Deployment
Better Data = Better Models!
see you again at 12pm (Mountain Time)