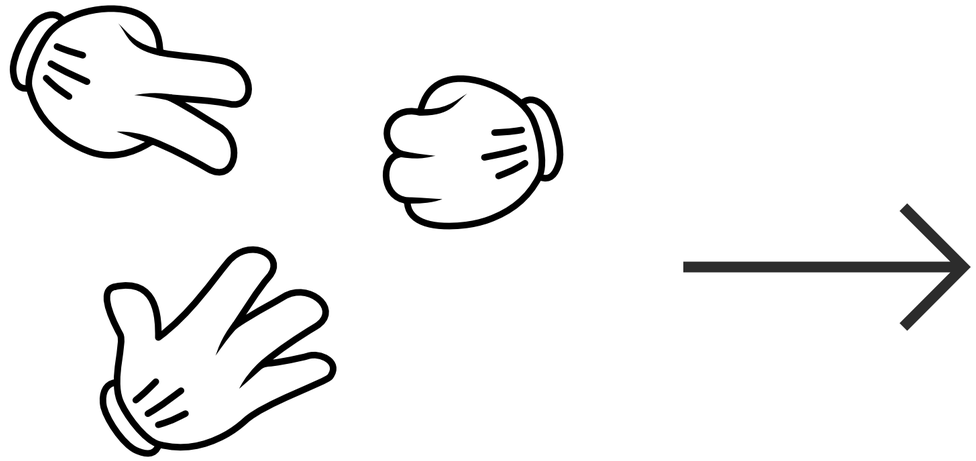
Rock Paper Scissors with Computer!



CASA0018 Coursework presentation

Agenda →

Introduction

Project Idea	Create an program on Arduino IDE to complete: • Detect which gesture the user gives • Randomly generate a choice • Use built-in LED as an output signal
Inspiration	• Image classification and object detection lectures • ‘How to win paper rock and scissors in Dr. Kawashima’s Brain Training’ game on Switch
Aim	Reach an accuracy up to 80%

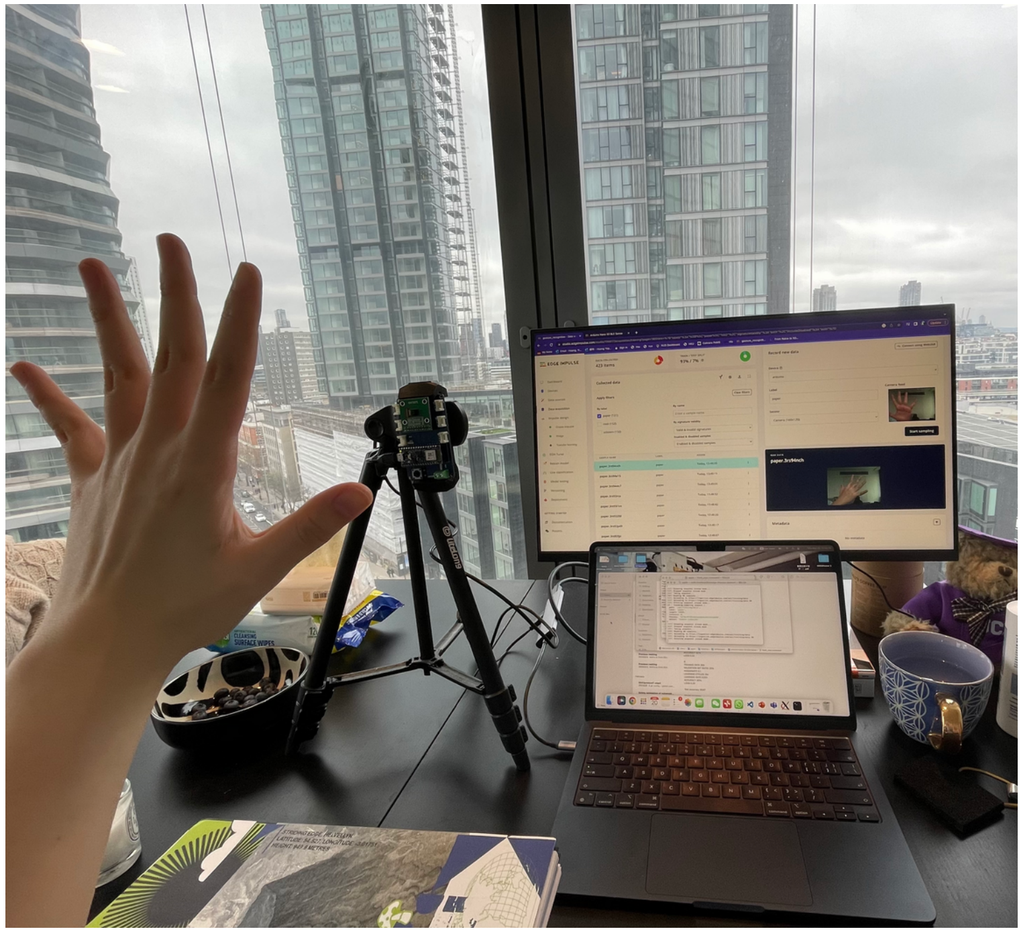
Back to Agenda Page

2.1 Data Collection

Back to Agenda Page

- Collected, labeled, split by Edge Impulse
- Dataset size 423 training/74 testing (85%/15%)
  - Paper 121 training/21 testing
  - Rock 152 training/27 testing
  - Scissors 150 training/26 testing

- Considered control variables in images:
  - Background
  - Left and right hands
  - Gestures by different people

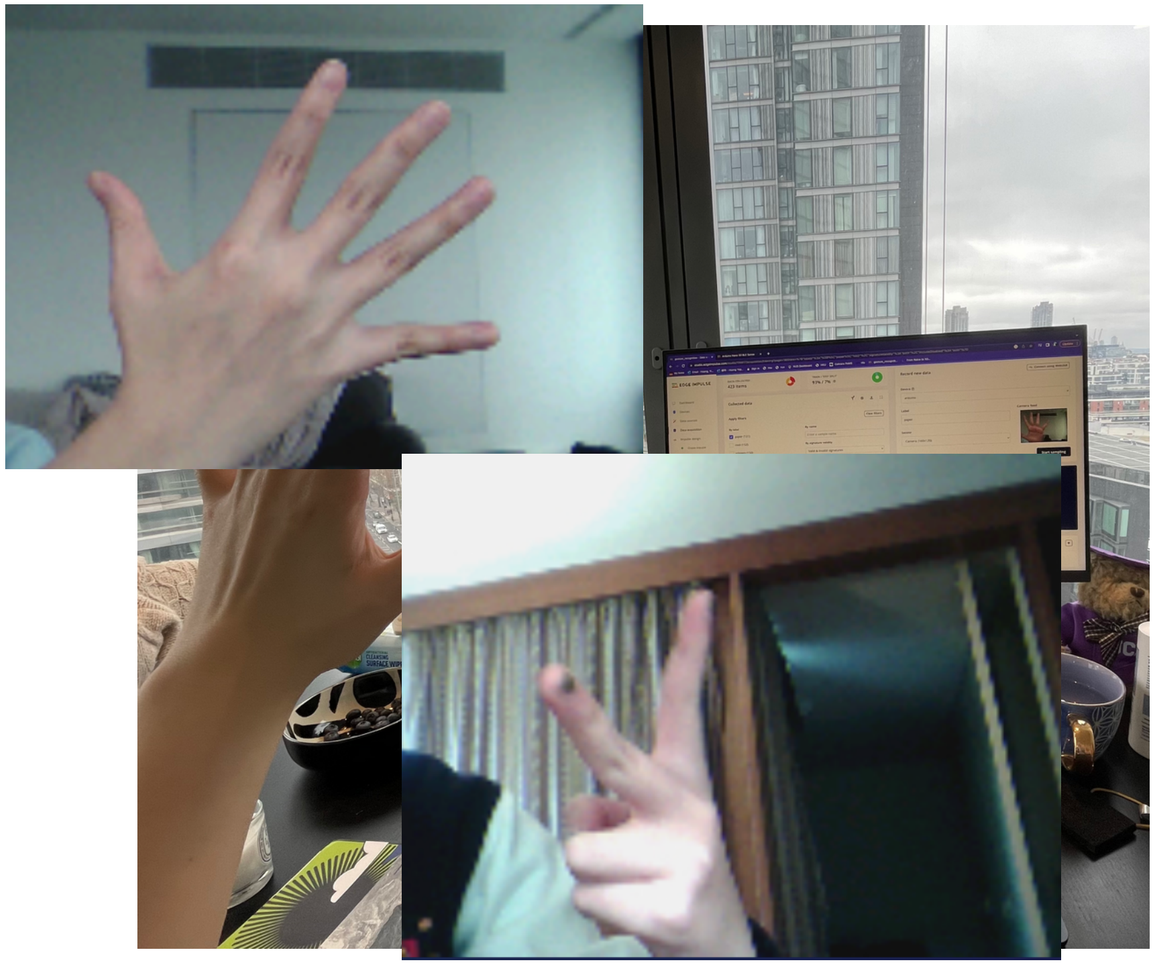


2.1 Data Collection →

Back to Agenda Page

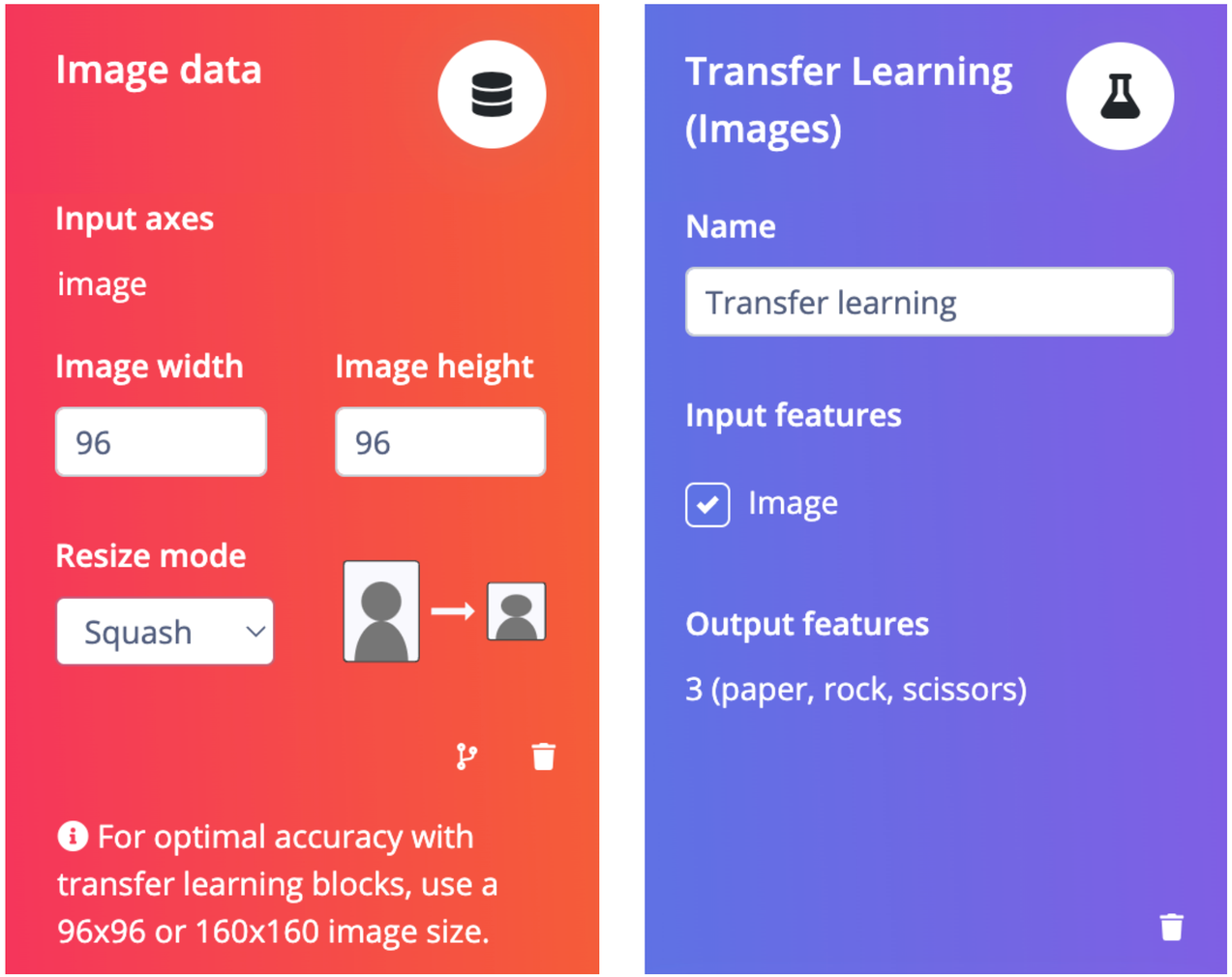
- Collected, labeled, split by Edge Impulse
- Dataset size 423 training/74 testing (85%/15%)
  - Paper 121 training/21 testing
  - Rock 152 training/27 testing
  - Scissors 150 training/26 testing

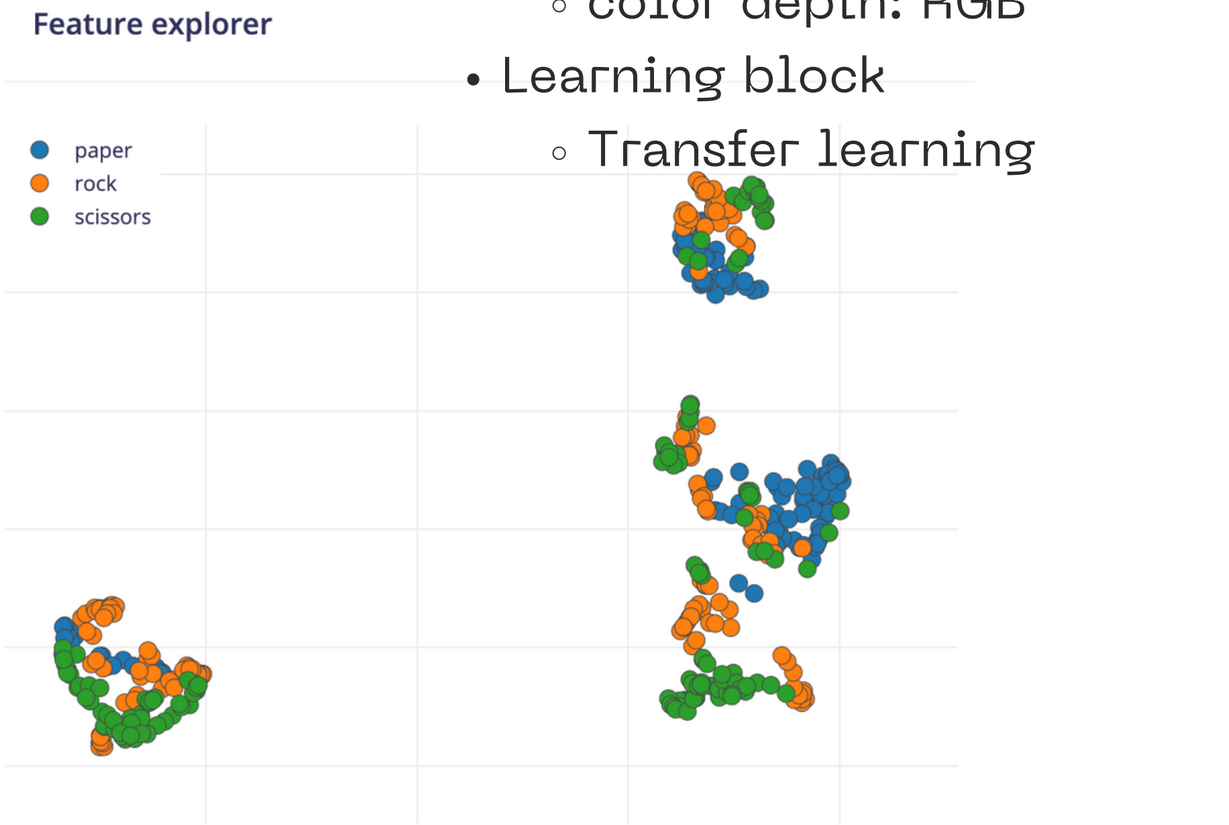
- Considered control variables in images:
  - Background
  - Left and right hands
  - Gestures by different people



2.2 Model building

Back to Agenda Page





- Parameters setting
  - image size: 96 x 96
  - resize mode: squash
  - color depth: RGB
- Learning block
  - Transfer learning

2.3 Model training and tesing

Back to Agenda Page

Neural Network settings

Training settings

Number of training cycles: 20

Learning rate: 0.005

Validation set size: 20 %

Auto-balance dataset: ☑

Data augmentation: ☑

Neural network architecture

Input layer (27,648 features)

MobileNetV2 96x96 0.35 (final layer: 8 neurons, 0.1 dropout)

Choose a different model

Output layer (3 classes)

Last training performance (validation set)

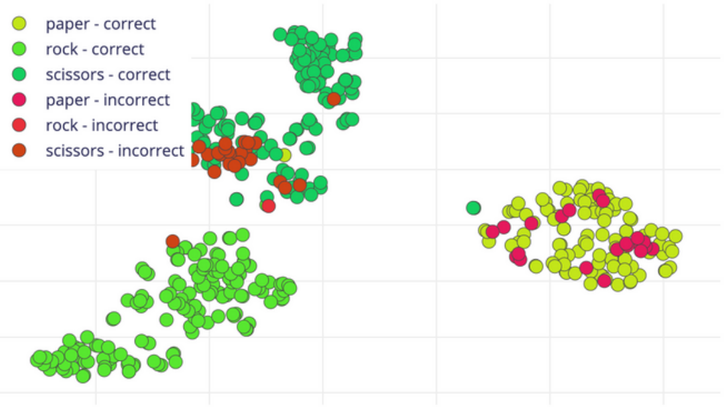
ACCURACY 91.8%

LOSS 0.23

Confusion matrix (validation set)

	PAPER	ROCK	SCISSORS
PAPER	92.3%	0%	7.7%
ROCK	0%	100%	0%
SCISSORS	0%	15.2%	84.8%
F1 SCORE	0.96	0.91	0.89

Data explorer (full training set)



On-device performance

INFERENCING... 1,974 ms.

PEAK RAM US... 346.9K

FLASH USAGE 570.5K

Model testing results

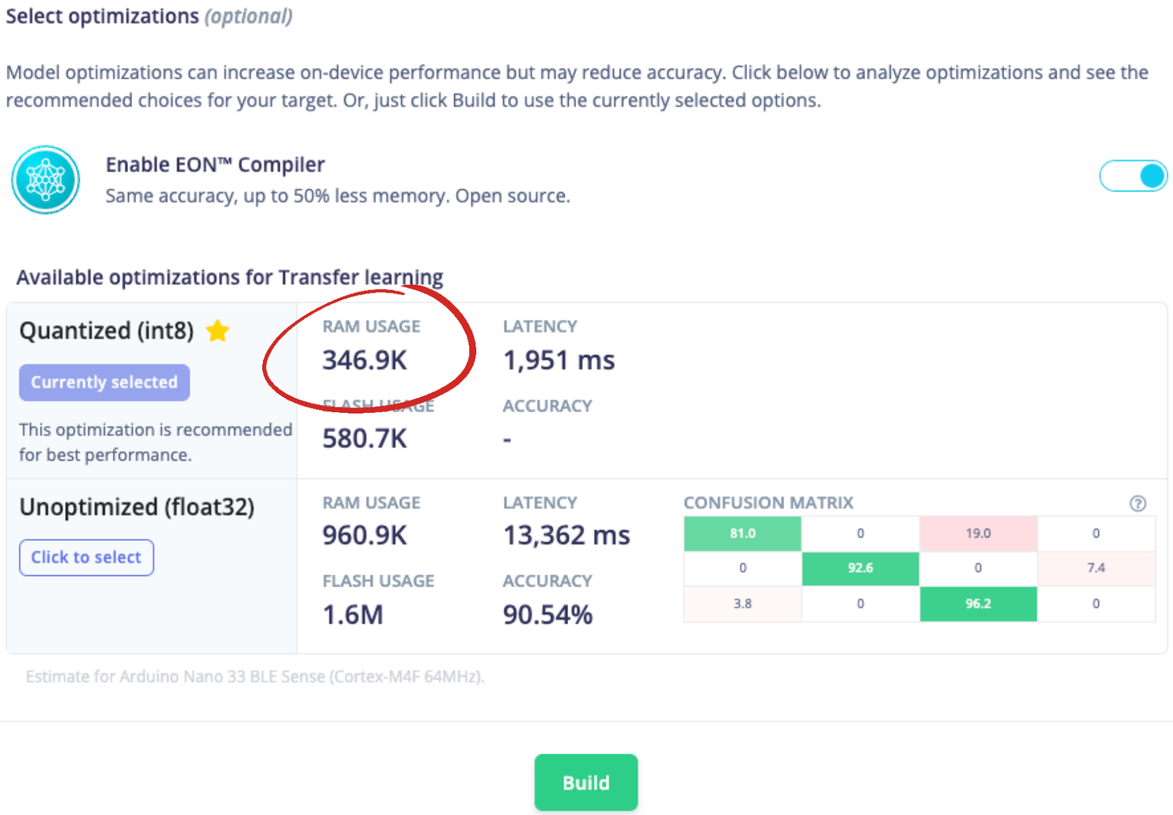
ACCURACY 86.49%

	PAPER	ROCK	SCISSORS	UNCERTAIN
PAPER	71.4%	0%	23.8%	4.8%
ROCK	11.1%	85.2%	0%	3.7%
SCISSORS	0%	0%	100%	0%
F1 SCORE	0.77	0.92	0.91	

Feature explorer

2.4 Model Deployment →

Back to Agenda Page



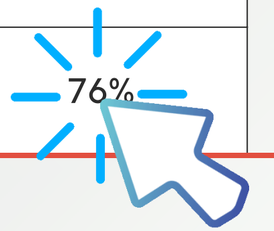
- Exceeds the model's maximum size: 256KB

2.5 Model parameters tuning

Back to Agenda Page

- With the help of EON tuner, different models with different parameters were experimeted

	Learning type	Image input	color depth	learning rate	epochs	Accuracy (validation)	Accuracy (test)
1	Classification	32 x 32	grayscale	0.001	50	89%	55%
2	Classification	32 x 32	RGB	0.001	50	87%	45%
3	Transfer Learning	96 x 96	RGB	0.0005	50	84%	62%
4	Transfer Learning	96 x 96	grayscale	0.005	50	75%	62%
5	Transfer Learning	96 x 96	RGB	0.005	50	73%	76%



2.6 Final optimal model

Back to Agenda Page

Neural Network settings

Training settings

Number of training cycles ⓘ 50

Learning rate ⓘ 0.005

Validation set size ⓘ 20 %

Auto-balance dataset ⓘ ☑

Data augmentation ⓘ ☑

Neural network architecture

Input layer (27,648 features)

MobileNetV1 96x96 0.1 (final layer: 32 neurons, 0.2 dropout)

Choose a different model

Output layer (3 classes)

Neural Network settings

Training settings

Validation set size ⓘ 20 %

Neural network architecture

```
34  base_model = tf.keras.applications.MobileNet(
35      input_shape = INPUT_SHAPE,
36      weights = WEIGHTS_PATH,
37      alpha = 0.1
38  )
39
40  base_model.trainable = True
41
42  model = Sequential()
43  model.add(InputLayer(input_shape=INPUT_SHAPE, name='x_input'))
44  # Don't include the base model's top layers
45  last_layer_index = -5
46  model.add(Model(inputs=base_model.inputs, outputs=base_model
        .layers[last_layer_index].output))
47  model.add(Reshape((-1, model.layers[-1].output.shape[3])))
48  model.add(Dense(32, activation='relu'))
49  model.add(Dropout(0.2))
50  model.add(Flatten())
51  model.add(Dense(classes, activation='softmax'))
52
53
54
55  BATCH_SIZE = 32
56  EPOCHS = args.epochs or 50
57  LEARNING_RATE = args.learning_rate or 0.005
58  train_dataset = train_dataset.batch(BATCH_SIZE, drop_remainder
        =False)
59  validation_dataset = validation_dataset.batch(BATCH_SIZE,
        drop_remainder=False)
60  callbacks.append(BatchLoggerCallback(BATCH_SIZE,
        train_sample_count, epochs=EPOCHS))
```

Last training performance (validation set)

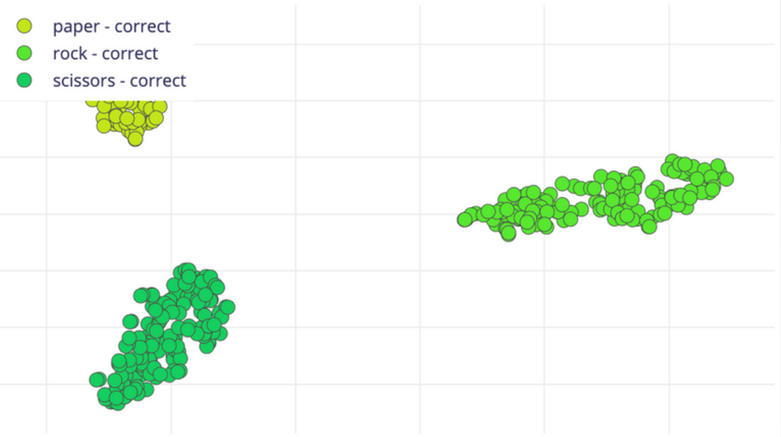
ACCURACY 100.0%

LOSS 0.00

Confusion matrix (validation set)

	PAPER	ROCK	SCISSORS
PAPER	100%	0%	0%
ROCK	0%	100%	0%
SCISSORS	0%	0%	100%
F1 SCORE	1.00	1.00	1.00

Data explorer (full training set) ⓘ



On-device performance ⓘ

INFERENCING... 344 ms.

PEAK RAM US... 66.9K

FLASH USAGE 111.2K

2.6 Final optimal model

Back to Agenda Page

Neural Network settings

Training settings

Number of training cycles: 50

Learning rate: 0.005

Validation set size: 20 %

Auto-balance dataset: ☑

Data augmentation: ☑

Neural network architecture

Input layer (27,648 features)

MobileNetV1 96x96 0.1 (final layer: 32 neurons, 0.2 dropout)

Choose a different model

Output layer (3 classes)

Neural Network settir

Training settings

Validation set size

Neural network architec

```
34  base_model = tf.
35      input_shape =
36      weights = WE
37      alpha = 0.1
38  )
39
40  base_model.train
41
42  model = Sequenti
43  model.add(InputL
44  # Don't include
45  last_layer_index
46  model.add(Model(
        .layers[last_
47  model.add(Reshap
48  model.add(Dense(
49  model.add(Dropou
50  model.add(Flatte
51  model.add(Dense(
52
53
54
55  BATCH_SIZE = 32
56  EPOCHS = args.ep
57  LEARNING_RATE =
58  train_dataset =
        =False)
59  validation_datas
        drop_remainde
60  callbacks.append
        train_sample_
```

Model testing results

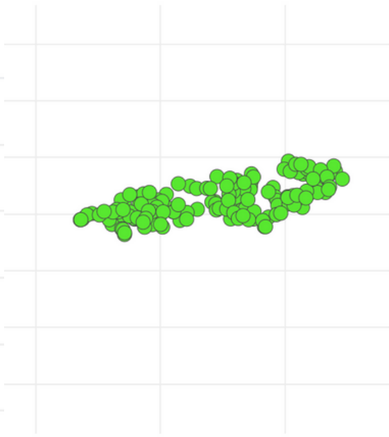
ACCURACY
83.78%

	PAPER	ROCK	SCISSORS	UNCERTAIN
PAPER	90.5%	0%	9.5%	0%
ROCK	18.5%	81.5%	0%	0%
SCISSORS	7.7%	11.5%	80.8%	0%
F1 SCORE	0.81	0.85	0.86	

Feature explorer

LOSS
0.00

ROCK	SCISSORS
0%	0%
100%	0%
0%	100%
1.00	1.00



AK RAM US...
5.9K

FLASH USAGE
111.2K

2.7 Program building

Back to Agenda Page

```
// find the label with the highest probability
float max_prob = 0;
const char* max_label = "";
for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
    if (result.classification[ix].value > max_prob) {
        max_prob = result.classification[ix].value;
        max_label = result.classification[ix].label;
    }
}

 // turn the LED off by making the voltage LOW



ei_printf("Your choice: %s (%.5f)\n", max_label, max_prob);
```

```
// generate a random choice and compare it with the max_label
const char* options[] = {"rock", "paper", "scissor"};
const char* random_choice = random_string(options, 3);
ei_printf("Randomly generated choice: %s\n", random_choice);
if (strcmp(max_label, random_choice) == 0) {
    ei_printf("It's a tie!\n");
    digitalWrite(GREEN, HIGH);
    delay(1000);
    digitalWrite(BLUE, LOW);
    digitalWrite(RED, LOW);
    digitalWrite(GREEN, LOW);

} else if (strcmp(max_label, "rock") == 0 && strcmp(random_choice, "scissor") == 0 ||
           strcmp(max_label, "paper") == 0 && strcmp(random_choice, "rock") == 0 ||
           strcmp(max_label, "scissor") == 0 && strcmp(random_choice, "paper") == 0) {
    ei_printf("You win!\n");
    digitalWrite(RED, HIGH);
    delay(1000);
    digitalWrite(GREEN, LOW);
    digitalWrite(RED, LOW);
    digitalWrite(BLUE, LOW);

} else {
    ei_printf("You lose!\n");
    digitalWrite(BLUE, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000);
    digitalWrite(RED, LOW);
    digitalWrite(GREEN, LOW);
    digitalWrite(BLUE, LOW);


}
```

- Print the label with highest probability
- Compare it with computer generated choice
- Output LED signal

3. Demostration

→

Back to Agenda Page

4. Limitations and future works

- Other gesture
- Detailed parameters modification
- Trade off between size and performances of the model

Back to Agenda Page