



Revision: 04

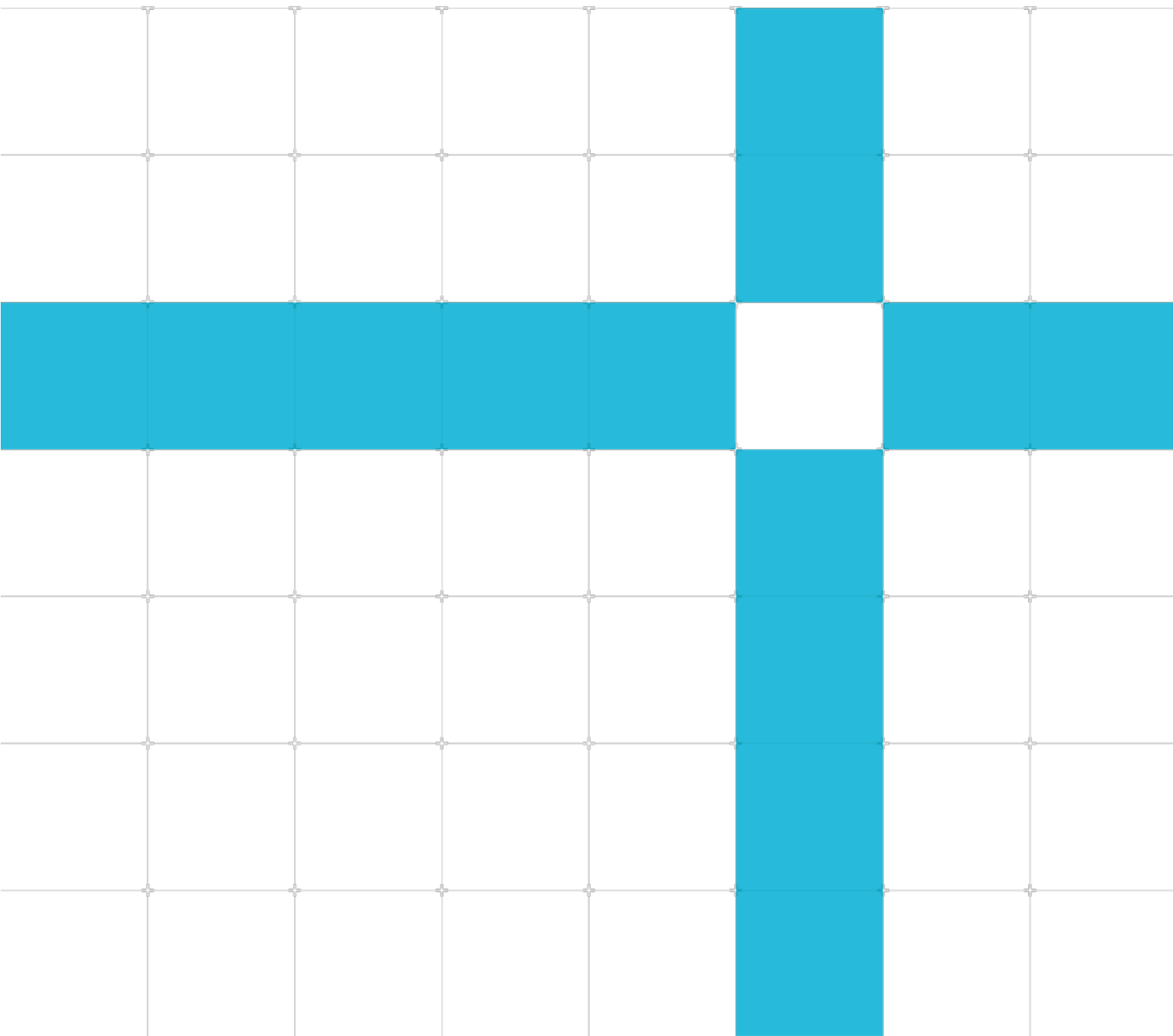
# Teach your Raspberry Pi, Episode 1: Yeah, World

Non-Confidential

Copyright © 2017-2020 Arm Limited (or its affiliates).  
All rights reserved.

**Issue 04**

ARM-ECM-0735791



## Teach your Raspberry Pi, Episode 1: Yeah, World

Copyright © 2017-2020 Arm Limited (or its affiliates). All rights reserved.

### Release information

### Document history

Issue	Date	Confidentiality	Change
01	19 December 2017	Non-Confidential	First release.
02	16 April 2018	Non-Confidential	Minor fixes to the Install TensorFlow section.
03	18 December 2019	Non-Confidential	Raspberry Pi 4 and Python 3 support added.
04	30 November 2020	Non-Confidential	Minor change to Install Library step.

## Confidential Proprietary Notice

This document is **CONFIDENTIAL** and any use by you is subject to the terms of the agreement between you and Arm or the terms of the agreement between you and the party authorized by Arm to disclose this document to you.

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. **No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.**

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information: **(i)** for the purposes of determining whether implementations infringe any third party patents; **(ii)** for developing technology or products which avoid any of Arm's intellectual property; or **(iii)** as a reference for modifying existing patents or patent applications or creating any continuation, continuation in part, or extension of existing patents or patent applications; or **(iv)** for generating data for publication or disclosure to third parties, which compares the performance or functionality of the Arm technology described in this document with any other products created by you or a third party, without obtaining Arm's prior written consent.

THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws.

Use of the word “partner” in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm's trademark usage guidelines at <http://www.arm.com/company/policies/trademarks>.

Copyright © 2017-2020 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20348)

## Confidentiality Status

### Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm. No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether implementations infringe any third party patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

Copyright © 2017-2020 Arm Limited (or its affiliates). All rights reserved.

Non-Confidential

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

The Arm corporate logo and words marked with ® or ™ are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm's trademark usage guidelines at <http://www.arm.com/company/policies/trademarks>.

Copyright © 2017-2020 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

## Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

## Product Status

The information in this document is Final, that is for a developed product.

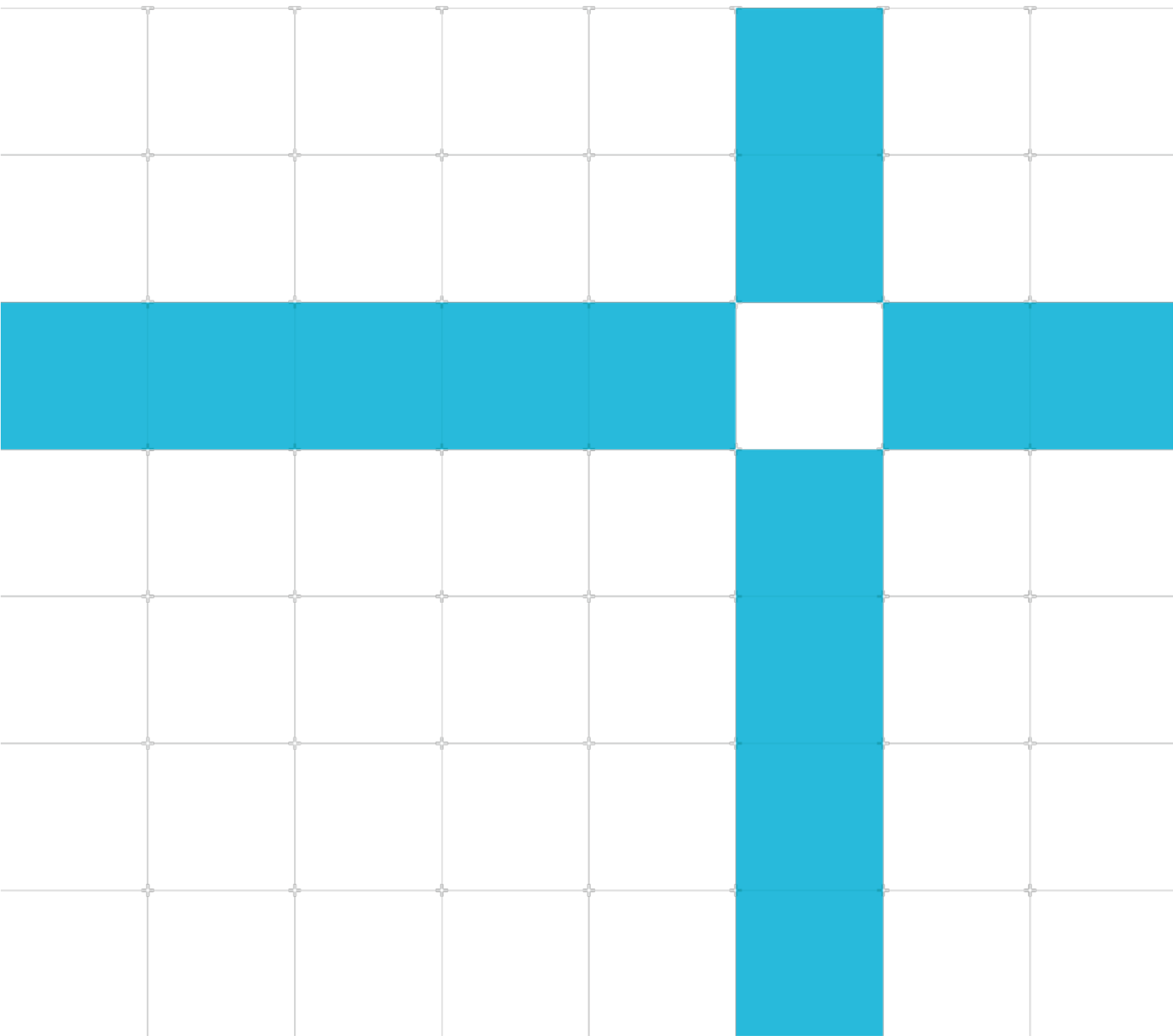
## Web Address

[developer.arm.com](http://developer.arm.com)

## Progressive terminology commitment

This document includes terms that can be offensive. We will replace these terms in a future issue of this document. If you find offensive terms in this document, please contact [terms@arm.com](mailto:terms@arm.com).

arm



# Contents

<b>1 Introduction.....</b>	<b>7</b>
1.1 Product revision status.....	7
1.2 Intended audience.....	7
1.3 Conventions .....	7
1.3.1 Glossary.....	7
1.3.2 Typographical conventions .....	8
1.4 Feedback.....	9
1.4.1 Feedback on this product.....	9
1.4.2 Feedback on content .....	9
<b>2 Overview .....</b>	<b>10</b>
<b>3 Before you begin.....</b>	<b>11</b>
<b>4 Install libraries.....</b>	<b>12</b>
4.1 Install TensorFlow.....	12
4.2 Install the Arm training scripts.....	12
4.3 Advanced information .....	12
<b>5 Train the AI with your own data.....</b>	<b>14</b>
<b>6 Set up your camera and environment .....</b>	<b>15</b>
6.1 Teaching by example .....	16
<b>7 Record data.....</b>	<b>17</b>
<b>8 Train a network on this data .....</b>	<b>19</b>
8.1 What is going on in this process?.....	19
8.2 Advanced information .....	20
<b>9 Run your new network .....</b>	<b>21</b>
<b>10 Next steps.....</b>	<b>22</b>
<b>Appendix A Revisions.....</b>	<b>23</b>

# 1 Introduction

## 1.1 Product revision status

The rpxy identifier indicates the revision status of the product described in this book, for example, r1p2, where:

rx

Identifies the major revision of the product, for example, r1.

Py

Identifies the minor revision or modification status of the product, for example, p2.

## 1.2 Intended audience

This guide is intended for developers who want to learn how to use the underlying Arm hardware in Raspberry Pis to run ML processes.

## 1.3 Conventions







The following subsections describe conventions used in Arm documents.

### 1.3.1 Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: <https://developer.arm.com/glossary>.

## 1.3.2 Typographical conventions

Convention	Use
<i>italic</i>	Introduces citations.
<b>bold</b>	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
monospace	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
monospace <b>bold</b>	Denotes language keywords when used outside example code.
monospace <u>underline</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <pre>MRC p15, 0, &lt;Rd&gt;, &lt;CRn&gt;, &lt;CRm&gt;, &lt;Opcode_2&gt;</pre>
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the Arm® Glossary. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.
 Caution	This represents a recommendation which, if not followed, might lead to system failure or damage.
 Warning	This represents a requirement for the system that, if not followed, might result in system failure or damage.
 Danger	This represents a requirement for the system that, if not followed, will result in system failure or damage.
 Note	This represents an important piece of information that needs your attention.
 Tip	This represents a useful tip that might make it easier, better or faster to perform a task.
 Remember	This is a reminder of something important that relates to the information you are reading.



## 1.4 Feedback

Arm welcomes feedback on this product and its documentation.

### 1.4.1 Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

### 1.4.2 Feedback on content

If you have comments on content, send an email to [errata@arm.com](mailto:errata@arm.com) and give:

- The title Teach your Raspberry Pi, Episode 1: Yeah, World.
- The number ARM-ECM-0735791.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.



Note

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader and cannot guarantee the quality of the represented document when used with any other PDF reader.

---

## 2 Overview

Did you know that you can train AI on your Raspberry Pi without any extra hardware or accelerators? In this guide, we use TensorFlow to train a Raspberry Pi to burst into applause whenever you raise your hands in the air using nothing more than a camera and the on-board Arm CPU of the Pi. This activity even works on the Pi Zero.

# 3 Before you begin

This guide assumes that you have:

- A Raspberry Pi. We have tested this activity with a Pi 4 Model B, Pi 3, and a Pi Zero.
- A camera module for the Pi. We have built this guide using a [Zero Cam](#).
- A fully functioning Pi running Raspbian that is connected to the Internet.
- Some basic Pi experience.
- The ability to connect and control your Raspberry Pi, either:
  - directly.
  - or via a Secure Socket Shell (SSH) which gives you a secure way to access your Raspberry Pi from your host machine over a network. To enable SSH on your Raspberry Pi, follow these [remote access](#) instructions.
  - or via Virtual Network Computing (VNC) which gives you remote access to your Raspberry Pi graphical desktop. To set up VNC on your Raspberry Pi, follow these [remote control](#) instructions.

This [video introduction](#) shows what you can achieve.

The next step is to install the libraries and download the scripts that are needed to train a neural network. Although all scripts can be run in either the Python 2 or Python 3 environment, we recommend using Python 3.

# 4 Install libraries

The next step is to install the libraries and download the scripts that are needed to train a neural network.

## 4.1 Install TensorFlow

For a Raspbian base install, the only dependency that you need to add is TensorFlow.

1. Install some TensorFlow prerequisites which include Python 3. Open a new terminal session and enter the following on the command line:

```
sudo apt update
sudo apt install python3-dev python3-pip
sudo apt install libatlas-base-dev      # required for numpy
sudo apt install libhdf5-dev
sudo pip3 install -U virtualenv         # system-wide install
```

2. Install the latest version of TensorFlow:

```
sudo pip3 install --upgrade tensorflow
```

3. If you get a “Memory Error” message during the TensorFlow install, disable the cache during installation with this command:

```
sudo pip3 install --no-cache-dir tensorflow
```

4. Verify the install with this command:

```
python3 -c "import tensorflow as tf; tf.enable_eager_execution();
print(tf.reduce_sum(tf.random_normal([1000, 1000])))"
```

If the TensorFlow installation is not verified, follow the guidance at [Install TensorFlow with pip page](#) for Raspberry Pi with Python 3 as some of the requirements may have been updated.

We estimate that the installation will take 30-60 minutes.

## 4.2 Install the Arm training scripts

Download or clone our [ML examples repository](#) from GitHub.

Navigate to the yeah-world directory by entering the following commands:

```
git clone https://github.com/ARM-software/ML-examples.git
cd ML-examples/yeah-world
```

There is nothing special about these scripts, they are easy to understand and modify. Feel free to explore and hack them with your own changes.

## 4.3 Advanced information

The Python source code is easy to follow and is split across these three files:

Copyright © 2017-2020 Arm Limited (or its affiliates). All rights reserved.  
Non-Confidential

- `record.py` captures images from the camera and saves them to disk.
- `train.py` loads saved images, converts them to features and trains a classifier on those features.
- `run.py` captures images from the camera, converts them to features, and classifies them. A random sound is also played if the image is of someone doing a cheering gesture.

These three helper files keep the three source code files as readable as possible:

- `camera.py` initializes the picamera module and optionally fluctuates the exposure and white balance during recording.
- `pinet.py` loads the pretrained MobileNet with TensorFlow and uses it to convert one image at a time into a set of features.
- `randomsound.py` uses pygame to play a random sound file from a directory.

# 5 Train the AI with your own data

Teaching a neural network to reliably recognize gestures from every human at every location and in every light condition presents a huge challenge. Meeting that challenge requires lots of varied examples and careful training. However, teaching a network to recognize your gestures in your environment is easy.

In fact, training your neural network in this way is so easy that we can do it right now on the Raspberry Pi itself. We can do this training with no GPU or accelerator to help with the required processing.

The training process has three steps:

1. Set up your camera and environment.
2. Record some new data.
3. Train a network on the data.

# 6 Set up your camera and environment

Neural networks can learn to detect and generalize small gestural changes that only affect a few dozen pixels on the screen, but it takes much more data and training time than we want to spend.

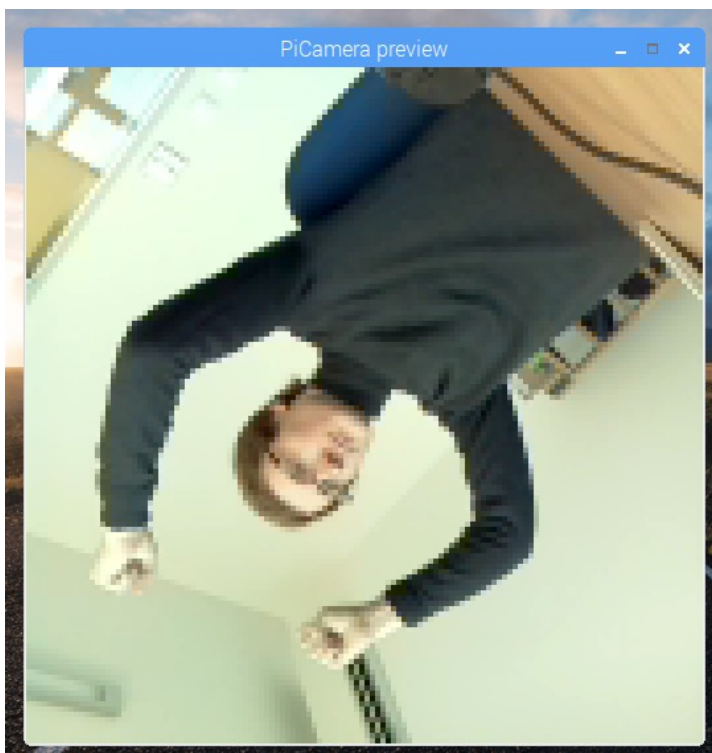
To make it easier for the neural network to learn from limited data, place the camera so that:

- Your gestures are roughly in the center of the camera view.
- You are close enough to the camera that your gesture affects a significant number of pixels.

A good rule is to place the camera in front of you so that your hands can reach the edges of the frame when you stretch your hands out in front of you. To preview what the record and run scripts will see, use the `preview.py` script with this command:

```
python3 preview.py
```

The setup in this example preview is acceptable:



It does not matter if the camera is the correct way up as long as its position remains consistent. For best results, ensure that:

- There is not too much activity happening in the background.
- There is good lighting and the camera has a good view of the subject.
- Your arms are in the picture even when they are extended.

When you are happy with the positioning of the camera and the lighting, either close the preview window or press ctrl-c at the console to exit the preview app.

## 6.1 Teaching by example

An AI is only as good as its data. When we train an AI, we teach it by example. In this case, you are giving the AI one set of pictures with your hands up and another set of pictures with your hands down. Each frame of video is one of these pictures.

Ideally, the AI learns that the principal difference between the two sets of pictures is the position of your hands. However, the AI might learn to recognize other things, for example:

- The angle at which you are leaning. This is because people often lean back slightly when they are seated and raising their hands.
- The overall lighting is brighter or dimmer. This is because your raised hands might cast a shadow on the camera when you raise them.
- If someone in the background is working in Word or watching YouTube, their monitor might change.

To help you get good results, here are some tips:

- Record variations of the gesture. With your hands in the air lean left and right, wave them higher and lower, twist left and right. Recording variations helps the network to recognize more of the slight changes that it will see in the real world.
- Record the same variations without the gesture. Try to do the same actions but with your hands down. This teaches the neural network that the chief difference between cheering and not cheering is the position of your hands, and nothing else.
- Record random things that you want the neural network to ignore. These can include things like scratching your head, standing up and walking away, jumping up and down, hiding, and covering the camera.

The most important thing is to be consistent. If you are teaching the network to detect your hands in the air, keep them up there in every frame of the hands in the air recording. Vary the position and angle of your hands, but do not lower them all the way down and then put them up again. If you do, you will include frames that look just like the hands down frames. This means that the network will assume that the difference is, for example, how your clothing is folded or how lights reflect off your glasses.



# 7 Record data

If you are using a Raspberry Pi Zero, or SSH or VNC, the live preview window can drop the frame rate during recording below 30fps. This means that less data is collected, which is a bad thing. To ensure recording is as close as possible to 30fps, run the following command in the terminal before entering further commands:

```
unset DISPLAY
```

If you keep the preview window open during training, notice that the lighting and color balance fluctuates rapidly. This fluctuation is intentional because the camera exposure and white balance are set randomly for each frame during training. This technique, which is a variant of domain randomization, shows lots of variances in something that might otherwise trick or confuse the AI: in this case, lighting and color balance. It is more common to artificially add this variance to the data at training time, but this is beyond the scope of this guide.

To record your data, follow these steps:

1. Create an example directory by entering this command:

```
mkdir example
```

- Record some actions with your hands in the air. When you run the following command, it counts down from three and then records you for 15 seconds:

```
python3 record.py example/yeah 15
```

Suggested actions: waving your hands in the air like you are cheering, lean left and right while doing so. Turn left and right. Lean forwards and backwards. Keep your hands pumping in the air above your head the whole time.

- Enter this command, then record the same behavior, but with your hands down:

```
python3 record.py example/sitting 15
```

Suggested actions: Lean left and right, turn left and right, lean forwards and backwards. Keep your hands down or at keyboard height the whole time.

- Enter this command, then record some random activities that you want the network to ignore so that these actions will not surprise the network:

```
python3 record.py example/random 15
```

Suggested actions: Cover your face, scratch your head, reach for and cover the camera, stand up and walk away, come back and sit down, get up and walk the other way.

After you have followed these steps, you will have three files of similar size that contain the three categories of gesture data that you recorded. To view the files, run the command:

```
$ ls -lh example/
```

The three files are:

```
total 166M
-rw-r--r-- 1 pi pi 49M Nov 27 10:59 random
-rw-r--r-- 1 pi pi 64M Nov 27 10:59 sitting
-rw-r--r-- 1 pi pi 54M Nov 27 10:57 yeah
```

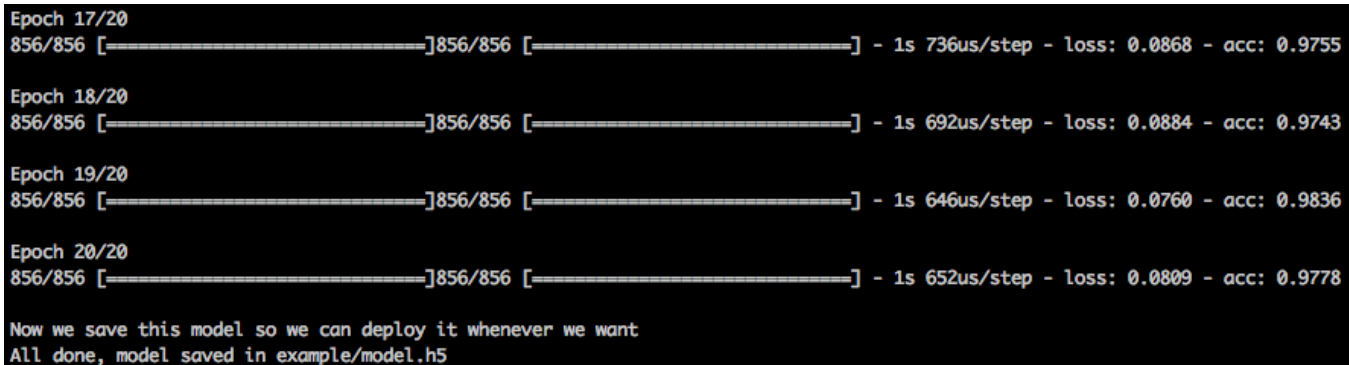
# 8 Train a network on this data

Running the `train.py` script will load your data files into memory and train a neural network to distinguish between them. There is good support in Keras and TensorFlow for training from disk without loading every example into RAM, but this script is a quick and easy way to get started. To run the script, enter the following command:

```
python3 train.py example/model.h5 example/yeah example/sitting example/random
```

The first argument is the name of the model file to save the neural network to, which is `model.h5`. You then list the different behavior video files that you have recorded. It is important that the first file is the file that contains your cheering gesture. All of the others are gestures the network should learn to ignore.

The slowest part of training is loading and converting the data. After that, the neural network should compute 20 iterations and then stop. In the example shown in this screenshot, the neural network converges



```
Epoch 17/20
856/856 [-----]856/856 [-----] - 1s 736us/step - loss: 0.0868 - acc: 0.9755

Epoch 18/20
856/856 [-----]856/856 [-----] - 1s 692us/step - loss: 0.0884 - acc: 0.9743

Epoch 19/20
856/856 [-----]856/856 [-----] - 1s 646us/step - loss: 0.0760 - acc: 0.9836

Epoch 20/20
856/856 [-----]856/856 [-----] - 1s 652us/step - loss: 0.0809 - acc: 0.9778

Now we save this model so we can deploy it whenever we want
All done, model saved in example/model.h5
```

to an accuracy of 98%.

## 8.1 What is going on in this process?

The script takes the training files in turn and loads all the frames from each file, which means that you need sufficient RAM to hold all the files. When the images are loaded, they are passed through a neural network, called MobileNet, that has been pre-trained on the ImageNet dataset.

MobileNet has already learned how to extract useful features, like edges, texture, and shapes from images. This step is the slowest part of training. If we were artificially adding noise and variation to the images, which is called domain randomization, this step would be even slower and longer. We are taking a short cut in this guide and skipping this step.

Having converted every frame into an array of features, these features are used to train a small classifier network that learns to predict the class (in this case the 0-based input file index) it came from.

Each epoch is one cycle through every frame. To prevent it latching on to overly specific things about individual frames, you add some random noise to the features. Adding noise may prevent the neural network from reaching 100% accuracy during training but makes the neural network more accurate when run on previously unseen images.

## 8.2 Advanced information

MobileNet was introduced by Google in 2017. It is a convolutional neural network that has been optimized for size and speed at the expense of accuracy. MobileNet is trained on the well-known ImageNet dataset, in which one million images are split into 1000 different classes. In this guide, we have removed the final layer that classifies images into 1000 categories removed from MobileNet. Instead, the `train.py` script uses Keras to build a new layer that classifies images into the number of categories that are passed to `train.py`.

If you read the code in the `train.py` file, you will notice that the classifier model is simple. The data is only augmented with gaussian noise instead of rotations, transforms, and occlusions. The learning rates and batch size are left at Keras defaults. A validation split with early stopping is not used. Domain randomization is applied to the data at source, and is not varied across epochs during training. This example is deliberately clean and sparse to keep it easy to understand and quick to train even on a Pi Zero. This means that you can experiment to improve your results.

# 9 Run your new network

Now you can run a neural network that is trained on your environment with a single command:

```
python3 run.py example/model.h5
```

Wave your hands in the air. Does it cheer? Even if your audio is not configured, the console prints the class predicted and displays either:

- 0: Cheering.
- 1: Sitting in view but not cheering.
- 2: Random behaviors. For example, standing, being out of the picture, and unexpected lighting conditions.

You can do a lot more than cheer. You can replace the files in the folder named sounds, and train a model to recognize different gestures, objects, or situations. For example, you can teach a Pi to:

- Greet you by name.
- Set off an alarm when someone is in view. For example, try hiding beneath a cardboard box or behind a cushion and sneaking up on it.
- Shout "THE COFFEE IS READY" when the coffee is ready.
- Tell the dog to get off the sofa when the dog gets on the sofa.

With a little python programming, you can also make your Pi do more interesting things in response to camera input, like setting GPIO pins, and sending alerts and emails. The applications are limited only by your imagination.

# 10 Next steps

This guide is the first in a series of guides about training your own gesture recognition on the Raspberry Pi using machine learning. There are some limitations to this approach:

- Changes to location, background, and even clothing can throw detection off.
- Recording longer example videos can cause out of memory errors during training.
- It is difficult to train networks to recognize subtle gestures that do not cause much visual change in the image.

You can easily extend the example code by modifying the classifier network, and by changing the actions that are performed when each category is detected. If you are already familiar with neural networks, we recommend that you add data augmentation techniques and explore different feature extraction networks like larger MobileNets and the Inception family.

We also explore these topics in the [Teach your Raspberry Pi, Episode 2: multi-gesture recognition](#) guide.

# Appendix A Revisions

This appendix describes the technical changes between released issues of this document.

**Table A-1 Issue 01**

Change	Location	Affects
First release	All	All

**Table A-2 Differences between issue 01 and issue 02**

Change	Location	Affects
Minor fixes to the Install TensorFlow section.	<a href="#">Install TensorFlow</a>	All

**Table A-3 Differences between issue 02 and issue 03**

Change	Location	Affects
Raspberry Pi 4 and Python 3 support added.	All	All

**Table A-4 Differences between issue 03 and issue 04**

Change	Location	Affects
Minor change to Install Libraries step.	<a href="#">Install Libraries</a>	All