EDGE IMPULSE

Edge AI Showcase:

Computer Vision with MobileNet and NVIDIA TAO, Powered by Alif Ensemble MCUs

Solution Guide

Introduction

This is the first in a series of guides that will take a deep dive into a high performance embedded machine learning model, running on Alif Ensemble MCUs. Learn about how these models work, what makes them suited to embedded inference, how you can train them yourself, and how to run them with speed and efficiency by leveraging the integrated NPU in Alif devices.

This article covers MobileNet, a popular family of deep learning model architectures designed for use on edge devices. MobileNet was originally published by a team at Google in 2017, and has been followed up with additional versions and papers. The first paper is titled “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”, and is available here: https://arxiv.org/abs/1704.04861

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications

Andrew G. Howard   Menglong Zhu   Bo Chen   Dmitry Kalenichenko
Weijun Wang   Tobias Weyand   Marco Andreetto   Hartwig Adam

Google Inc.
{howarda,menglong,bochen,dkalenichenko,weijunw,weyand,anm,hadam}@google.com

arXiv:1704.04861v1 [cs.CV] 17 Apr 2017

Abstract

We present a class of efficient models called MobileNets for mobile and embedded vision applications. MobileNets are based on a streamlined architecture that uses depthwise separable convolutions to build light weight deep neural networks. We introduce two simple global hyper-parameters that efficiently trade off between latency and accuracy. These hyper-parameters allow the model builder to choose the right sized model for their application based on the constraints of the problem. We present extensive experiments on resource and accuracy tradeoffs and show strong performance compared to other popular models on ImageNet classification. We then demonstrate the effectiveness of MobileNets across a wide range of applications and use cases including object detection, finegrain classification, face attributes and large scale geo-localization.

1. Introduction

Convolutional neural networks have become ubiquitous in computer vision ever since AlexNet [19] popularized deep convolutional neural networks by winning the ImageNet Challenge: ILSVRC 2012 [24]. The general trend has been to make deeper and more complicated networks in order to achieve higher accuracy [27, 31, 29, 8]. However, these advances to improve accuracy are not necessarily making networks more efficient with respect to size and speed. In many real world applications such as robotics, self-driving car and augmented reality, the recognition tasks need to be carried out in a timely fashion on a computationally limited platform.

This paper describes an efficient network architecture and a set of two hyper-parameters in order to build very small, low latency models that can be easily matched to the design requirements for mobile and embedded vision applications. Section 2 reviews prior work in building small models. Section 3 describes the MobileNet architecture and two hyper-parameters width multiplier and resolution multiplier to define smaller and more efficient MobileNets. Section 4 describes experiments on ImageNet as well a variety of different applications and use cases. Section 5 closes with a summary and conclusion.

2. Prior Work

There has been rising interest in building small and efficient neural networks in the recent literature, e.g. [16, 34, 12, 36, 22]. Many different approaches can be generally categorized into either compressing pretrained networks or training small networks directly. This paper proposes a class of network architectures that allows a model developer to specifically choose a small network that matches the resource restrictions (latency, size) for their application. MobileNets primarily focus on optimizing for latency but also yield small networks. Many papers on small networks focus only on size but do not consider speed.

MobileNets are built primarily from depthwise separable convolutions initially introduced in [26] and subsequently used in Inception models [13] to reduce the computation in the first few layers. Flattened networks [16] build a network out of fully factorized convolutions and showed the potential of extremely factorized networks. Independent of this current paper, Factorized Networks[34] introduces a similar factorized convolution as well as the use of topological connections. Subsequently, the Xception network [3] demonstrated how to scale up depthwise separable filters to out perform Inception V3 networks. Another small network is Squeezenet [12] which uses a bottleneck approach to design a very small network. Other reduced computation networks include structured transform networks [28] and deep fried convnets [37].

A different approach for obtaining small networks is shrinking, factorizing or compressing pretrained networks. Compression based on product quantization [36], hashing

1

What is MobileNet?

MobileNet is an example of a convolutional neural network, or CNN. CNNs are designed to efficiently extract features from images. A typical CNN consists of a stack of convolutional layers. The first layer takes the image as input in the form of a two-dimensional matrix. The convolutional layer "slides" a set of filters across the matrix, "striding" across multiple pixels at a time. With each stride, the output of each filter is collected in the layer's output. Thanks to the strides, this process results in a matrix that has a smaller height and width, and whose elements contain one value for each filter. This is called a feature map.

Stacked convolutional layers gradually reduce the height and width of the feature map, and increase its dimensions. The image is converted from a large grid of pixels into a small grid of features, with the values of the features corresponding to the content of the image in that region. In a trained network, the features in earlier layers will correspond to simple shapes and patterns, while those in later layers refer to higher level concepts and objects.

In essence, a convolutional model converts a raw image into a conceptual map that describes the image's content. This turns out to be a very useful representation: it can be used for many downstream computer vision tasks. For example, a classifier can be trained on the feature map to distinguish different classes of image. Alternatively, an object detection model might be trained to locate specific objects within the image.

The MobileNet family of architectures are designed to be small and efficient. Their size and computational cost can be controlled via two main variables: the input image resolution, which in turn impacts the size of the subsequent convolutional feature maps and the amount of memory they consume, and the width multiplier, α, which controls the number of filters and their associated weights.

This means a MobileNet model can easily be tailored for different targets. Its weights can range from a few dozen kilobytes to tens or hundreds of megabytes. MobileNet models of a few megabytes in size can be highly effective, and competitive with models that are much larger.

The secret to MobileNet's efficiency is its use of depth-wise separable convolutions. These are a variation on the traditional convolutional layer that are much more computationally efficient, with little reduction in accuracy.

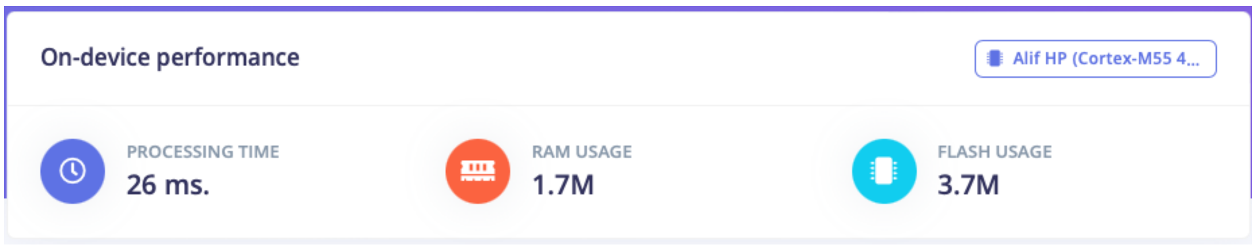
The MobileNet family has evolved over time. The original MobileNet made use of depth-wise separable convolutions as mentioned, and its successor, MobileNetV2, introduced a structure known as an inverted residual layer. This inverted residual layer allows MobileNetV2 to efficiently pass information between distant layers during training, resulting in better performance. The MobileNetV3 architecture was created using neural architecture search, a process that automates the discovery of efficient deep learning architectures. MobileNetV4 included novel structures designed to further optimize performance.

MobileNet on the Alif Ensemble MCU

The Alif Ensemble family is designed to deliver high performance in AI/ML applications, by combining the latest-generation Arm Cortex-M55 MCU cores along with Arm Ethos-U55 neural processing units (NPUs) to deliver fast, efficient inference on-device.

The Ethos-U55 NPU is an accelerator described as a microNPU, designed to bring deep learning acceleration to the energy-efficient edge. The U55 is ideally suited for running architectures like MobileNet with impressively low latency, and minimal power consumption.

The profiling tools in Edge Impulse help show the performance of the Ensemble MCUs. For example, the following project shows a fairly large MobileNet (input size 224, α width multiplier 1.0) running in 26ms per inference. This particular Mobilenet comes from ARM's model zoo, and was uploaded to Edge Impulse using Edge Impulse's BYOM feature.


On-device performance

Alif HP (Cortex-M55 4...

PROCESSING TIME
26 ms.

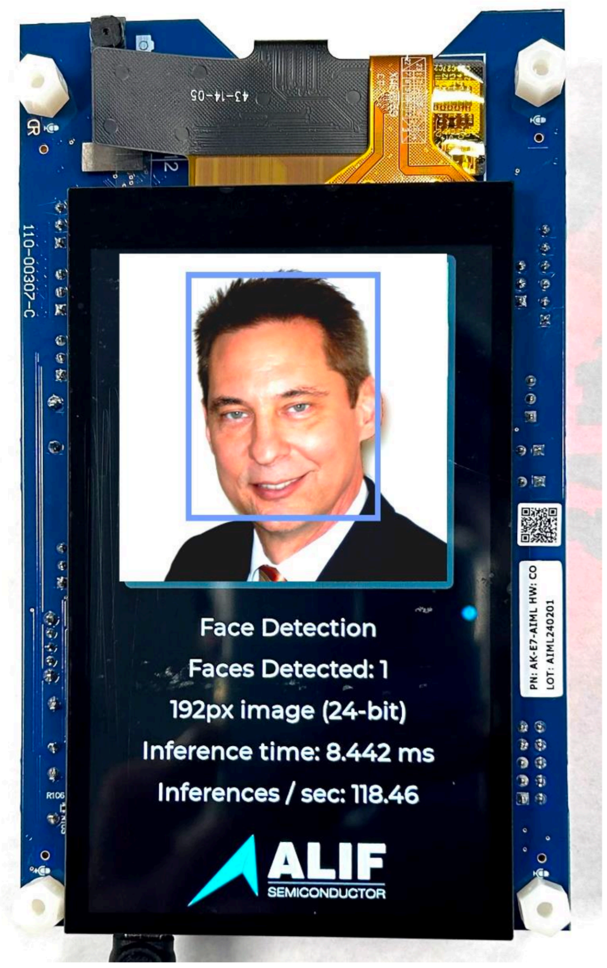
RAM USAGE
1.7M

FLASH USAGE
3.7M

The project can be found here, for closer inspection:
https://studio.edgeimpulse.com/public/316244/latest

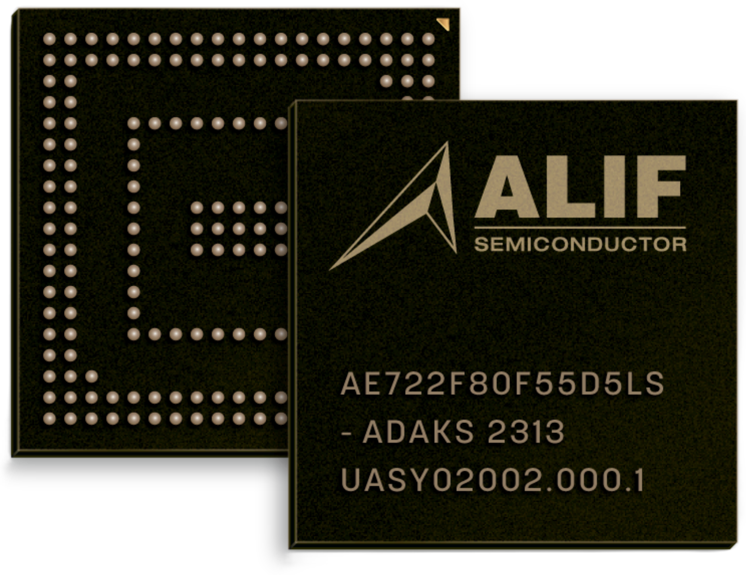
At 26ms, this is fast enough for real-time inference on multiple streams of video from the same device—incredibly impressive for such an efficient and low cost platform.

The Ethos-U55 is designed to work with int8 quantized models, which are models that have been compressed so that their (typically) float32 numbers are reduced to an 8-bit representation. The Ethos-U55 microNPU performs inference directly in the int8 space, leading to high performance and a 4x smaller model size.



When to use Alif Ensemble MCUs

The Ensemble family scales from a single core to a new class of multi-core devices, that combine up to two Cortex-M55 MCU cores, up to two Cortex-A32 CPU cores capable of running high-level operating systems, and up to two Ethos-U55 microNPUs for artificial intelligence and machine learning acceleration. Ensemble processors are designed to deliver an uplift in performance of at least two orders of magnitude compared to executing similar workloads on traditional 32-bit MCUs across a wide range of AI models.



This means that your machine learning tasks will run significantly faster on Alif devices, and even more important, you will save power at the same time. Executing an inference operation on a traditional microcontroller will force it to run at 100% CPU utilization throughout the inference process.

Since Alif's architecture cuts down the inference time to only a fraction of what it would have been if run on the CPU core, you end up extending your battery life at the same rate. There really isn't anything quite like it in the market today.

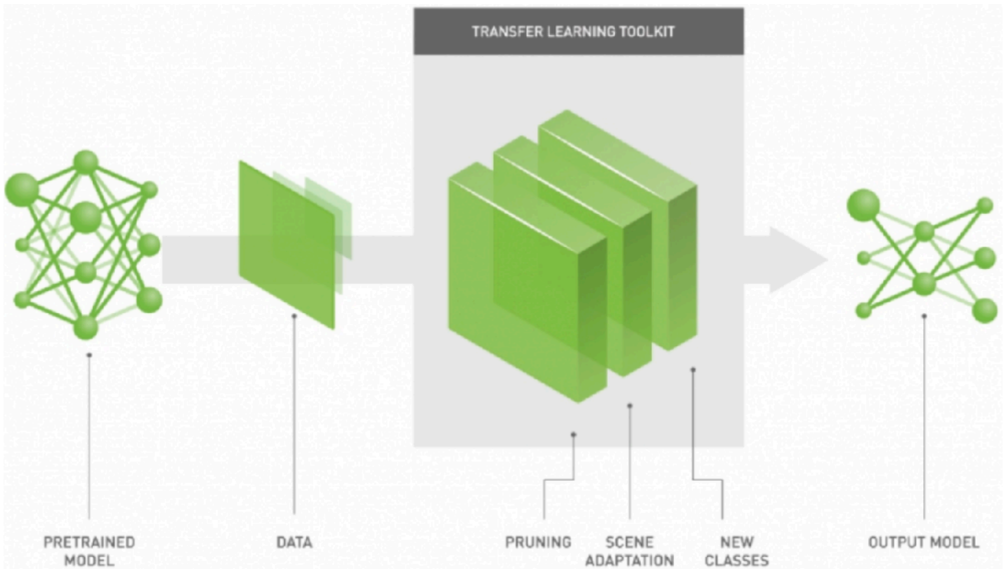
Also, because the Ensemble MCUs contain high-performance M55 cores, they are not restricted to basic microcontroller use-cases such as vibration detection or keyword spotting, but are capable of more complex tasks like language processing and vision-based applications as well. They come in a standard MCU footprint, and do not need a heat-sink.

A few common applications for the Ensemble devices are:

- Industrial monitoring
- Health care monitoring
- Smart wearables
- Smart machine vision cameras
- Infrastructure and building automation
- Audio processing and hearables

Object Detection with NVIDIA TAO on the Alif Ensemble MCU

The NVIDIA TAO (Train, Adapt, Optimize) framework is designed to simplify the task of creating custom computer vision applications, by leveraging NVIDIA's subject matter expertise and catalog of ready-made AI models. TAO allows users to customize these models by adding their own data, fine-tuning, then deploying the final output as an ONNX model. Traditionally these were meant for deployment to devices containing an NVIDIA GPU, but with the performance uplift of the Ethos U55 NPU, Alif Ensemble MCUs are now able to perform inference as well.

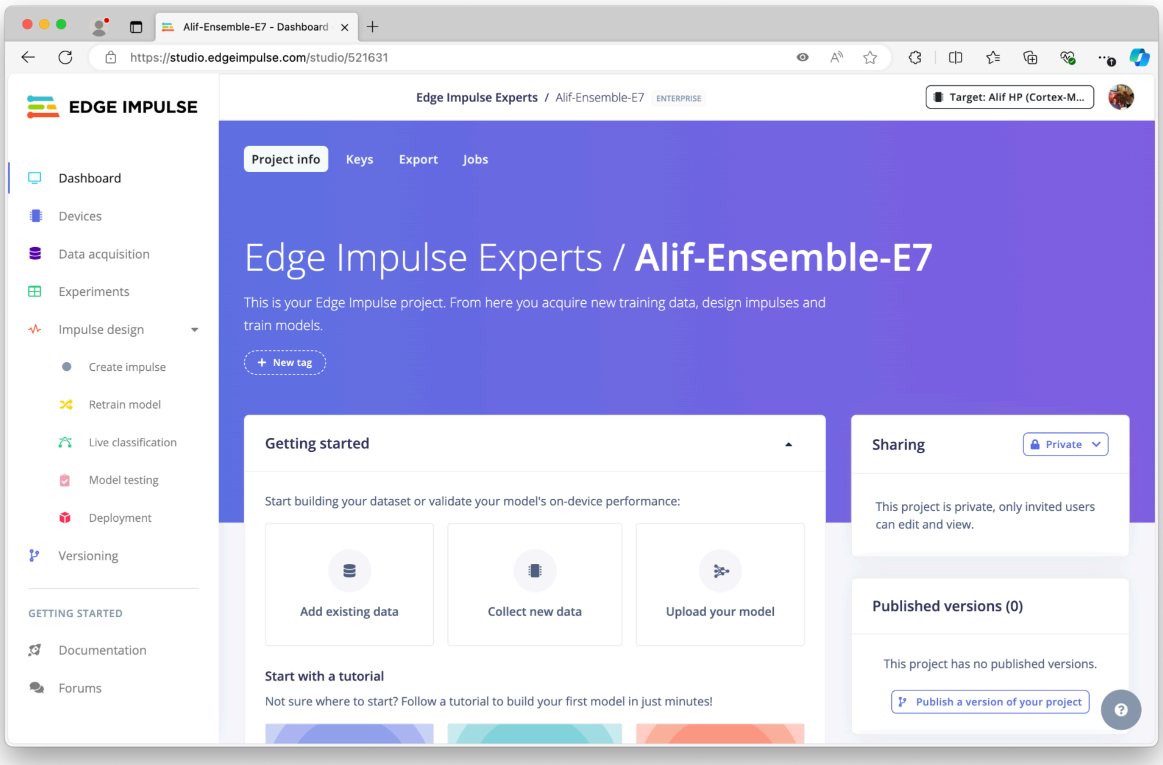


Source: https://docs.nvidia.com/tao/index.html

There are two easy ways to deploy NVIDIA TAO models to Alif Ensemble MCUs using Edge Impulse. The first method leverages the Edge Impulse Studio, which is a workflow-driven GUI that walks a user through the process of dataset curation, model training, and model deployment to a device. The second method can be used by experienced machine learning engineers working in their own development environments, via the Edge Impulse Python SDK.

Option 1 – Edge Impulse Studio

Building a machine learning model inside the Edge Impulse Studio is a great way to get started with computer vision and object detection applications. To begin, simply visit https://studio.edgeimpulse.com/ and login. You can create a new project, and provide a name, after which you will be brought to the Dashboard.



This is your Edge Impulse project. From here you acquire new training data, design impulses and train models.

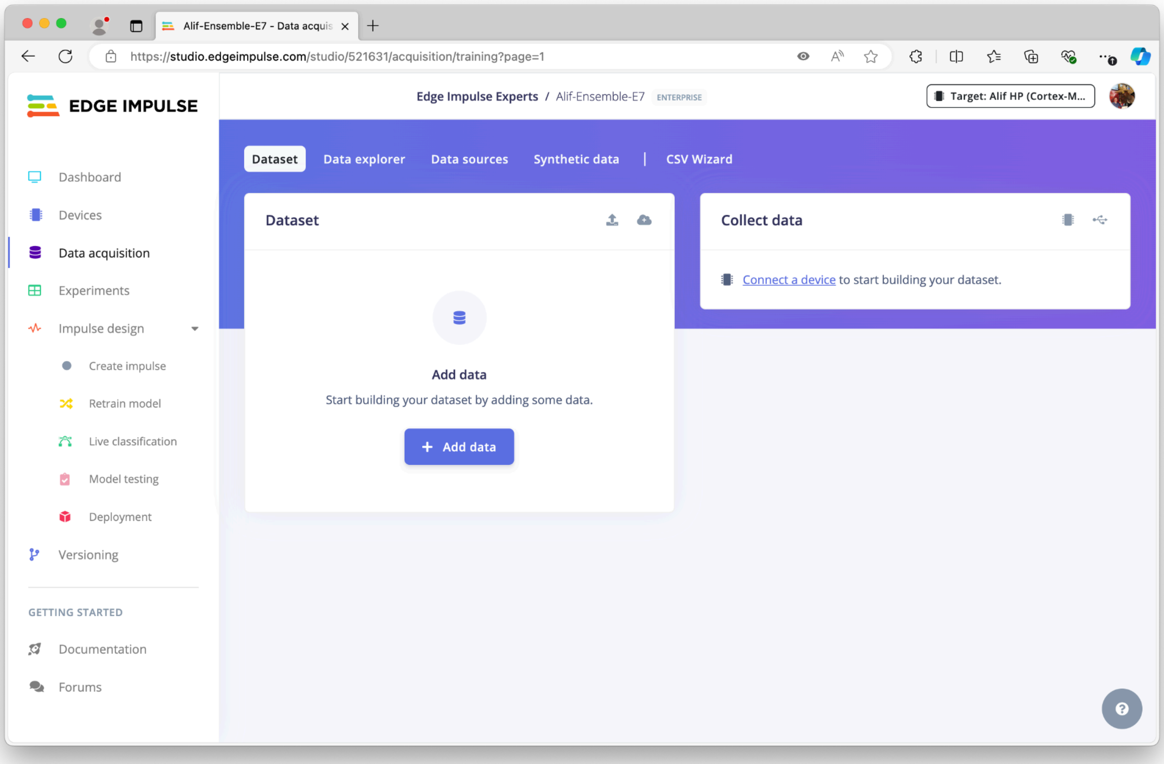
Start building your dataset or validate your model's on-device performance:

Not sure where to start? Follow a tutorial to build your first model in just minutes!

This project is private, only invited users can edit and view.

This project has no published versions.

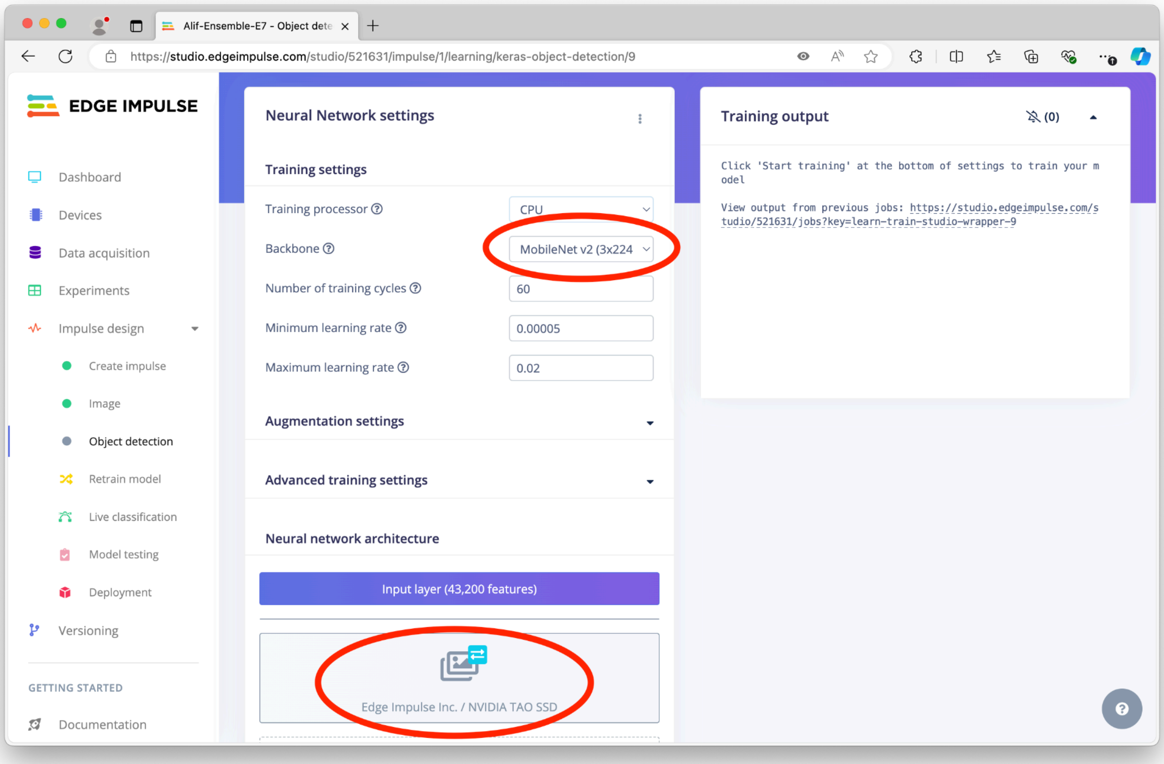
Object detection projects will need images collected, specifically images that contain the item of interest to identify, with bounding boxes drawn around the target. If you already have images collected, you can upload them by clicking on ***Data acquisition***, and then ***Add data***.


Alif-Ensemble-E7 - Data acquis
https://studio.edgeimpulse.com/studio/521631/acquisition/training?page=1
EDGE IMPULSE
Edge Impulse Experts / Alif-Ensemble-E7 ENTERPRISE
Target: Alif HP (Cortex-M...
Dashboard
Devices
Data acquisition
Experiments
Impulse design
Create impulse
Retrain model
Live classification
Model testing
Deployment
Versioning
GETTING STARTED
Documentation
Forums
Dataset
Data explorer
Data sources
Synthetic data
CSV Wizard
Dataset
Add data
Start building your dataset by adding some data.
+ Add data
Collect data
Connect a device to start building your dataset.

If you do not have existing images, you can also collect images with a smartphone or from a connected Ensemble AI/ML AppKit. Once the images have been captured, you can draw bounding boxes around the object of interest by clicking on the ***Labeling queue***.

With your dataset prepared, you can move on to ***Impulse design***, which is the workflow that helps to define the machine learning pipeline and then configure the neural network. Here, you'll choose an Image Processing block, an Object Detection Learning block, and then click ***Save Impulse*** in order to move on to Feature generation and have a look at the Feature explorer to see how your data is clustered.

Once complete, you can click on ***Object detection*** on the left navigation to arrive at the neural network settings. You can set parameters such as number of training epochs, learning rate, and add augmentation, as well as select from a variety of NVIDIA TAO architectures and MobileNet backbones. Once ready, click on the ***Start training*** button.



Neural Network settings

Training settings

Training processor	CPU
Backbone	MobileNet v2 (3x224
Number of training cycles	60
Minimum learning rate	0.00005
Maximum learning rate	0.02

Neural network architecture

Input layer (43,200 features)

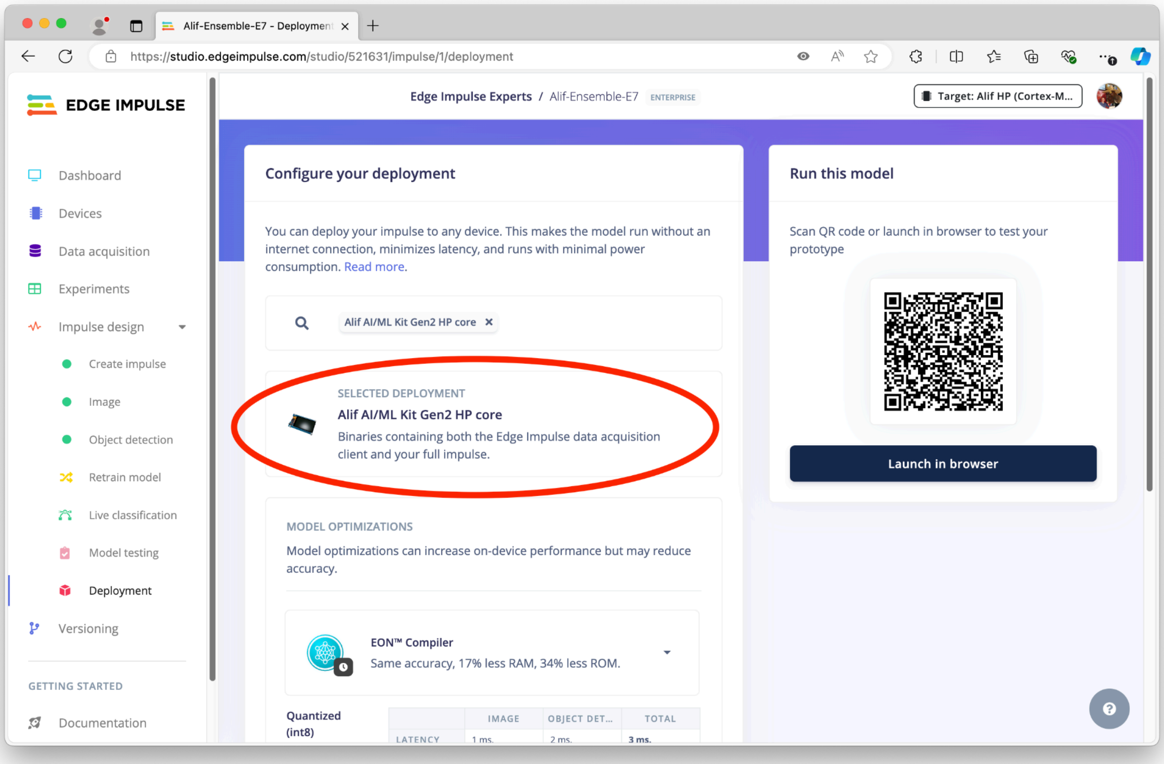
Edge Impulse Inc. / NVIDIA TAO SSD

```
Click 'Start training' at the bottom of settings to train your m
odel

View output from previous jobs: https://studio.edgeimpulse.com/s
tudio/521631/jobs?key=learn-train-studio-wrapper-9
```

Upon completion, you can test the performance of your resulting model by either using ***Live classification*** or ***Model testing***, which will apply the built model to live input from either an attached camera or Ensemble AI/ML AppKit, or set-aside Test images from the dataset respectively.

Finally, you can click on ***Deployment*** on the left navigation menu, and then choose from a variety of firmware, libraries, and target devices. To quickly ensure model functionality before integrating with your own application and code base, you can select either the ***Alif AI/ML Kit Gen2 HE core*** or the ***Alif AI/ML Kit Gen2 HP core*** depending on which unit you have, by beginning to type into the search box.



Configure your deployment

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.

Alif AI/ML Kit Gen2 HP core

SELECTED DEPLOYMENT
Alif AI/ML Kit Gen2 HP core
Binaries containing both the Edge Impulse data acquisition client and your full impulse.

MODEL OPTIMIZATIONS
Model optimizations can increase on-device performance but may reduce accuracy.

EON™ Compiler
Same accuracy, 17% less RAM, 34% less ROM.

Quantized (int8)		IMAGE	OBJECT DET...	TOTAL
	LATENCY	1 ms.	2 ms.	3 ms.

Run this model

Scan QR code or launch in browser to test your prototype

This will provide you with a downloaded .zip file, that contains a flashing utility and firmware that is ready to be deployed onto the Ensemble AI/ML AppKit. Once flashed, the board will automatically run inference with the model that you’ve created, making for quick and easy testing and validation that your model works. From there, you can alternatively download the model as a library, then begin the process of integrating into your application framework.

Option 2 – Edge Impulse Python SDK

Expert ML developers may prefer to train models in their own Python environment, for example in a Jupyter notebook. In this case, they can easily profile models on the Alif Ensemble MCUs – and create an Ethos U55 Library for deployment – using the Edge Impulse Python SDK using two simple commands:

```
Unset

profile = ei.model.profile(model=model, device='alif-hp')

print(profile.summary())
```

```
Unset

ei.model.deploy(model=model,
               deploy_target="alif-hp",

               model_input_type=ei.model.input_type.OtherInput(),

               model_output_type=ei.model.output_type.Classification(),

               output_directory=".")
```

As this option is intended for experience ML engineers, it is best to refer to the most recent official Edge Impulse documentation, which can be found here:

https://docs.edgeimpulse.com/docs/tools/edge-impulse-python-sdk#profile

What more can you do with the Alif Ensemble family and Edge Impulse?

Other models can be developed from scratch with Edge Impulse for the Alif Ensemble devices. For example, the ARM Model Zoo has a number of Sound Event Detection and Keyword Spotting models, but what if you wanted to change the data to alter their inference? You can create equivalent models using Edge Impulse by "cloning" these projects, changing the data collected, retraining the model, and then deploying to the Alif device.

Task	**Model**	**Edge Impulse**	**ARM Model Zoo**
Image Classification	MobilenetV1	Image Classification for Visual Wake Words using Transfer Learning on MobileNetV1	Visual Wake Words ARM Model Zoo
Keyword Spotting	MobilenetV1	Keyword Spotting using Transfer Learning on MobilenetV1	Keyword Spotting ARM Model Zoo
Keyword Spotting	DS-CNN	Keyword Spotting using DS-CNN network	Keyword Spotting ARM Model Zoo
Sound Event Classification	CNN	Sound Event Detection using 1D Convolution (CNN) networks	N/A - ARM Model Zoo does not contain Sound Event Detection models
Object Detection	Mobilenet-based and YOLO-based models	Object Detection using MobilenetV2 SSD FPN Lite	Object Detection ARM Model Zoo
Object Detection	Mobilenet-based and YOLO-based models	Object Detection using FOMO and YOLOV5	Object Detection ARM Model Zoo
Motion Classification	Full Connected (Dense) network	Motion Recognition with Dense Layer Classifier with Anomaly Detection	N/A - ARM Model Zoo does not contain Motion Classification Tasks

Wrapping Up

In this Solutions Guide, we've described the MobileNet computer vision architecture, and why it is an important part of Object Detection machine learning projects. We then detailed the NVIDIA TAO framework, and how it plays a role in building reliable vision applications thanks to the subject matter expertise provided by NVIDIA researchers and specialists. Finally, we go through a quick tutorial of how to build a TAO model using Edge Impulse, then deploy it to the Alif Ensemble AI/ML AppKit for high-performance, low-power edge AI inference – accelerated by the Ethos U55. For more information and to get started with the Alif Ensemble on Edge Impulse, you can head to the Alif Ensemble portal in the Edge Impulse documentation:

https://docs.edgeimpulse.com/docs/edge-ai-hardware/mcu-+-ai-accelerators/alif-ensemble

EDGE IMPULSE

ALIF SEMICONDUCTOR