World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(REVIEW ARTICLE)

Check for updates

Convolutional neural networks for real-time object detection with raspberry Pi

Mohit Jain * and Adit Shah

University of Illinois, Urbana Champaign, USA.

World Journal of Advanced Engineering Technology and Sciences, 2021, 04(01), 087–105

Publication history: Received on 28 September 2021; revised on 21 December 2021; accepted on 23 December 2021

Article DOI: https://doi.org/10.30574/wjaets.2021.4.1.0067

Abstract

With CSSNs integrated into Raspberry Pi, finding objects quickly in real-time at the edge is now possible. This article covers everything you need to know about deploying CNNs on Raspberry Pi, starting with model maintenance, through training, and ending with real-life use and improvements. MobileNet and Tiny-YOLO are the lightweight architectures we study, and they successfully provide accurate results despite facing harsh hardware rules typical in edge computing. IA is set up using the instructions in this guide, with TensorFlow and OpenCV used and optimization done by quantization and pruning. We also study how the Google Coral USB and Intel Neural Compute Stick 2 accelerators can boost performance without too much stress on the Raspberry Pi. In addition, the article shows how CNN-powered Raspberry Pi can be used effectively in smart surveillance, driverless cars, and home automation. We also look at what is new in edge AI, hardware-improved Raspberry Pi devices, and the increasing popularity of TinyML, outlining future options for using AI on a budget. By bringing powerful deep learning together with easy-to-use hardware, this article shows developers, educators, and enthusiasts how to locally create intelligent devices that respond in real-time. A result of this expertise is that users can take their AI ideas and make them work in real situations.

Keywords: Convolutional Neural Networks; Raspberry Pi; Real-Time Object Detection; Edge AI; Tinyml

1. Introduction

1.1. Edge AI is on the Rise

What started with big data processors and fast GPUs is now AI in your phone, living room, and workshop. This change, called Edge AI, is making big waves in the tech industry today. Rather than putting all its AI on remote servers, edge computing allows us to run it locally, even on simple gadgets such as smartphones and our star, the Raspberry Pi.

Exactly why is this important? Just imagine you're using an AI camera that recognizes when someone walks into your house. There will be short delays, lasting a few seconds, when requesting results. It doesn't work well when you have to be alerted quickly. Edge AI means all these processes happen soon on your device. The Raspberry Pi takes the footage, handles it using a neural network, and sends a notification without needing the cloud.

But it's not only measured by how fast it happens. By using Edge AI, applications in healthcare, security, and devices can be private and economical with bandwidth. If you can, you don't want private videos shared online.

Another benefit? Being able to handle more users and doing so affordably. Also, you won't need expensive servers or worry that the internet will fail your business. One Raspberry Pi joined with a good CNN model, is now skilled enough to do object detection that big PCs were once needed for.

* Corresponding author: Mohit Jain.

That's why Edge AI represents a real breakthrough, letting more people use AI and making it simpler, more portable, and resource-friendly. If you're starting, learning with friends, or building a business, you shouldn't hesitate to get involved now.

1.2. What Object Detection Does for Us in Today's World

If we're honest, the world and technology are evolving rapidly. It's possible to find object detection in smartphones that identify faces and in cars that operate independently. Much of what we consider everyday conveniences now depends on statistics.

A system using object detection can instantly spot and follow objects when presented with image or video data. Now that machines and systems are smarter, recognizing a stop sign or a stranger at your door is something we need our AI to do.

Why is all of this important? Since it changes the way machines interact with the things around us. Now, they participate in their environment by watching, thinking, and acting on what they see. Imagine how retail would be different if you could scan and pay for goods automatically, doctors could run automated tests on patients, farmers could monitor plants more easily, and guards could watch for anyone suspicious more effectively.

So, smart homes automatically use object detection to check on children outside and light indoor spaces when someone is inside. Thanks to telemetry, healthcare organizations can quickly spot unusual changes in a patient's condition. Automatic identification in software is also possible with Raspberry Pi cameras and CNN models so that animals can be studied safely in their environments without a researcher's help.

It becomes even more interesting because it's no longer just companies with huge supercomputers taking part. Thanks to the Raspberry Pi and good CNN designs, anyone can set up an object detection system that works in real-time.

2. Learning the Basics

2.1. What Do We Mean by Convolutional Neural Networks (CNNs)?

If you've ever been amazed by your phone unlocking with a photo or Google Photos automatically collecting your dog pics, CNNs was at work. These networks are popular for working with images—they perform well when looking for visual patterns.

What, then, is it that makes CNNs unique? CNNs act like our brains when we process sights. Just as your eyes tell your brain what is in front of you using shapes, colors, and movements, CNNs put images through convolutional layers to do the same thing. Like a filter, these layers examine an image, searching for edges, corners, textures, and finally, reaching entire objects.

At the heart of a CNN is its use of convolutional layers to extract outlines, details, colors and different textures from an image. Before long, these filters gradually get better at detecting various complex patterns. CNNs stand out because they can pick out useful features from images by themselves, without anyone having to do this manually.

And the great news is you have a great foundation to build from without starting from step one. Because of YOLO, MobileNet, and SSD, you can use CNNs trained with large datasets. You can adjust these models to fit your needs, even on a device like the Raspberry Pi that doesn't use much power.

What difference does this make? You can create effective, up-to-the-minute object detection systems using Hiro without having a background in deep learning or special technology. You can teach CNNs to look at images, make sense of them, and decide on a course of action.

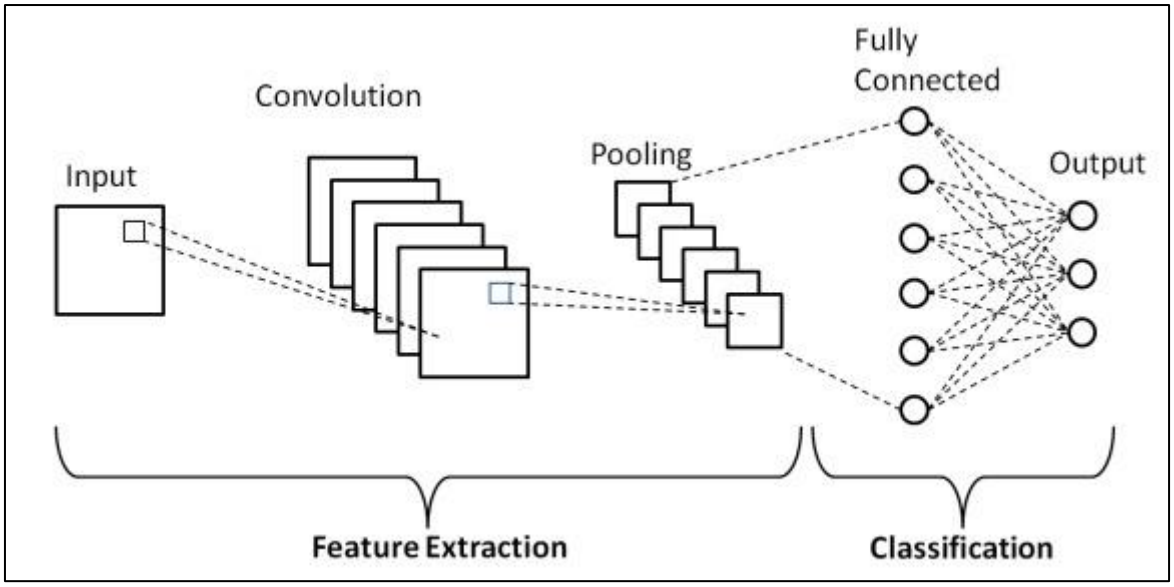

Figure 1 Convolutional Neural Networks (CNNs)

2.2. What Real-Time Object Detection Accomplishes

How quickly an operation runs is the most important thing in AI. Warning systems often need to tell us what is on the road correctly and rapidly. Only with real-time object detection can we handle today's technology's needs.

Take a moment and picture a car without a steering wheel or pedals. The outcome might be very serious if the system recognizes a person just seconds late in crossing. In another situation, a factory uses robotic arms to position different items going down a conveyor belt. A lag in the system disrupts the whole assembly line. That makes it so important for machines to process and act on images immediately.

The AI in your system checks each video input in real-time—as parts of the scene are happening on the monitor. Time is critical in these applications: surveillance, robotics, autonomous vehicles, and interactive gaming and augmented reality.

Naturally, the problem lies in making everything function on something like the Raspberry Pi. That's when model efficiency and hardware optimizations play a role. Designers made MobileNet and Tiny-YOLO operate more quickly on mobile devices that don't use much power. Real-time detection in security or AI isn't only possible with them; it's also practical.

These kinds of systems tend to keep users more involved and interested. When machines can respond to new environments, they become useful and easy to understand. Real-time detection allows robots to overcome challenges and changes cameras from passive viewers into crowds of active agents.

Object detection at the moment isn't simply a high-tech feature; it's vital for today's AIs that work in unpredictable places.

2.3. The Role of Embedded Systems in AI

Embedded systems are some of modern technology's most important unsung heroes, so let's explore them. They are stand-alone computers embedded in other products intended for carrying out certain tasks properly. Your smart things often have a microcontroller, like your smart thermostat, digital camera, or microwave. They are now being used to support intelligent applications as well.

The push toward decentralization in AI is bringing embedded systems into the main position. Rather than give data away to a distant server, on-site devices like the Raspberry Pi can now process it all by themselves. This change is exciting for real-time applications since it works well even when cloud connectivity is low, expensive, or ugly.

Today, the descriptions for Raspberry Pi, Arduino, and Jetson Nano are compact, easy to use, and powerful. They work well with machine learning and communicate with sensors and command actuators, so they perfectly fit AI at the edge.

Still, what makes CNN-based object detection depend so much on embedded systems? At the core, on-device inference lets your AI run by itself, so it doesn't need to rely on access to the cloud constantly. With this, users get faster results and greater privacy protection.

In addition, embedded systems are very economical. It doesn't take much money to start creating your smart systems. A camera, a CNN model, and a $35 Raspberry Pi can enable us to build prototypes beyond what we could have built in the past.

3. Raspberry Pi being used as an Edge Device

3.1. How Raspberry Pi 4 and 5 can be used in AI Tasks

Running AI applications is flexible and affordable on the Raspberry Pi series, and the Raspberry Pi 4 and its newer brother, the Raspberry Pi 5, are stellar choices for edge computing. Despite taking up little space and using little energy, they are surprisingly strong for real-time object detection using CNNs.

We'll look at this step by step. A quad-core ARM Cortex-A72 CPU clocked at 1.5GHz, up to 8GB of RAM, dual monitors, and USB 3.0 ports are all available features on the Raspberry Pi 4. Using this design, the device is suitable for regular tasks and running deep learning models after a little optimization.

The Raspberry Pi 5 is the next device, with a real upgrade to a 64-bit processor, 8GB of memory, and eMMC flash storage that you can access through add-ons. The Lite model is up to two to three times faster than the earlier Pi 4, allowing AI tasks to be completed more easily than with the Pi 4.

After these upgrades, object detection, face recognition, and video analytics have become possible for everyone. Using YOLOv5s, MobileNet and Tiny-YOLO doesn’t cause any serious problems because frameworks like TensorFlow Lite and PyTorch Mobile use little phone memory.

With GPIO pins, a camera and tools such as the Coral USB Accelerator and Intel Neural Compute Stick 2, the board’s AI performance boosts. This means that both enthusiasts and serious edge AI developers use Raspberry Pi.

By and large, Raspberry Pi 4 and 5 show that a computer's size doesn't always count—as long as you're looking for detailed, energy-efficient, and local AI.

3.2. Why Raspberry Pi is perfect for Both Prototyping and Deployment

When getting started in AI for either edge computing or detecting real-time objects, getting a Raspberry Pi is a great move. However, what is it that makes it suitable for both prototyping and putting an application into production?

Accessibility should be our first discussion point. Thanks to its low price of only $35, it’s easy to purchase a Raspberry Pi. Because of this, students, teachers, makers and early teams wanting to develop their first software save money on hardware by using cloud computing. Next, less experienced users can manage the platform easily, but it's also powerful enough for more advanced users. Since there is a large community and plenty of documentation, it's simple to get going with Docker. Whether you are using the Android Studio, flashing an OS, or installing OpenCV and TensorFlow, you'll find the directions easy to follow—and someone is usually available on a forum to solve any problems you encounter.

Using Raspberry Pi for prototyping means you have a lot of options. All it takes is connecting a camera module, connecting a Coral or NCS2, and running your CNN models within hours. You can see how object detection works in images shot with various lighting, distances, and angles. Rapidly going through tests and updates is helpful when working out the best model and hardware.

Most importantly, you can take your prototype and use it directly on your device. Raspberry Pi is strong enough for use in technology solutions that stay in place for a long time. Using the Tensor on Edge module can detect objects reliably in a drone, on a robot, or in your home security system.

That isn't the only thing sports teach us. Because Raspberry Pi is economical in space, uses minimum energy, and has a variety of inputs and outputs, it's suitable for embedded and portable systems. You need a power bank and a Wi-Fi connection; you don't need a data center.

Raspberry Pi simplifies bringing your AI ideas into reality, which is why it's so useful for AI at the edge.

3.3. Exploring the Comparison of Raspberry Pi to Other Edge Devices

Edge AI has led the market to be filled with devices claiming strong performance but low energy consumption. Even though Raspberry Pi leads the edge computing space, reviewing its benefits is helpful compared to NVIDIA Jetson Nano, Google Coral Dev Board, and Intel NUC.

To get us started, we'll use the Jetson Nano board. Thanks to its powerful NVIDIA Maxwell GPU and 128 CUDA cores, a deep learning approach is highly supported in this device. People using NVIDIA's CUDA and TensorRT can boost GPU performance on the H100. Yet, its power consumption and price are higher than a Raspberry Pi's, and because fewer people use it, the number of pre-made resources is also smaller.

Next, we'll look at the Google Coral Dev Board. This gadget is set up with the Edge TPU, aiming exclusively at TensorFlow Lite. Regarding inference speed, it's much faster than Raspberry Pi. Even so, it is more stubborn; it offers fewer framework choices and has more hardware support rules. Google's system works best if you mostly use their services.

Next, Intel NUC is a compact PC that can get the job done with desktop power. You can easily train big neural networks even if your machine is slow. It's also unnecessary (and is not affordable) for much of what happens on the edges. Moreover, you don't get the same amount of GPIO and embedded hardware support as you do with the Raspberry Pi right out of the box.

What is the Raspberry Pi best at? It's ideal because it is economical, flexible, and has community support. Its relatively low horsepower may mean it isn't first, but its sensors, large group of users, and flexibility make it the most popular platform for object detection with CNNs.

Anyone new to creating things or wanting to expand their project rapidly should look to Raspberry Pi. If you find your application needs more power, attaching it to an external accelerator is likely what you should do. The Raspberry Pi is the easiest and most useful edge device you can buy for AI today.

Table 1 Hardware Comparison for Edge AI Devices

Device	CPU	RAM	AI Accelerator	Cost (USD)	Edge AI Suitability
Raspberry Pi 5	Quad-core Cortex-A76	8 GB	External (TPU)	$75	High
NVIDIA Jetson Nano	Quad-core ARM Cortex-A57	4 GB	Built-in GPU	$99	Very High
Google Coral Dev	Quad-core Cortex-A53	1 GB	Edge TPU	$129	Very High
Intel NUC	x86/i7 options	8–32 GB	Optional	$300+	Extremely High

4. Integrating CNNs with Raspberry Pi

4.1. Selecting the Right CNN Architecture

Choosing which CNN architecture to use is crucial when starting with Raspberry Pi object detection. Which method you choose will change your progress and how smoothly your project runs. Because there are strict resource limits on Raspberry Pi 4 and 5, getting a gun that mixes speed, accuracy, and how big it is is important.

Although ResNet, VGGNet, and Inception do well on many tasks, they are unsuitable when memory and processing capacity are restricted. Forces would make running such models on a Raspberry Pi difficult and impractical.

Rather, choosing a light set of architectures is your best option. MobileNet, in particular, is made to operate on mobile and embedded systems. Because it uses depthwise separable convolutions, it reduces the number of parameters, is fast, and uses less memory than traditional models. A lot of people also turn to Tiny-YOLO. The YOLOv3 model provides more

efficient object detection in real time than the original YOLO model. EfficientDet Lite and SqueezeNet should be considered by anyone looking for a model that works quickly and is small.

Finding the proper architecture is about keeping latency low, providing accurate results, and maintaining a moderate model size. You want the model to work in real time, so latency is important. Safety and security become even more important when accuracy is needed in the model's applications. You must keep the model size small so it won't use too much memory, often limited to the Raspberry Pi.

Depending on their application, most people will benefit from MobileNet or Tiny-YOLO. They give the best of both worlds regarding speed and performance and are available on many frameworks, including TensorFlow Lite and OpenCV. Above all, your decision should fit the goals of your software, work well with the Raspberry Pi's surroundings, and play nicely within the Pi's hardware limitations.

Table 2 Comparison of Lightweight CNN Architectures

CNN Model	Model Size	Accuracy (mAP)	Inference Speed (on Pi 4)	Suitable Use Case
MobileNet SSD	~16 MB	Medium	Fast (50–100 ms/frame)	General object detection
Tiny-YOLOv3	~33 MB	Medium-High	Medium (100–150 ms/frame)	Real-time tracking
SqueezeNet	~5 MB	Low-Medium	Very Fast (30–70 ms/frame)	Basic detection tasks
EfficientDet-Lite	~20 MB	High	Slow-Moderate	High-accuracy tasks

4.2. Pre-trained Models vs Custom Training

When beginning an object detection setup on Raspberry Pi with a CNN, you must figure out whether using a readymade model or creating your own is better. Pre-trained models are always a good and practical choice unless you're limited by hardware or time.

Modern models are created after training on a huge amount of data in ImageNet, COCO, or PASCAL VOC. You don't need to train them—they recognize many things from the packaging. These models make things easier because you get to benefit from the information that's been learned earlier. Rather than creating a model from scratch, you can improve a model you get from the ground using transfer learning.

You use transfer learning by starting with a trained model and training only the model's last layers on a smaller data set for your task. It reduces the burden on computers and requires much less memory than other methods of microsimulation of aging that have been used. With Raspberry Pi, you can make powerful models work for your needs right on your PI, thanks to edge computing.

In most cases, making your training data isn't required until your objects don't appear anywhere in prior datasets. A significant amount of labeled information and considerable computer resources are also needed. Because of its limited resources, most people don't train a model on a Raspberry Pi, and it is usually saved for individuals with powerful computers.

Most of the time, using a pre-trained model and applying transfer learning makes your system effective and reliable much quicker. Thanks to the strength and experience of well-known designs, this approach improves your model's function and ability to succeed.

4.3. Model Optimization for Edge Devices

Having a suitable architecture and a qualified model won't automatically make your Raspberry Pi run as fast as needed. Following this step, the model operates well on an edge computer with limited resources. In its absence, even simple models might work noisily or get slow and use a lot of battery.

Quantization is the main technique we should use. Doing this reduces the number of bits in your weights from 32 to 8. Generally, there is only a small impact on how accurate the system is, but using these methods improves how fast and efficient it is. With the Raspberry Pi, a difference between lagging and real-time performance can make all the difference.

It's also important to use pruning techniques. Removing minor weights during pruning makes operating the neural network easier and faster. As a result, you can allocate more resources, and your interpretations usually take less time. After a while, the model's accuracy may fall slightly, but you can restore or improve it by pruning and finetuning.

To best adapt to new software, companies must convert their models. Putting your trained model in TensorFlow Lite or ONNX allows your project to be optimized for devices at the edge. TensorFlow Lite was developed with the Raspberry Pi in mind, as it includes quantization and can work safely with accelerated hardware.

Anyone interested in accelerators should know that the Google Coral USB Accelerator or the Intel Neural Compute Stick 2 can greatly boost performance. By letting the Raspberry Pi offload inference, these devices help detection occur more quickly and effectively. One example is a model that needs 400 milliseconds to handle one frame on a CPU but just 20 milliseconds with an accelerator.

The model size also needs to be lowered. It's possible to simplify the model structure, reduce the resolution of each image, or make the model only predict fewer output classes. Each parcel of unused memory and processing power adds to a hassle-free system that runs longer.

In other words, optimization turns your model into a solution you can use in real life and deploy onto the Raspberry Pi and similar systems. It allows your system to function capably, even when resources are limited, and ensures it functions well.

5. Hardware and Software equipment is needed while working.

5.1. How to Prepare the Raspberry Pi for Artificial Intelligence

You should prepare everything for AI on your Raspberry Pi before beginning any real-time model training or object recognition. The setup is essential since it supports everything you'll do with machine learning. It's good that the Raspberry Pi community is so active because beginners will find plenty of helpful tools and explanations.

The first thing to do is pick the right Raspberry Pi OS. Many people use Raspberry Pi OS (codenamed Raspbian) because it is the most supported by the Company. To benefit from the Raspberry Pi 4 or 5 and fully use their RAM, use the 64-bit version on your AI projects.

Upgrade all system packages for compatibility and safety as soon as you have installed the OS. You can update the system with basic commands such as sudo apt update and sudo apt upgrade. The next thing to do is enable the camera interface in the Raspberry Pi Configuration utility if you plan to add the Pi Camera Module to work with actual-time detection of objects.

Because Python is most commonly used in AI, ensure you have installed Python 3, which comes with pip, the Python package manager. It's wise to get virtual environment tools to install each project with its libraries.

Once that's done, install NumPy and SciPy, as they are fundamental to the numerical side of machine learning. After that, you should install TensorFlow Lite, PyTorch (ARM versions), and OpenCV for your device. With these libraries, builders can set up and manage deep learning applications.

Access to GPIO, a camera module, and USB and HDMI options enables you to build your own AI systems with a Raspberry Pi. Irrespective of what you're building, your Raspberry Pi needs to be fully installed and regularly maintained as the base. Ensuring a good foundation guarantees the rest of your planning will go as smoothly as required.

5.2. Libraries and Frameworks

Proper library and framework installation should be your focus when your Raspberry Pi is ready. Now, because of these tools, you can build, train and deploy your models with added ease.

Many people use TensorFlow as a machine learning framework and devices running TensorFlow Lite also receive support. Because it is built for small devices, TensorFlow Lite lets you use features such as quantization and speed up your models with external help from Google Coral. On the Raspberry Pi, you can use MobileNet and EfficientDet object detectors thanks to the good performance of TensorFlow Lite.

Many people use PyTorch for its wide reputation as a flexible and capable framework. PyTorch isn't as small as TensorFlow Lite, but you can work with it on Raspberry Pi with an ARM version. PyTorch Mobile and TorchScript let you export models that can be used for quick inference, and they become especially useful when you want to alter how your model works.

Trying to do anything in computer vision without OpenCV is difficult. The library is built for image processing and lets you manipulate video, filter images, track objects, and do much more. With the help of TensorFlow or PyTorch, OpenCV performs both resizing and drawing bounding boxes around found objects.

All of these libraries were made to support and work alongside the others. OpenCV can grab video from a source and TensorFlow Lite uses it to draw conclusions that are sent to outputs in no time. Thanks to Keras, TensorFlow and Deep Learning, developers can build AI apps for the Raspberry Pi.

While it may take time to organize these frameworks initially, you will enjoy smooth development using high-performing on-device AI.

5.3. Deployment Tools and Accelerators

Deployment is the exciting final stage after you train your model, where you need the AI to work with your Raspberry Pi. That's why tools and hardware accelerators are required to let real-time object detection happen even on a lightweight smartphone.

Edge devices using models rely on TensorFlow Lite or TFLite as the first choice. It is created to help machine learning models perform and feel much faster using just a CPU, not a powerful GPU. TFLite models are compact, fast, and easy to add to your code. They enable models to be turned into quantized data forms, cutting their size and inference steps and integrating them with many accelerators.

The Google Coral USB Accelerator is considered a big standout out of all the accelerators. An Edge TPU is built onto the SoC to handle AI functions. The Raspberry Pi makes inference much faster, even though models can run in real-time. Since TensorFlow Lite is at the heart of the Coral API, adopting it is easy for existing approaches.

Intel Neural Compute Stick 2 (NCS2) is a great tool to use, too. Myriad X VPU from Intel and the OpenVINO toolkit are involved in this solution. NCS2 can speed up running models developed in both TensorFlow and Caffe. Once tied to the Raspberry Pi, execution speeds are much quicker than achieved on a computer's CPU alone.

OpenVINO is another powerful kit for improving and running deep learning models. Whenever your app needs extra custom control, Raspberry Pi is a better fit, thanks to its wide support for hardware accelerators.

Your deployment decision and selection of an accelerator rely on your system's architecture, the time needed for application response, and your available resources. Whether TFLite is enough for you or you add Coral or NCS2, proper devices can help make the Raspberry Pi work much like servers and GPUs—except instantly—saving you time and money.

6. Developing a Real-Time Object Detection Systems

6.1. Collecting and Preprocessing Data

No AI model would function without data. Like all computers, advanced neural networks depend on good data to work well. Collecting and properly processing your data is one of the key things to do when making a real-time object detection system for Raspberry Pi. It tells you if your forecasts will be accurate, sound, and able to work well when applied in real life.

First, you have to identify the objects your model will be able to detect. It doesn't matter if the animal notices cars, pets, tools, fruits, or moves of certain people. If your data covers just the objects important for your application, your model will do much better. A Raspberry Pi camera can collect data or download COCO, PASCAL VOC, or Open Images datasets. Still, remember that some public data is broad enough to include many extra classes you won't need and could make your model run more slowly when inference takes place.

If you do all the data collection by hand, you must maintain consistent methods. Shoot your pictures in many kinds of lighting, positions, and backgrounds. Almost all-natural language processing tasks use this to help the model handle cases in real time. You could ensure that your data is fully collected by having a script capture pictures every few seconds automatically or once motion is detected.

Labeling your images is the step that follows creating them. LabelImg, VoTT, and MakeSense.ai make it easy to outline objects and save those annotations in Pascal VOC XML or YOLO formats. Labeling your images properly is a requirement because it decides whether your model can find and recognize things accurately.

Labeled data must then be preprocessed. It often means making all your images the same size, adjusting the intensity values, and adding fake images by flipping, rotating, or adjusting brightness. They make your data more diverse by modifying the images without collecting more pictures.

When your data is good, your model will be good, too. Spend time during this part of the process because it's one of the best ways to improve your results later.

6.2. Training and Evaluating the Model

When your data is organized and included, it's time to begin training the object detection model. At this point, your CNN is taught to find out what each image contains and tell apart various objects. It's not enough to limit training to throwing data into a neural network—it needs a well-defined plan, practice, and review steps.

Rather than training on the Raspberry Pi, handling training on a different device is better. Using today's hardware isn't enough to hold the Eqc calculations required by model training. A better solution is to use Google Colab, AWS, or Azure on the cloud. The optimized Pi model can be exported for inference during training.

Choose the suitable approach to architecture at the beginning. A good first step if you use TensorFlow is to look at MobileNet SSD or EfficientDet Lite models. PyTorch users usually choose YOLOv5s (the small version) thanks to its good compromise between performance and size. As soon as your data is selected, determine your batch size, learning rate, number of training iterations, and number of input features.

Each time the model is trained, it runs your data through a fresh pass or epoch to change its parameters to lower prediction errors. You should check the model's accuracy by checking mAP, IoU, and loss functions. They show you if the model improves or fits only the examples in the training set.

After you finish training, test the model on a distinct voting set. All the images in this dataset should be ones the model hasn't seen. This part matters most because it displays whether the model can work properly. Low accuracy calls for you to check your data, try a new architecture, or play with the values of hyperparameters.

After you're finished, export the trained model to your computer. Use either TensorFlow's tflite_convert or PyTorch's torch.jit.trace to get the model set up for deployment. Ensure the model, label entry point, and configuration files are saved. The Raspberry Pi will employ them to make inferences.

Learning how to train a model is a fun and productive task. You put all your data work into this stage, and it provides the base on which your object detection process relies.

6.3. Real-Time Inference on Raspberry Pi

You're ready to see how your CNN works in reality on the Raspberry Pi. Inference means the model uses training to interpret and predict from fresh input forms such as a live video. We want this system to achieve quick processing and excellent accuracy, allowing it to function well in real situations.

Your first task should be to move the trained model onto the Raspberry Pi. Set the format to TensorFlow Lite (.tflite) if it's a TFLite model or TorchScript (.pt) if it uses PyTorch. When on the Pi, make a Python program that reads the model and prepares it to be used for predictions. The script should also connect to your Pi Camera or USB webcam to take video frames as they happen.

OpenCV greatly assists the work here. It can record and process video frames and outline areas around objects. Conventionally, an image is captured, resized, and normalized, processed by the model, and the outcome is displayed in boxed and labeled form—all in just a second.

Several different factors influence whether real-time performance is achieved. The model comes first when defining a business problem. Light models like MobileNet SSD and Tiny-YOLO are the better options. After that comes optimization. Autonomous driving model versions without high precision are significantly faster. Hardware accelerators are the third tool on our list. You can dramatically lower inference time by using a Coral USB Accelerator or Intel NCS2. The result is often an inference of 50 milliseconds or less per frame.

While doing inference, watch how your system performs. Keep track of your CPU in two ways: top and valenced measure_temp. When there is poor cooling, lots of stress on a computer for too long can cause it to crash or reach maximum capacity.

If the proper tools are used, it's doable to carry out real-time object detection on a Raspberry Pi. The camera reads, alters, and screens the images in real-time. Today, thanks to edge AI, what used to need a large desktop can now be run from your phone.

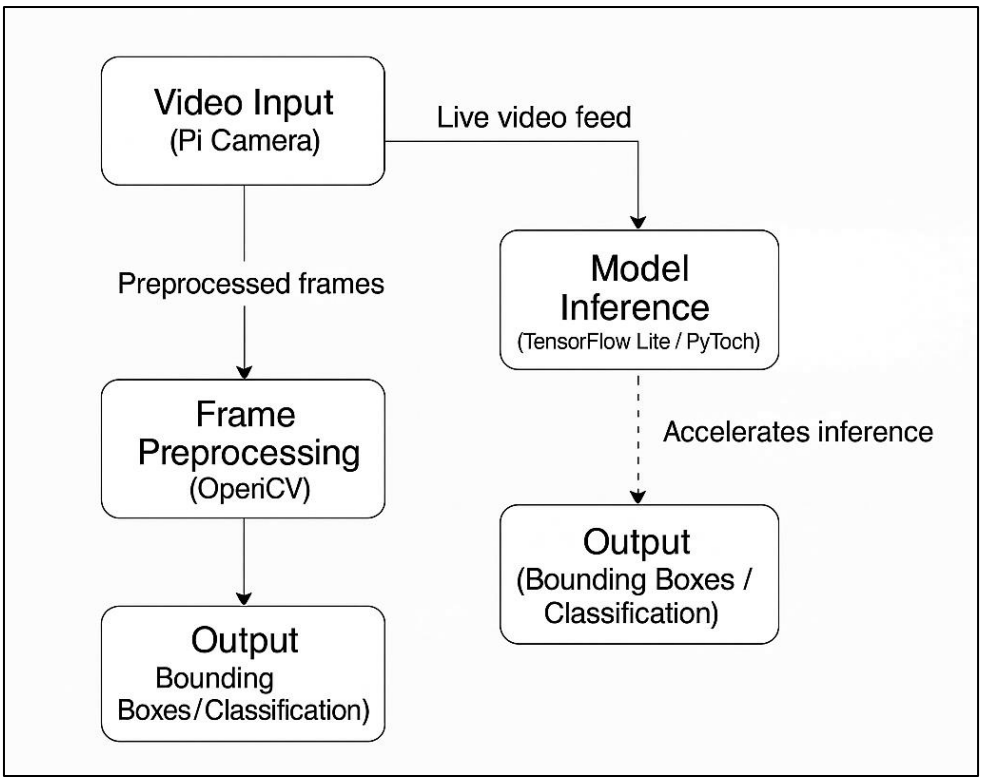

Figure 2 System Architecture for Real-Time Object Detection

7. Performance Optimization

7.1. Quantization and Pruning techniques

Making a neural network work well on edge devices like the Raspberry Pi is no longer an option—it's vital. A model can still be unusable without optimization, resulting in slow efficiency and high resource costs. Among the strongest ways to make CNN models edge-friendly are quantization and pruning.

To quantify your model means lowering the accuracy of the numbers describing its parameters. We usually save the weights and activations in the network using 32-bit numbers. As a result, audio is represented by either 16-bit or 8-bit integers. Making the model smaller helps it run faster on Raspberry Pis and devices with limited memory and quicker integer calculation. Using quantized models makes them twice to four times faster, with little sacrifice to the results.

Pruning, instead, tries to remove weights that play a minor role in making the prediction. It's like removing what you don't need. Following training or after completion, the model detects contacts that play minor roles in its operation and trims them for a simpler and faster design. As a result, models are more rapid to check and use less memory, helping devices with small RAM.

With TensorFlow Lite and PyTorch, quantization and pruning become tools. Post-training quantization is supported automatically in TensorFlow Lite, but PyTorch gives you pruning options through its torch.nn.utils.prune module. Combining these techniques produces a model that is both effective and takes less space but still can detect objects well on the Raspberry Pi.

Quantization and pruning are the best tools in your kit. You can use them to run cloud models on a Raspberry Pi, and they won't limit your ability to use real-time AI.

7.2. Making Edge TPUs Faster

Although the Raspberry Pi is impressive for what it does, using deep learning models can make it work at its highest capacity. That's when devices like Edge TPUs are useful. Designed for AI applications, they improve how quickly your model works and allow real-time object detection.

Many Raspberry Pi enthusiasts use the Google Coral USB Accelerator since it has an Edge TPU chip. This chip aims to run quantized models made with TensorFlow Lite rapidly. After plugging in your Google TPU, your CPU will no longer run the inference calculations, making computation for new data far faster.

It doesn't take long to get the Coral Accelerator set up. You must sign in, choose your device, ensure Edge TPU is selected, and compile your model with TFLite, which is normally converted to a quantized TFLite model with TensorFlow's tools. From that point, the Coral API does the inference, running on video frames faster than the Pi's CPU could all by itself.

Observations made with actual hardware have displayed outstanding results. Since an Edge TPU can often produce 3D object detection 25 times faster than a CPU, a sluggish system can be turned into a responsive real-time app. This difference in performance matters greatly for things like surveillance, robots, and self-driving cars.

In addition to being super fast, the Edge TPU saves power. Because it saves energy, it is highly recommended for anyone building portable or battery-powered gadgets with a Raspberry Pi. Using managed memory also gives the system space to handle GPIO, keep files, and handle different background operations.

Edge TPUs make your Raspberry Pi run faster than before. If you want to achieve low latency and good resource management, it's among the most worthwhile things you can spend your money on.

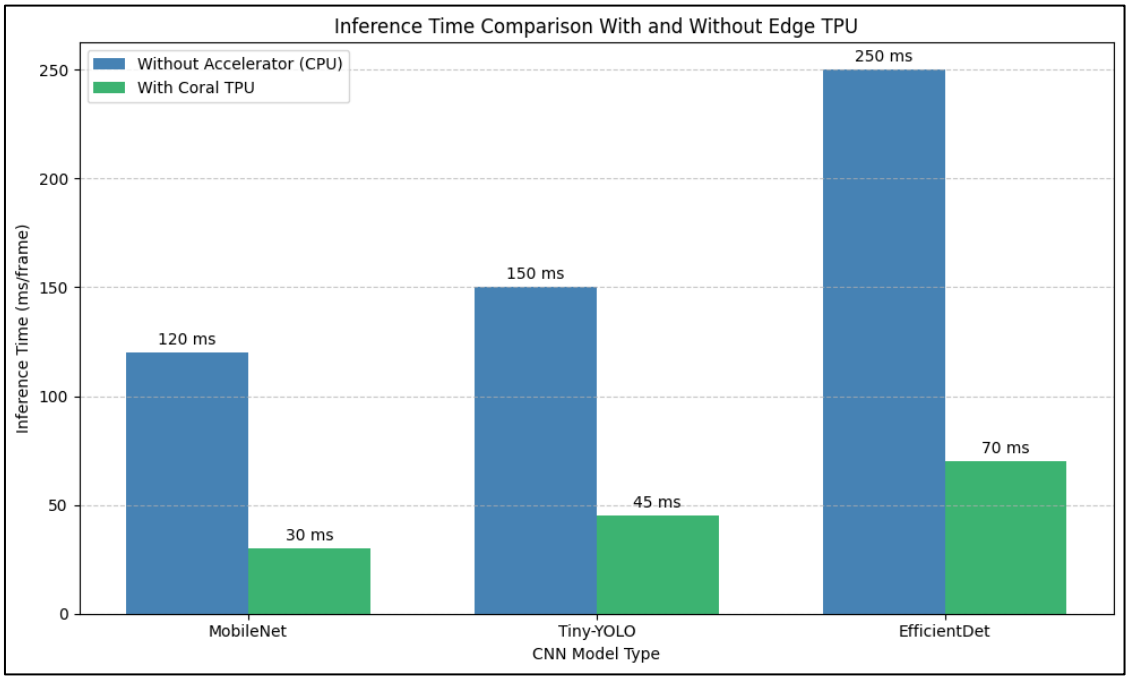

Figure 3 Inference Speed With and Without Accelerators

7.3. Reducing Latency and Power Consumption

Speed and energy usage are the biggest issues once an object detector is trained and put to use on a Raspberry Pi. The right model is insufficient; fast inference requires software, hardware, and environmental attention.

The term latency measures the time it takes your system to move from an input to an output. You must grab a frame, get inferences, and display objects faster than a second to detect objects. Most applications in robotics and surveillance require decisions to be made quickly, so every millisecond matters.

When trying to reduce latency, begin with the most important steps. Try out Tiny-YOLO or MobileNet SSD for your model, which is both lightweight and ensures it is quantized, so it works faster. After that, try to reduce the input image's resolution. While it's true that high-res photos are more accurate, they will always take extra time to be processed. Adjusting the image resolution to 300x300 or 224x224 can save plenty of inference time and only reduce accuracy by a little bit.

If you optimize the software your developers use, that will also pay off a lot. You should use TensorFlow Lite instead of the regular TensorFlow. Inference speed on devices at the edge is the main focus for TFLite. OpenCV enables you to handle both of these processes simultaneously. Your Raspberry Pi won't need to finish every task before starting a new one.

Now, we'll look at how much power these devices use. If you run deep learning models nonstop, it won't last long on your power supply. This is an even bigger problem for remote or mobile applications, which must save energy. The solution? Make sure you use the latest information when making your updates. Rather than dealing with every frame, handle only one each second or carry out your modeling only when movement is seen.

Hosting computation to special devices such as the Coral USB or Intel NCS2 is a useful additional technique. These gadgets decrease latency and run on processors that don't use as much power as normal CPUs.

To achieve this, the Company must focus on many little details. All the optimization measures can make your Raspberry Pi an excellent and efficient AI edge device.

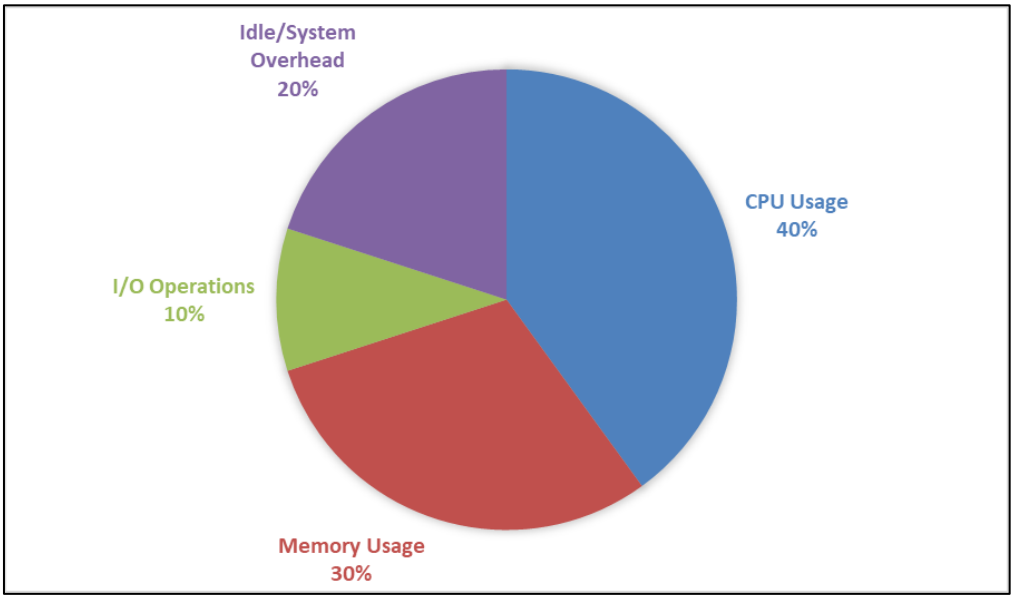

Chart 1 Resource Usage Breakdown during Inference

8. Challenges and Limitations

8.1. The Limits and Blockages of Hardware

What gives the Raspberry Pi its abilities, whether low cost or many possibilities, is ultimately the computer's hardware. These constraints are challenges when using CNNs to detect objects in real-time. A particularly important challenge is

the amount of processing power needed. Since ARM-based CPUs are in the Raspberry Pi 4 and 5, they are very efficient but do not deliver the same performance as top-class desktop chips. When we try to use complicated CNN models, we realize that their working speeds become very slow without optimization, which keeps us from processing things fast enough in real time.

Memory capacity forms another obstacle. If you try to use MobileNet or Tiny-YOLO on high-resolution images or several processes, your device's RAM can fill up very quickly. Raspberry Pi 4 can handle up to 8GB of RAM, but that's not as much as in most small computers. System crashes or throttling can appear if you do not use memory correctly.

How fast data is being read and written needs to be given some thought. Most people working with Raspberry Pi store information on microSD cards over SSDs, since microSD cards are faster for storage. Completing frequent writes during logging or data processing may harm the card's performance and eventually corrupt it. The last supporting technology is thermal management. If you don't use cooling systems, having the Raspberry Pi work heavily will cause it to overheat and trigger thermal throttling. Throttling causes the CPU to slow down, which changes real-time performance.

So, while using Raspberry Pi for edge AI opens many doors, it's important to build systems that fit around the platform's shortcomings. If you want your application to run well and remain dependable, select the right models, optimize your code, and handle your hardware efficiently.

8.2. Problems Relating to Software Compatibility

Trying to keep all your AI tools and frameworks in sync on a Raspberry Pi is much like walking a tightrope. Object detection systems built on CNNs frequently cause developers a lot of frustration because of software compatibility problems. Unlike on a regular desktop, Raspberry Pi's design and limits make it so not all AI tools operate correctly.

Although both TensorFlow and PyTorch provide ARM compatibility, getting them to run on Raspberry Pi isn't always easy. TensorFlow Lite is safer and more supported, though everything new it offers usually comes later. PyTorch support has improved recently, but you may encounter issues with certain packages requiring particular pinning, making the development environment more complicated.

Managing dependencies within a codebase is often a big problem. Many popular Python libraries for computer vision and AI applications on the Raspberry Pi usually have overlapping dependencies. Occasionally, updating one component triggers another problem requiring constant repairs. Virtual environments help, yet sometimes compatibility still isn't possible.

Additionally, not all modeling formats or conversion tools act the same way on Raspberry Pi as on x86 systems. Exporting and changing the model format may lead to some problems when it is finally deployed. Some useful model optimization features reported in existing documentation might be missing from edge-optimized versions.

8.3. Environmental Factors Affecting Accuracy

No matter how advanced your neural network is, the way it functions depends greatly on the condition of the environment around it. Detection of real objects in real-time isn't possible just in a lab, as you will often find in the wild, messy conditions where Raspberry Pi operates. The settings in which testing occurs can bring many factors that lower the accuracy.

It's often the lights themselves that make the difference. Models built with images taken in bright conditions will have difficulty processing dark, glared, or shadowy environments. Such settings may result in failures to detect or incorrect alarms. With outdoor work, the problems are also greater because the amount of light changes during the day or over the seasons. When the camera quality is low because of the Raspberry Pi, high-contrast scenes, reflection, and backlighting pose special challenges to convolutional neural networks (CNNs).

In addition, motion blur can be a problem. Because the cameras on these systems are not very strong and processing capacity is low, achieving clear images of objects as they move becomes hard. Blurred pictures of fast-moving objects can make it tough for detection models to find important, accurate information.

Extra things in the background and things that block the subject will confuse the model's understanding. If what you are interested in is either partly covered or close to things that look much the same, the model may not detect it well. As with perspective, differences in object size or angle from those seen in training data can cause the system to misinterpret things.

If your camera catches dust, rain, or smudges in its lens, it can lower the quality of your image and disturb the model's results. Such outdoor solutions are most at risk if they are not suitably protected.

You should tackle these problems by ensuring your model learns from real-world diversity in datasets. You can use data augmentation to add brightness, raise or decrease contrast, and add noise so the training input resembles these conditions. Also, you can use real-time feedback to monitor and respond to changes in the environment.

Even though environmental factors are part of every system's regular operation, you can significantly improve your system's ability to withstand them by preparing for them.

Table 3 Common Environmental Challenges and Mitigation Techniques

Environmental Factor	Impact on Detection	Recommended Solution
Low Light	Missed detections	Train with varied lighting conditions
Motion Blur	Reduced confidence	Use cameras with higher shutter speeds
Occlusion	Partial detection or failure	Use multi-angle data during training
Weather/Dust	Image noise	Add lens protection, use data augmentation

9. Real-World Applications

9.1. Smart Surveillance Systems

Convolutional neural networks on Raspberry Pi are most useful in smart surveillance systems. Regular CCTV systems record video, but someone or a system in the cloud is needed to look at and analyze what is filmed. Cognitive networking processes happen live at the edge when they run directly on a Raspberry Pi.

Picture a security camera that can determine who or what is near, warn you about intruders, tell the difference between humans and pets, and instantly notify you when something out of the ordinary occurs. It's no longer a subject you just read about. Thanks to models such as MobileNet SSD or Tiny-YOLO on a Raspberry Pi, you can recognize objects, monitor motion, and pick out faces while the system runs.

There are big benefits to financial markets. First, since only important things, like motion or someone being in front of the sensor, are detected, bandwidth use is reduced. In addition, it protects privacy because none of your videos are sent away; they are all handled on your device locally. An increased focus on data security is a major reason this helps residential and corporate users.

Moreover, with GPIO pins or relay modules, your Raspberry Pi can also act physically. Should a trespass be detected, the alarm can sound, the lights could switch on, and doors might lock on their own. With IR cameras, your system will see in the dark and work continuously, day and night.

Costing little to build, Raspberry Pi-powered security systems attract the interest of school administrators, small business owners, and homeowners alike. You don't have to use expensive machinery or unique software. A Pi, a camera, and a trained CNN will complete the job.

Smart surveillance is changing our approach to keeping safe. They aren't only for seeing—they think, make judgments, and take immediate action.

9.2. Autonomous Drones and Vehicles

One of the most exciting and complex areas for object detection is automated navigation, and Raspberry Pi is helping make it easier for more people to use. Coupling CNNs with real-time video means drones and small robotic vehicles can sense their environment, decide what to do, and move alone.

You can use CNNs in drones to avoid obstacles, navigate toward objects, and perform visual SLAM (where you both locate yourself and make a map while moving). Thanks to Raspberry Pi, real-time changes the drone senses can be

handled locally, so it goes ahead without having to tap its ground station or server all the time. Because of this, it is most important in remote and disaster-hit places where latency could arise.

That same technology is often used in robotic vehicles to help with warehouse tasks and checking crops on farms. Raspberry Pi technology allows the robot to travel between crops, alert users to any pests or weeds, and send alerts if action is necessary. Now, tasks that used to be time-consuming and monotonous can be managed.

Since Raspberry Pi is both lightweight and cheap, it's very popular for this project. CNN-powered object detection on a Raspberry Pi enables amateurs working on prototypes and engineers working on larger robots to work in the field.

Robotics and AI are also well-suited topics to teach in this technology stack in the classroom. With the board, students can display, in practice, the underlying principles of computer vision, control systems, and embedded AI in just one device.

There is a huge opportunity here. Raspberry Pi is helping make the future happen, where both consumer gadgets and larger machines can understand and handle tasks independently.

9.3. Home Automation and IoT Use Cases

Smart homes and IoT are taking off quickly, largely thanks to CNNs on Raspberry Pi. There are as many ways as there are inventions regarding smart doorbells, intelligent lights, automatic pet feeders, and HVAC systems that sense your presence.

Facial recognition technology is commonly used to manage who gets into a facility. A Raspberry Pi near your door allows it to use a camera to identify family members, open the door, or alter the lighting or music settings. It's like a doorman who never goes home and never makes a simple mistake with a face.

Object detection makes it easier to promote safety and provide better services. Pi can monitor your driveway and tell you when a car is there. With a smart fridge, you wouldn't have to worry about forgetting whether food expires soon, as the fridge will tell you. Even small amounts of creativity allow you to automate chores to make your home feel near-wonderful.

Many people know voice assistants and smart speakers, but visual recognition takes things further. Thanks to its ability to recognize gestures or sign language, the Raspberry Pi lets those with hearing impairments feel more involved in many activities.

For IoT systems that connect and rely on many devices interacting based on what is happening, CNNs give the unique abilities needed for true automation. On the Raspberry Pi, the sensor data is interpreted, what's happening around it is learned, and devices can be controlled in response.

The Raspberry Pi is perfect for building your home automation projects because it is cheap, supported by many, and adjustable. Because of computer vision and deep learning, today's homes are smarter, more than just being connected.

10. Future of CNNs on Raspberry Pi

10.1. Progress in Edge AI

Edge AI is moving forward quickly, and Raspberry Pi is confidently part of that growth. With CNNs getting more efficient and access to tools widening, we can now use data center tools in homes, schools, factories, and on the go.

The expanded collection of pre-trained models designed for edge devices is a big improvement. Not only are frameworks like TensorFlow Lite, PyTorch Mobile, and ONNX Runtime used in edge computing—but they are also being improved. Now, these platforms have smaller systems that are as fast and correct as the original, larger models.

More companies are now paying attention to systems that use both cloud and edge for processing. As a result, inferences are dealt with on the Raspberry Pi itself and synced with the cloud for updates or more thorough examination when necessary. As a result, developers can ensure outstanding performance and accuracy, even when adding extra advanced functions to the edge device.

Another major change is underway in training models. Federated learning, called edge training, is now possible, so Raspberry Pi devices can improve models online without sharing their data directly. It helps keep privacy safe as systems grow in their learning ability.

We're not waiting on Edge AI—we already use it here and now. That intelligence doesn't just come from having a big budget, or a powerful graphics card is being shown with Raspberry Pi. The stronger and faster CNNs get, the more opportunities arise for AI in edge technology.

10.2. Impact of New Raspberry Pi Hardware

As time passes, every new Raspberry Pi launch expands the range of activities that can be managed on a $35–$75 board. Raspberry Pi 5 is a big improvement over its earlier models, with a speedier CPU, more bandwidth for memory, and the ability to use PCIe-based storage and external GPUs.

With these improved pieces of hardware, complex, reliable CNNs run at higher efficiency and speed. More RAM lets your device run larger models, manage different tasks well, and support processes like recording high-resolution video and detecting objects simultaneously.

Most excitingly, PCIe support means using more advanced accelerators and storage than before. Therefore, Raspberry Pi is strengthening its ability to serve serious uses in edge AI and low-cost DIY and school projects.

Better camera modules and added MIPI CSI and DSI support help improve vision applications and make it simpler to include effective video input for improved detection and tracking. Now that hardware matches what developers want to do with software, development is smoother and faster.

Each version of the Raspberry Pi brings AI a step nearer to everyone. When the hardware evolves, you can make your edge computing systems smarter, faster, and able to scale better, delivering all the benefits edge computing promises.

10.3. Trends in TinyML and Lightweight AI

TinyML will shape the direction of AI in small devices, and the Raspberry Pi will drive this change. Its main objective is to produce machine-learning models that work efficiently on small and power-saving controllers. Although the Raspberry Pi is stronger than typical microcontrollers, it is a great platform for building and running TinyML apps.

TinyML focuses on designing models to take up little space and speed in a phone yet still supply useful information. Use motion detection to identify spoken words or tell if someone or something is in an image, all with devices lasting up to months on a single charge.

Thanks to new research in model compression, quantization, and neural architecture search, designers can now fit more into such space-limited devices. Thanks to TensorFlow Lite for Microcontrollers and Edge Impulse, developers can easily make and operate smaller models.

TinyML adds more use and flexibility to what Raspberry Pi users can do with their projects. Rather than using complete models, developers can set up toned-down editions that work smoothly, are better suited to outdoor logistics, and run on less power and connection.

TinyML has great educational potential in the academic sector. It shows students how to use software well and consider hardware boundaries when designing AI.

11. Conclusion

CNNs have completely changed how machines see and interpret things, and teaming them with the flexibility and affordability of Raspberry Pi has created many exciting new opportunities. Thanks to real-time detection in cameras and robotics, machine learning is now changing deployment architecture for edge AI.

We have looked at every necessary facet of getting CNNs working on a Raspberry Pi, from their operation to choosing the right structure, optimizing for faster results, putting hardware together, and looking into real applications. We've taken care of likely challenges, including limits with hardware, waiting for things to process, and existing software, and shown how to solve them with detailed design, efficient models, and tools that run on the hardware.

The exciting thing about this topic is how accessible it is. You don't need a big budget or fancy server to use AI. Now that Raspberry Pi, a camera, and a trained model are accessible, it's possible to build systems that respond to real-time events—which used to seem almost impossible not too long ago.

Things aren't over after this point. As the Raspberry Pi gets updated and new improved CNN architectures are established, the divide between inexpensive and powerful AI devices is becoming much smaller. With Edge AI, TinyML, and federated learning, we are making even smarter and more autonomous systems possible to be both practical and powerful.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] Khalifa, A. F., Elmahdy, H. N., & Badr, E. (2021). Real-time human detection model for edge devices. arXiv preprint arXiv:2111.10653. https://arxiv.org/abs/2111.10653

[2] Mazzia, V., Salvetti, F., Khaliq, A., & Chiaberge, M. (2020). Real-time apple detection system using embedded systems with hardware accelerators: An edge AI application. arXiv preprint arXiv:2004.13410. https://arxiv.org/abs/2004.13410

[3] Zaki, P. S., William, M. M., Soliman, B. K., Alexsan, K. G., Khalil, K., & El-Moursy, M. (2020). Traffic signs detection and recognition system using deep learning. arXiv preprint arXiv:2003.03256. https://arxiv.org/abs/2003.03256

[4] Yang, S., Gong, Z., Ye, K., Wei, Y., Huang, Z., & Huang, Z. (2019). EdgeCNN: Convolutional neural network classification model with small inputs for edge computing. arXiv preprint arXiv:1909.13522. https://arxiv.org/abs/1909.13522

[5] Modrzyk, N. (2020). Real-time IoT imaging with deep neural networks: Using Java on the Raspberry Pi 4. Apress. https://doi.org/10.1007/978-1-4842-5722-7

[6] Karp, R., & Swiderska, Z. (2021). Automatic generation of graphical game assets using GAN (pp. 7–12). https://doi.org/10.1145/3477911.3477913

[7] Khoi, T. Q., Quang, N. A., & Hieu, N. K. (2021). Object detection for drones on Raspberry Pi: Potentials and challenges. IOP Conference Series: Materials Science and Engineering, 1109(1), 012033. https://doi.org/10.1088/1757-899X/1109/1/012033

[8] Jain, M., & Shah, A. (2020). A multi-modal CNN framework for integrating medical imaging for COVID-19 Diagnosis. World Journal of Advanced Research and Reviews, 8(3), 475–493. https://doi.org/10.30574/wjarr.2020.8.3.0418

[9] Jung, H. W. (2019). Optimizing the performance of convolutional neural networks on Raspberry Pi for real-time object detection. Computer Science & Information Technology (CS & IT), 9(17), 121–130. https://aircconline.com/csit/abstract/v9n17/csit91711.html

[10] Velasco-Montero, D., Fernández-Berni, J., Carmona-Galán, R., & Rodríguez-Vázquez, Á. (2018). Performance analysis of real-time DNN inference on Raspberry Pi. In Real-Time Image and Video Processing 2018 (Vol. 10670, p. 106700F). SPIE. https://doi.org/10.1117/12.2309763

[11] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 779–788). https://doi.org/10.1109/CVPR.2016.91

[12] Girshick, R. (2015). Fast R-CNN. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1440–1448). https://doi.org/10.1109/ICCV.2015.169

[13] Mesuga, R., & Bayanay, B. (2021). A deep transfer learning approach to identifying glitch waveform in gravitational wave data. arXiv. https://doi.org/10.48550/arXiv.2107.01863

[14] Kaushik, P., & Jain, M. (2018). Design of low power CMOS low pass filter for biomedical application. International Journal of Electrical Engineering & Technology (IJEET), 9(5).

[15] Kumar, Y., Saini, S., & Payal, R. (2020). Comparative Analysis for Fraud Detection Using Logistic Regression, Random Forest and Support Vector Machine. SSRN Electronic Journal.

[16] Höppner, S., Baesens, B., Verbeke, W., & Verdonck, T. (2020). Instance-Dependent Cost-Sensitive Learning for Detecting Transfer Fraud. arXiv preprint arXiv:2005.02488.

[17] Niu, X., Wang, L., & Yang, X. (2019). A Comparison Study of Credit Card Fraud Detection: Supervised versus Unsupervised. arXiv preprint arXiv:1904.10604.

[18] Bhat, N. (2019). Fraud detection: Feature selection-over sampling. Kaggle. Retrieved from https://www.kaggle.com/code/nareshbhat/fraud-detection-feature-selection-over-sampling

[19] Olaitan, V. O. (2020). Feature-based selection technique for credit card fraud detection. Master's Thesis, National College of Ireland. Retrieved from https://norma.ncirl.ie/5122/1/olaitanvictoriaolanlokun.pdf

[20] Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. (2017). Credit card fraud detection: A realistic modeling and a novel learning strategy. IEEE Transactions on Neural Networks and Learning Systems, 29(8), 3784–3797. https://doi.org/10.1109/TNNLS.2017.2736643

[21] Carcillo, F., Dal Pozzolo, A., Le Borgne, Y. A., Caelen, O., Mazzer, Y., & Bontempi, G. (2019). Scarff: A scalable framework for streaming credit card fraud detection with spark. Information Fusion, 41, 182–194. https://doi.org/10.1016/j.inffus.2017.09.005

[22] West, J., & Bhattacharya, M. (2016). Intelligent financial fraud detection: A comprehensive review. Computers & Security, 57, 47–66. https://doi.org/10.1016/j.cose.2015.09.005

[23] Zareapoor, M., & Shamsolmoali, P. (2015). Application of credit card fraud detection: Based on bagging ensemble classifier. Procedia Computer Science, 48, 679–685. https://doi.org/10.1016/j.procs.2015.04.201

[24] Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. Decision Support Systems, 50(3), 602–613. https://doi.org/10.1016/j.dss.2010.08.008

[25] Patel, H., & Zaveri, M. (2011). Credit card fraud detection using neural network. International Journal of Innovative Research in Computer and Communication Engineering, 1(2), 1–6. https://www.ijircce.com/upload/2011/october/1_Credit.pdf

[26] Puneet Kaushik, Mohit Jain , Gayatri Patidar, Paradayil Rhea Eapen, Chandra Prabha Sharma (2018). Smart Floor Cleaning Robot Using Android. International Journal of Electronics Engineering. https://www.csjournals.com/IJEE/PDF10-2/64.%20Puneet.pdf

[27] Duman, E., & Ozcelik, M. H. (2011). Detecting credit card fraud by genetic algorithm and scatter search. Expert Systems with Applications, 38(10), 13057–13063. https://doi.org/10.1016/j.eswa.2011.04.102

[28] Kaushik, P., Jain, M., & Jain, A. (2018). A pixel-based digital medical images protection using genetic algorithm. International Journal of Electronics and Communication Engineering, 31-37. http://www.irphouse.com/ijece18/ijecev11n1_05.pdf

[29] Kaushik, P., Jain, M., & Shah, A. (2018). A Low Power Low Voltage CMOS Based Operational Transconductance Amplifier for Biomedical Application. https://ijsetr.com/uploads/136245IJSETR17012-283.pdf

[30] Kaushik, P., & Jain, M. (2018). Design of low power CMOS low pass filter for biomedical application. International Journal of Electrical Engineering & Technology (IJEET), 9(5).

[31] Nabati, R., & Qi, H. (2019). "RRPN: Radar Region Proposal Network for Object Detection in Autonomous Vehicles." 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 2019, pp. 3093-3097, doi: 10.1109/ICIP.2019.8803392.

[32] Rawat, W., & Wang, Z. (2017). "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review." Neural Computation, 29(9), pp. 2352-2449, Sept. 2017, doi: 10.1162/neco_a_00990.

[33] Wang, W., et al. (2019). Medical image classification using deep learning. In Intelligent Systems Reference Library (pp. 33–51). https://doi.org/10.1007/978-3-030-32606-7_3

[34] Alom, M. Z., et al. (2018). The history began from AlexNet: A comprehensive survey on deep learning approaches. arXiv. https://arxiv.org/abs/1803.01164

[35] Frid-Adar, M., et al. (2018). GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. Neurocomputing, 321, 321–331. https://doi.org/10.1016/j.neucom.2018.09.013

[36] Jiao, L., & Zhao, J. (2019). A survey on the new generation of deep learning in image processing. IEEE Access, 7, 172231–172263. https://doi.org/10.1109/ACCESS.2019.2956508

[37] Wang, L., Chen, W., Yang, W., Bi, F., & Yu, F. R. (2020). A state-of-the-art review on image synthesis with generative adversarial networks. IEEE Access, 8, 63514–63537. https://doi.org/10.1109/ACCESS.2020.2982224

[38] Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. Journal of Big Data, 6(1), Article 60. https://doi.org/10.1186/s40537-019-0197-0

[39] Kayalibay, B., et al. (2017). CNN-based segmentation of medical imaging data. arXiv. https://arxiv.org/abs/1701.03056

[40] Kaushik, P., & Jain, M. (2018). A low power SRAM cell for high speed applications using 90nm technology. International Journal of Electrical Engineering, 10(2), 6. https://www.csjournals.com/IJEE/PDF10-2/66.%20Puneet.pdf

[41] Esfahani, S. N., & Latifi, S. (2019, July 13). A survey of state-of-the-art GAN-based approaches to image synthesis. In Proceedings of the 9th International Conference on Computer Science, Engineering and Applications (CCSEA 2019). https://doi.org/10.5121/csit.2019.90906

[42] Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031

[43] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In European Conference on Computer Vision (pp. 740–755). Springer. https://doi.org/10.1007/978-3-319-10602-1_48

[44] Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations. https://arxiv.org/abs/1412.6980

[45] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., . . . Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. arXiv.org. https://arxiv.org/abs/1605.08695

[46] Jung, H. W. (2019). Optimizing the performance of convolutional neural networks on Raspberry Pi for real-time object detection. Computer Science & Information Technology (CS & IT), 9(17), 121–130. https://aircconline.com/csit/abstract/v9n17/csit91711.html

[47] Khoi, T. Q., Quang, N. A., & Hieu, N. K. (2021). Object detection for drones on Raspberry Pi: Potentials and challenges. IOP Conference Series: Materials Science and Engineering, 1109(1), 012033. https://doi.org/10.1088/1757-899X/1109/1/012033

[48] Modrzyk, N. (2020). Real-time IoT imaging with deep neural networks: Using Java on the Raspberry Pi 4. Apress. https://doi.org/10.1007/978-1-4842-5722-7

[49] Kaushik, P. (2018). STUDY AND ANALYSIS OF IMAGE ENCRYPTION ALGORITHM BASED ON ARNOLD TRANSFORMATION. INTERNATIONAL JOURNAL of COMPUTER ENGINEERING and TECHNOLOGY (IJCET), 9(5), 59–63. https://iaeme.com/Home/article_id/IJCET_09_05_008