Nottingham Trent University

School of Science and Technology

**Evaluating NVIDIA Jetson Nano Power and Performance for Edge Computing Use**

**(Changed from Rugged Computer Vision Edge Accelerator System)**

by

Ivica Matic

in

2022

Project report in part fulfilment

of the requirements for the degree of

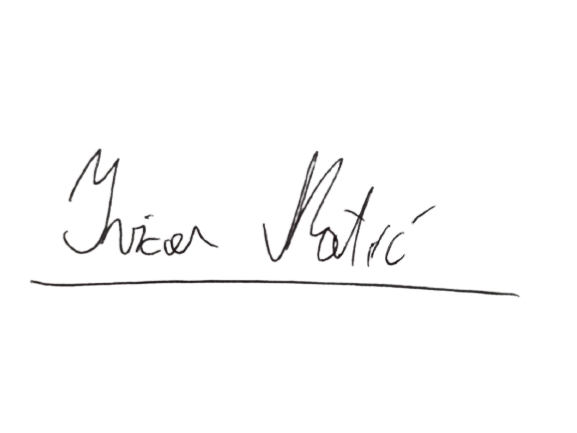
Bachelor of Science with Honours

In

Software Engineering

I hereby declare that I am the sole author of this report. I authorize Nottingham Trent University to lend this report to other institutions or individuals for scholarly research.

I also authorize Nottingham Trent University to reproduce this report by photocopying or by other means, in total or in part, at the request of other institutions or individuals for scholarly research.



Ivica Matić

Abstract

We live in a world filled with Internet-connected electronic devices. Together with internet access being more accessible to people around the world and IoT (internet of things) becoming much more popular every year, we have reached a point where the number of connected devices is surpassing the 50 billion mark and continuing to rise every new day. Many of these devices are used to perform some sort of computer vision operation.

This work will characterise and demonstrate the usage of the NVIDIA Jetson Nano computing board for performing object classification at the Edge, looking at the performance from both data throughput and power usage perspectives. Together with findings on Jetson’s performance and comparison with the other compute board alternatives, brings all the accompanying software used for the evaluation and benchmarking.

Findings show that Jetson Nano, considering its low power and moderate performance, combined with modern object algorithms such as YOLOv5, when effectively used and configured might be a viable alternative to the cloud for providing image inference capabilities to the Edge computing devices.

Acknowledgments

I would like to thank my lecturer, supervisor and mentor, Dr Pedro Machado for all his experience and knowledge shared with me throughout this project and my time at the University. Studying from him and expanding my knowledge from his mentorship throughout these four years was the most pleasurable and joyful academic experience one could ask for.

I would like to thank Mr Flemming Christensen and Sundance Multiprocessor Ltd for providing the LynSyn Lite device used to prove my measurements for this paper.

I am very grateful to my dear friend and university colleague Mr. Petar Peychev. Working alongside him on various projects from the beginning of my university journey all the way to the end was an experience I will never forget, with a lot of new and exciting skills learned from Petar on every occasion.

Special thanks to my family for providing their unconditional love, word of advice and comfort in the harshest times.

I am forever thankful to my flatmates and dearest friends Leonardo and Oliver for being the best support one could ask for.

Table of Contents

[Abstract iii](#_Toc101468275)

[Acknowledgments iv](#_Toc101468276)

[Table of Contents v](#_Toc101468277)

[List of Figures ix](#_Toc101468278)

[List of Tables xii](#_Toc101468279)

[List of AbbreviatIons xiii](#_Toc101468280)

[CHAPTER 1 15](#_Toc101468281)

[INTRODUCTION 15](#_Toc101468282)

[1.1 Brief literature review 17](#_Toc101468283)

[1.2 Alternative solutions and services 20](#_Toc101468284)

[1.2.1 Hardware-based alternatives 20](#_Toc101468285)

[1.2.2 Cloud-based alternatives 26](#_Toc101468286)

[CHAPTER 2 27](#_Toc101468287)

[CONTEXT 27](#_Toc101468288)

[2.1 Jetson Nano 27](#_Toc101468289)

[2.2 Performance figures 29](#_Toc101468290)

[2.3 YOLO 32](#_Toc101468291)

[2.3.1 Performance and accuracy 33](#_Toc101468292)

[2.4 LynSyn Lite 35](#_Toc101468293)

[2.4.1 LynSyn Tool Modification 37](#_Toc101468294)

[CHAPTER 3 39](#_Toc101468295)

[New Ideas 39](#_Toc101468296)

[3.1 New Benchmark Approach 39](#_Toc101468297)

[3.2 New Performance Findings 40](#_Toc101468298)

[CHAPTER 4 41](#_Toc101468299)

[INVESTIGATION 41](#_Toc101468300)

[4.1 Project Planning – Project Tune Down 41](#_Toc101468301)

[4.1.1 Initial project plan 42](#_Toc101468302)

[4.1.2 Risk assessment 43](#_Toc101468303)

[4.2 Power-profiling YOLOv5 models running on Jetson Nano 48](#_Toc101468304)

[4.2.1 Measuring equipment 48](#_Toc101468305)

[4.2.2 Measuring software 49](#_Toc101468306)

[4.2.3 Measuring approach 57](#_Toc101468307)

[CHAPTER 5 60](#_Toc101468308)

[RESULTS / DISCUSSION 60](#_Toc101468309)

[5.1 Elaborating results 60](#_Toc101468310)

[5.1.1 Accuracy 60](#_Toc101468311)

[5.1.2 Performance Figures – Frame Rate 61](#_Toc101468312)

[5.1.3 Understanding Energy Consumption Figures 66](#_Toc101468313)

[5.2 Comparing Edge computing with a cloud solution 70](#_Toc101468314)

[5.3 Results 74](#_Toc101468315)

[CHAPTER 6 75](#_Toc101468316)

[CONCLUSIONS / FUTURE WORK 75](#_Toc101468317)

[6.1 Conclusions 75](#_Toc101468318)

[6.2 Future work 77](#_Toc101468319)

[6.2.1 Try out smaller size neural networks 77](#_Toc101468320)

[6.2.2 Benchmark more devices 77](#_Toc101468321)

[6.2.3 Create more example-software 77](#_Toc101468322)

[6.3 Legal, Social, Ethical and Professional Issues 78](#_Toc101468323)

[6.3.1 Legal issues 78](#_Toc101468324)

[6.3.2 Social issues 78](#_Toc101468325)

[6.3.3 Ethical issues 78](#_Toc101468326)

[6.3.4 Professional issues 79](#_Toc101468327)

[6.4 Synoptic Reflections 80](#_Toc101468328)

[References 81](#_Toc101468329)

[Bibliography 83](#_Toc101468330)

[appendix A. NVIDIA JETSON NANO TECHNICAL SPECIFICATION 86](#_Toc101468331)

[appendix B. NVIDIA JETSON NANO PERFORMANCE COMPARISON 87](#_Toc101468332)

[appendix c. power and performance Measurement results 90](#_Toc101468333)

[appendix d. Calculation for potential battery runtime 93](#_Toc101468334)

[APPENDIX E. Initial project Gannt Chart 94](#_Toc101468335)

[APPENDIX F. Equipment used 95](#_Toc101468336)

List of Figures

[Figure 1: U.S. Edge Computing Market Size from 2017 to 2028 18](#_Toc101468339)

[Figure 2: Global Computer Vision Market size in Billions from 2018 to 2030 19](#_Toc101468340)

[Figure 3: Raspberry PI 4 21](#_Toc101468341)

[Figure 4: Kria KV260 Development Board on a custom 3D Printed stand 23](#_Toc101468342)

[Figure 5: Kria KV260 next to NVIDIA Jetson Nano 24](#_Toc101468343)

[Figure 6: Inference performed on Video Camera attached to KV260 Development Board 25](#_Toc101468344)

[Figure 7: NVIDIA Jetson Nano Single Board Computer 27](#_Toc101468345)

[Figure 8: Comparing Jetson Nano Performance to Coral Dev Board and Raspberry PI3 with Intel Compute Stick 29](#_Toc101468346)

[Figure 9: Jetson Nano performance running various deep learning inference networks 30](#_Toc101468347)

[Figure 10: LynSyn Lite Power Profiling Board 35](#_Toc101468348)

[Figure 11: Jumper Wire Connected to the Microcontroller side of the Status LED Current Limiting Resistor 38](#_Toc101468349)

[Figure 12: Original project Gantt Chart 42](#_Toc101468350)

[Figure 13: Connection diagram for benchmark tests 48](#_Toc101468351)

[Figure 14: LynSyn Lite Viewer Software – Power measurement represented via power over time chart 50](#_Toc101468352)

[Figure 15: LynSyn Lite Viewer Software – Power measurement as represented in a tabular way 51](#_Toc101468353)

[Figure 16: LynSyn Lite Viewer Software – Power Profiling starting form – user prompted to enter power profiling duration 51](#_Toc101468354)

[Figure 17: Benchmark measuring program described via a state diagram showing a continuous benchmarking process 59](#_Toc101468355)

[Figure 18: FPS Performance of various sized YOLOv5 models on Jetson Nano 61](#_Toc101468356)

[Figure 19: YOLOv5 and YOLOV3 Model accuracies – as reported online 62](#_Toc101468357)

[Figure 20: YOLOv5 Network Sizes 63](#_Toc101468358)

[Figure 21: YOLOv5 Inference Performance Comparison between CPU and GPU mode on Jetson Nano 64](#_Toc101468359)

[Figure 22: YOLOv5 Inference Performance in CPU mode on Jetson Nano 65](#_Toc101468360)

[Figure 23: Power consumption per processed frame in mWh/frame 67](#_Toc101468361)

[Figure 24: Power consumption per processed frame in mWh/frame (YOLOv5n and YOLOv5 68](#_Toc101468362)

[Figure 25: Potential Battery Powering Options for Jetson Nano 69](#_Toc101468363)

[Figure 26: Frame rate inference comparison between the Azure Object classification to the YOLOv5 running at the Edge 71](#_Toc101468364)

[Figure 27: Azure Object Detection Performance Compared to Edge Deployed YOLOv5 - mWh/frame 72](#_Toc101468365)

[Figure 28: Azure Object Detection Performance Compared to Edge Deployed YoloV5 (Jetson in CPU only mode) – mWh/frame 73](#_Toc101468366)

[Figure 29: Azure Object Detection Performance Compared to Edge Deployed YoloV5 (Jetson in CPU only mode) – FPS 73](#_Toc101468367)

List of Tables

[Table 1. Free Tier Comparison Between Major Cloud Providers 26](#_Toc101467297)

[Table 2: Performance and accuracy of pre-trained YOLOv5 models trained on the COCO2017 dataset 33](#_Toc101467298)

[Table 3: Initial Risk Assessment 43](#_Toc101467299)

[Table 4: Jetson Nano Power Measuring Files on JetPack Image 55](#_Toc101467300)

[Table 5: Other power-related measuring files 56](#_Toc101467301)

[Table 6: Measurement Results obtained via Jetson Nano 74](#_Toc101467302)

List of AbbreviatIons

AI – Artificial Intelligence

BCS - British Computer Society

CAGR – Computed annual growth rate

CNN - Convolutional Neural Network

CPU - Central Processing Unit

CUDA - Compute Unified Device Architecture

EDGE - Computing paradigm of moving execution devices closer to the data source

ESD – Electrostatic Discharge

EU – European Union

FPGA - Field Programmable Gate Array

FPS - Frames per Second

GPIO - General Purpose Input Output

GPU - Graphics Processing Unit

IC – Integrated Circuit

IO – Input/Output

JTAG - Joint Test Action Group

LED - Light Emitting Diode

MSRP - Manufacturer Suggested Retail Price

RAM - Random Access Memory

SBC - Single Board Computer

SDK – Software Development Kit

SOM – System on module

SSH – Secure Shell

V – Electrical Volts

W – Electrical Watts

YOLO - You Only Look Once, an Object Detection Neural Network

YOLOv3 - Third version of YOLO Object Detection Neural Network

YOLOv5 - Fifth version of YOLO Object Detection Neural Network



INTRODUCTION

The idea of this project is to improve the quality of service for the existing and new embedded devices on the Edge that require the use of computer vision capabilities provided by the cloud. By bridging the gap between Edge deployed IoT devices and their cloud-based processing endpoint goal is to minimise the required energy and latency by bringing the execution device closer to the end-user, minimising the energy yield required to send the data for processing and obtaining the result. Doing so would also see the benefit of reducing the latency which could further improve the reaction time of the embedded systems requiring such functionalities (i.e., smart doorbells with the face-recognition feature, Automatic Number Plate Recognition, etc.).

Other potential advantages of utilising the approach described above are:

* Maximising the throughput due to lower latency

Because the processing endpoint is placed much closer to the execution device latency is significantly lower. Therefore, more frames of video data(images) can be ­­processed in the same amount of time (assuming no transfer speed bottlenecks on the execution device).­­

* Providing capabilities in places with no Internet access

Because we can see smart devices and products everywhere we run into issues when we cannot reliably (or even possibly) connect them to an Internet network. With the endpoint being internet-independent, we can place the equipment in various odd places like mines, underground car parks, and similar, supplying us with computing capabilities even without internet access.

Given the above, the objective of this project is to benchmark an NVIDIA Jetson Nano single-board-computer, a device that could further improve the usability and reliability of the Edge deployed application which requires some sort of computer-vision capabilities that before required communication with some other data processing endpoint on the cloud.

The aims of the project are:

* Evaluate NVIDIA Jetson Nano SBC in terms of its power and performance
* Compare NVIDIA Jetson Nano to other SBCs and platforms

## Brief literature review

One can see technological advancements everywhere we turn in our day to day lives. As technology is becoming more affordable and easier to produce, we can see the rise in the sheer amount of hardware that is deployed throughout the world for many specific applications. Even with the popular prediction of 50 billion internet-connected devices by 2020 being disputed back in 2016, it must be admitted that 46 billion devices connected as of the end of 2021 are a noticeably substantial number (A. Nordrum, 2016).

Taking into consideration that early IoT systems could only send and collect data for analysis and processing, and nowadays, with remarkable advances being made in embedded system-on-a-chip devices enabling us to do much more processing on the Edge side than ever before, we can only assume the number of connected devices will rise even further (N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob and M. Imran, 2018).

The role of Edge computing on the Internet of Things can be observed from the market predictions for the United States.

Chart, bar chart

Description automatically generated

Figure 1:  
U.S. Edge Computing Market Size from 2017 to 2028  
(Lai, 2021)

“In recent years, Edge computing has soared in popularity. According to Grand View Research, the Edge computing market was valued at $4.68 billion in 2020, with an expected compound annual growth rate (CAGR) of 38.4 per cent from 2021 to 2028.” (Lai, 2021).

If we compare the data above with the rise of computer vision global market size, the popularity increase in computer vision applications closely follows the rise of the Edge computing market described above. This proves that computer vision plays a big part in the Edge computing market, driving the future need for Edge deployed computer vision devices even further.

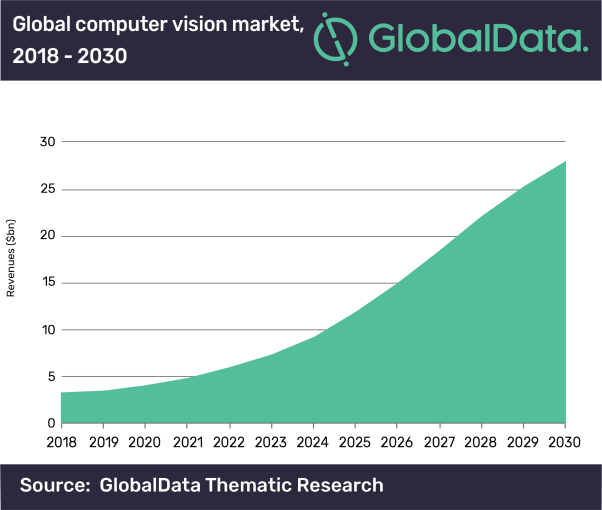


Figure 2:  
Global Computer Vision Market size in Billions from 2018 to 2030  
(Globaldata n.d.)

Given the co-relation above, it can be determined that the rise in the number of deployed IoT systems follows the growth in Edge computing and computer vision popularity. With the number of deploying systems growing rapidly we need to consider the power that would be needed to do all the processing.

Therefore, investing the time and effort to characterise and power profile Edge deployed computer vision applications to better understand their impact and behaviour will more than likely result in highly transferable knowledge which could later prove of value.

## Alternative solutions and services

### Hardware-based alternatives

With the prices of Single Board Computers dropping rapidly, the market is full of alternative computing devices that could carry out the task of computer vision on the Edge. Advancements in embedded technology enabled us to have more processing power at the Edge than ever before while keeping (or lowering) the device size, the required power and overall operating environment impact.

Some of the devices that could be practical candidates for the task are:

* Raspberry PI family devices (Raspberry 3 and 4)
* FPGA Devices (Xilinx Kria KV260)
* Other devices in Jetson Family

#### Raspberry PIs

Raspberry PI is a family of single-board computers in small form factor, developed in the United Kingdom by the Raspberry PI Foundation. Starting production back in 2012, the first generation of Raspberry PI featured modes 256MB of RAM and a single-core Broadcom BCM2835 processor clocked at 700MHz.

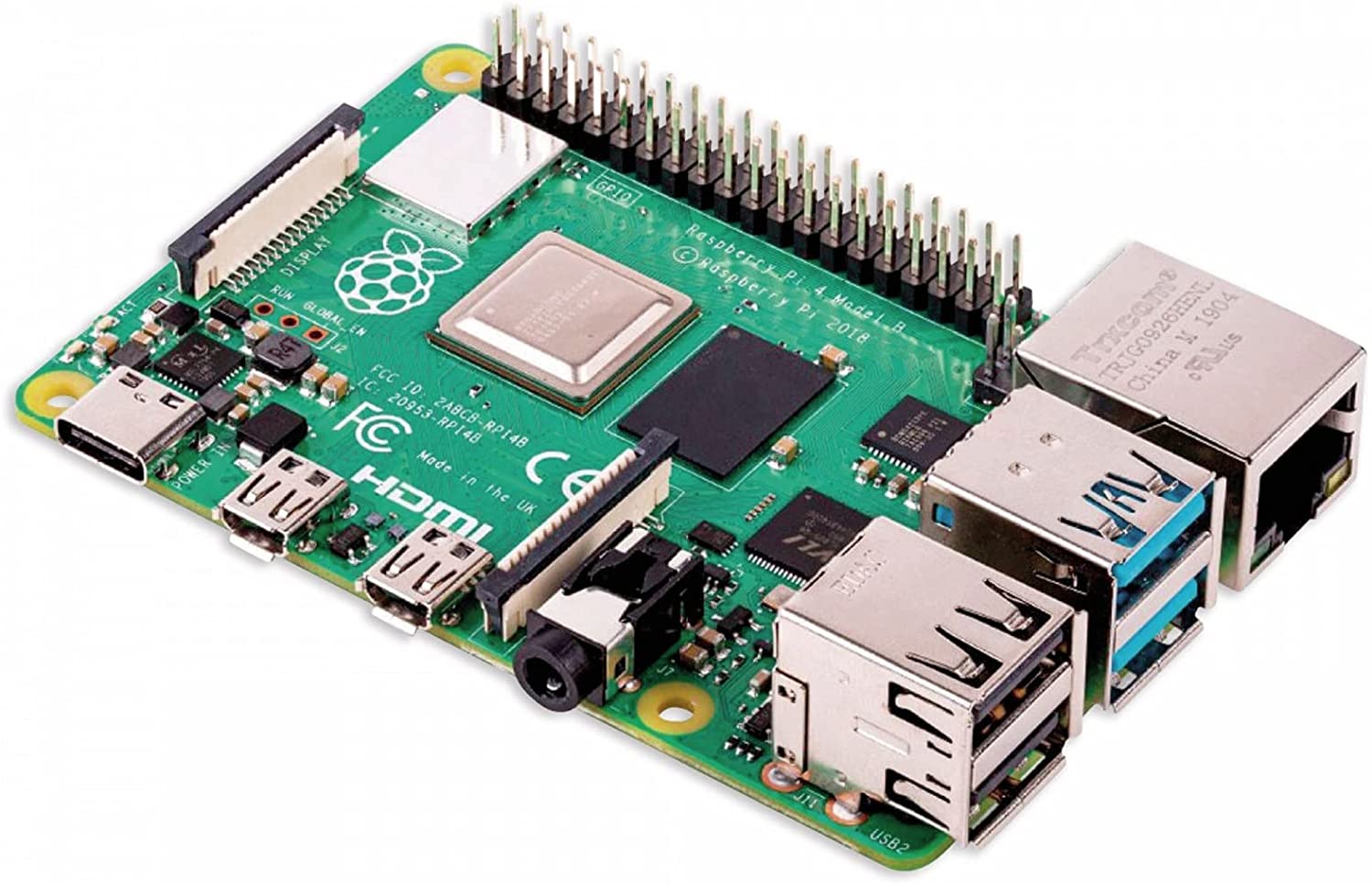


Figure 3:  
Raspberry PI 4  
(Pi Foundation n.d.)

Originally used to promote the teaching of computer science in developing countries, it became much more popular than predicted and was soon widely adopted by hobbyists in their various projects.

The advantage that plays into PIs hand is that it was one of the first SBC boards to catch the eyes of the hobbyists and makers, so today, after more than 10 years of producing various board revisions, the community based around their products is big and thriving with various tutorials and how-to knowledge.

Nowadays, the latest Raspberry PI SBC comes with up to 8Gb of RAM and a Quad-core Cortex-A72 CPU, making it a much better-suited alternative for more demanding Edge computing tasks.

#### Xilinx Kria KV260

Kria KV260 board is Xilinx developed FPGA alternative to classic GPU devices on the Edge, specialised to run advanced vision applications without requiring complex hardware design knowledge.



Figure 4:  
Kria KV260 Development Board on a custom 3D Printed stand

Based on K26 SOM, the Kria KV260 serves as a development kit that can be similarly used by developers just like Jetson Nano, enabling rapid prototyping and model evaluation in the desktop environment before moving to the actual production and running the design on the field.



Figure 5:  
Kria KV260 next to NVIDIA Jetson Nano

With the MSRP price of 199 USD quoted on Xilinx's official website, the KV260 is placed similarly close to NVIDIA Jetson Nano, making it a practical alternative for deploying the FPGA passed alternative on the Edge. However, because it is necessary to convert standardly trained neural network models which were trained and meant to be run on the GPU devices, there is a limitation on which neural networks can be used, as they need to be quantized to use INT8 weights instead of floating-point numbers so they can be run on FPGA architecture.

Quantization is a process in which original floating-point weights of the neural network are converted to integer values, calibrated by including a subset of original training (but unlabelled) images. Some accuracy loss is incurred in this process, but the performance of the network is increased significantly, especially for devices optimised to work with integer numbers, i.e., FPGAs.

On the performance side, the KV260 manages to achieve around 13 fps while using the INT8 Quantized version of YOLOV3 CNN, trained on the COCO2014 dataset, representing 80 common classes, while running at around 10 W average power (reported via power profiling tool built-in into kv260 Vitis AI image).

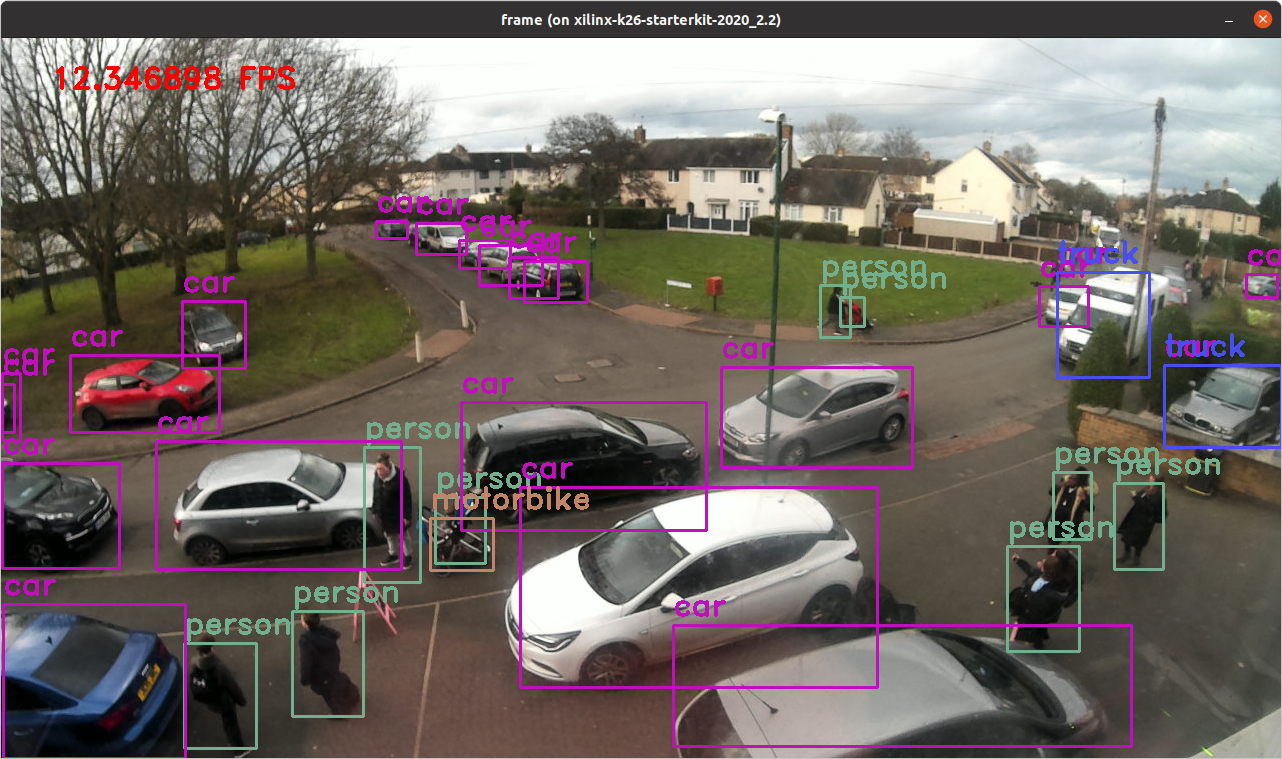


Figure 6:  
Inference performed on Video Camera attached to KV260 Development Board

From the performance described above, and given the previously stated introductory price, the KV260 board looks like a viable alternative to NVIDIA Jetson Nano for embedded computer vision applications. The biggest drawback of the KV260 approach would be the rather limited ability of the Vitis AI Toolchain used to quantize the CNNs to run on the K26 device.

### Cloud-based alternatives

With the Computer Vision services being widely used increasingly with each new day, instead of just running our local machines with GPU capabilities, nowadays there are online services that can be utilised to carry out computer vision tasks. This usually includes sending frames (or videos) to a remote endpoint on the internet and obtaining the response, in which inference results (or any other) are included.

Most of the time, if there is the ability to access network resources, it is more cost-effective to employ cloud-based Computer Vision services than hosting them locally, due to the initial hardware cost of obtaining the appropriate equipment and associated energy costs of running inference equipment throughout the whole year. Even further, many cloud providers, including AWS (Amazon Web Services), Azure, and Google Cloud Platform all offer free tier Computer Vision service, further lowering introductory costs, while allowing developers to do their research and optimise their Edge applications. For some smaller-scale projects, free tier service might even suffice for production use.

Table 1. Free Tier Comparison Between Major Cloud Providers

|  |  |
| --- | --- |
| Service / Cloud Provider | Number of calls in the free tier |
| AWS Rekognition API | Free tier for 12 months, 5000 images per month |
| Google Cloud Vision API | 1000 images per month |
| Microsoft Azure Computer Vision API | 5000 images per month |



CONTEXT

## Jetson Nano

Jetson Nano is a single-board computer developed by NVIDIA Corporation. Its primary use is to enable quick and reasonably priced entry-level access to deploying Edge AI applications and devices.

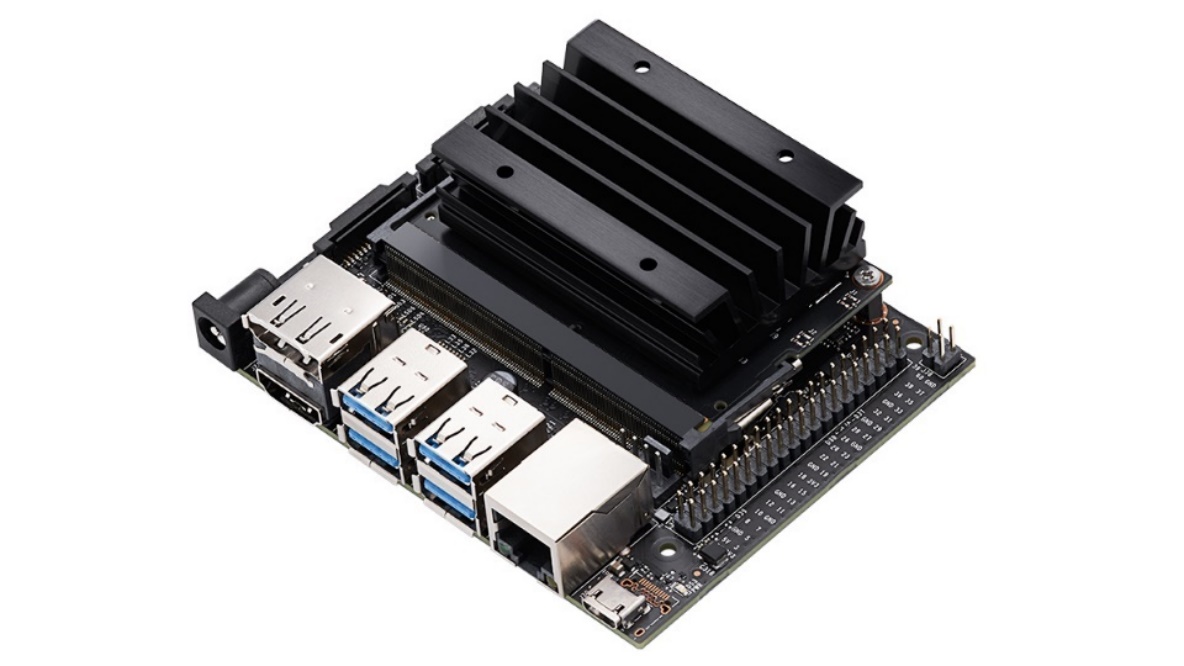


Figure 7:  
NVIDIA Jetson Nano Single Board Computer  
(NVIDIA n.d.)

Given its low introductory price of 99 USD, small footprint size of not more than 70x70 mm and low power usage, which, with further configuration can be as low as 5W, Jetson Nano is a suitable computing board for accelerating computer-vision based functionalities at the Edge.

Jetson Nano is a small development kit style computer that let us accelerate various tasks such as neural networks for image classification, object detection, segmentation, and speech processing.

In addition to its low size and power usage, Jetson Nano, together with other products from NVIDIA’s SBC line-up is becoming more popular with the increase in Edge devices with computer-vision/AI purposes. The considerable influence comes from NVIDIA’s forum, and the community formed around the product line-up, supplying great tutorials and examples.

NVIDIA Jetson Boards (and all other SBCs from NVIDIA) are officially supported via the NVIDIA Jetpack SDK, including all Linux drivers and packages together with bootloader, Ubuntu desktop environment and a complete set of libraries for GPU accelerated computing.

Jetpack is also a good starting point for using NVIDIAs SBC platforms, as it features many samples, documentation, and developer tools for both SBCs and standard development machines.

Detailed technical specifications of the Jetson Nano can be found in Appendix A.

## Performance figures

With the Jetson Nano being widely used by makers and communities around the world, and the fact that it is not the latest product (introduction date: March 2019), some benchmark data is available on the internet.

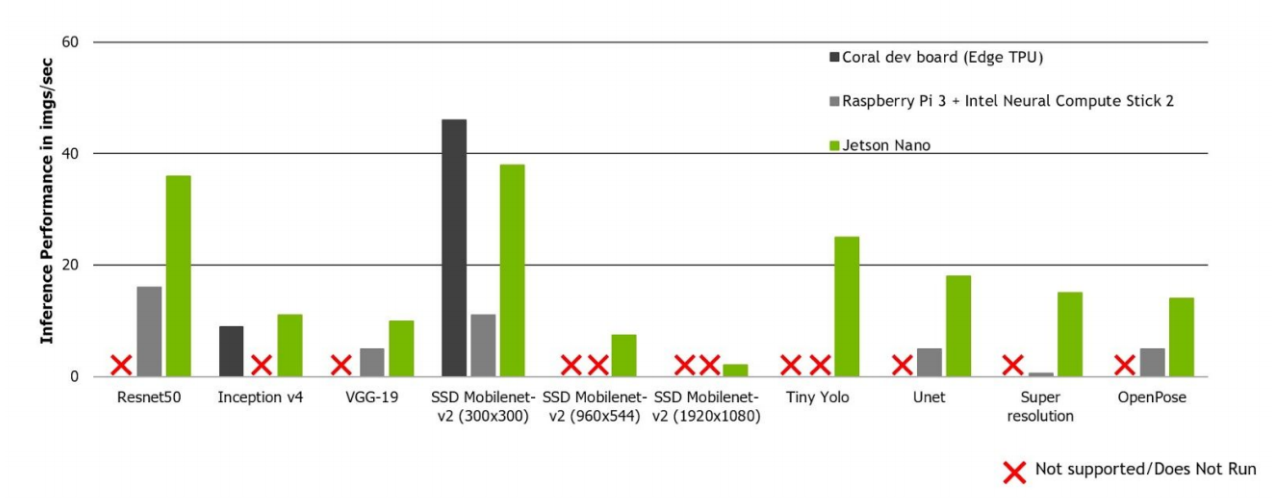


Figure 8:  
Comparing Jetson Nano Performance to Coral Dev Board and Raspberry PI3 with Intel Compute Stick  
(Introducing the NVIDIA Jetson Nano, 2019)

From the figure above, we can see that Jetson Nano surpasses the performance of the Coral Dev Board and is just marginally slower than Raspberry PI3 + Intel Neural Compute Stick 2 combination.

The big factor that brings a performance boost to the Jetson Nano is the fact that the GPU device onboard supports NVIDIA CUDA.

CUDA is a parallel computing programming model created by NVIDIA to help developers accelerate their applications using NVIDIA GPU devices. The whole ecosystem features more than 150 CUDA-based libraries enabling the developers' easy access to even the most powerful acceleration functionalities.

Thankfully, using high-level Python libraries such as PyTorch and TensorFlow, application developers do not need to worry about writing CUDA code as the libraries above provide the abstraction level enabling ease of use.

Therefore, as the Jetson Nano board holds a GPU on board, it can utilize CUDA and it is better supported, which we can see from the fact that the other two compute solutions were not able to complete all the benchmarks from the chart above.



Figure 9:  
Jetson Nano performance running various deep learning inference networks  
(Jetson Benchmarks, n.d.)

From the above, we can see that the Jetson Nano board delivers acceptable and usable performance with various deep learning inference networks and frameworks. Of particular interest to us is Tiny YOLOv3 – Running at 416x416px resolution, managing to score a decent 25 fps, as we are going to use the newer YoloV5 for our usage examples and performance benchmarks.

As we are interested closely in the YOLOv5 performance and power consumption of the system running the models, in our benchmarks we will capture the performance of the system and power usage using the LynSyn Lite profiling tool and on-board power measuring sensor(s) while running diverse sizes of YOLOv5 models. See section 3.

The table with benchmark data from the figure above is available in Appendix B.

## YOLO

The algorithm You Only Look Once (YOLO) is an object detection algorithm that works by dividing the input image into smaller chunks in which every chunk handles detecting the objects within its bounds.

At the current moment, there have been officially released five versions of the YOLO algorithm, with the latest, YOLOv5 being released in May 2020. The biggest novelty of YOLOv5 is that it is implemented using the Pytorch framework, making it more appealing to the new users and more supported on various hardware.

The YOLOv5 neural network is trained on high-performance GPU devices or accelerators, with the models trained on the COCO2017 [[1]](#footnote-2)dataset being published on the YOLOv5 official GitHub page. Because training is a very intensive and costly process, in this work the pre-trained YOLOv5 neural networks will be used.

The most popular previous version of the YOLO algorithm is YOLOv3, developed by Joseph Redmon and Ali Farhadi.

The biggest difference between YOLOv3 and YOLOv5 is that version 5 is implemented in PyTorch while version 3 (and all versions before) are implemented in Darknet, an open-source neural network framework written in C and CUDA.

### Performance and accuracy

Table 2: Performance and accuracy of pre-trained YOLOv5 models trained on the COCO2017 dataset (Jocher 2021)

Table

Description automatically generated

To evaluate the performance of YOLOv5, its creator, Ultralytics[[2]](#footnote-3), trained the different size versions of the network on the COCO2017 dataset to create the benchmark data from the figure above. From the benchmark data present, we are mostly interested in the mean-average precision values (mAP), as the “Speed” columns are created with hardware way more powerful than available on the Jetson Nano development board (NVIDIA Tesla V100 GPU).

From the figure above we can see the difference in the parameter count of the YOLOv5 neural network directly corresponds to the accuracy and the speed of the model. By upgrading to the next model size, we can see that accuracy rises at the cost of speed.

In section 4 we will expand on this knowledge by presenting power-profiling data while running various size YOLOv5 models with different input sources.

## LynSyn Lite

A picture containing text, electronics, circuit

Description automatically generated

Figure 10:  
LynSyn Lite Power Profiling Board

LynSyn Lite is a power-profiling device developed by Sundance Multiprocessor Ltd. Featuring three sensors that measure both current and voltage, LynSyn lite can power profile any device with power requirements in the range of onboard sensors.

LynSyn Lite can be synchronised with the host board via the JTAG interface, but as Jetson Nano currently does not support JTAG debug capabilities we will synchronise the measurement via Jetson’s GPIO pins.

Primarily developed for power-profiling ARM applications on Xilinx FPGA devices, LynSyn Lite also serves the purpose of a JTAG programming tool.

As we are not going to use JTAG, we will have LynSyn running in a free-run mode, triggering the Jetson Nano via GPIO pin to start and stop the program for the measurement.

### LynSyn Tool Modification

As the LynSyn tool is designed to debug host applications using JTAG sampling, and our device under test, Jetson Nano does not support JTAG Debugging we needed to find an alternative way to synchronise the LynSyn measurement tool and benchmark software running on Jetson Nano.

After a quick observation of the LynSyn Lite tool, it was clear that there is a status LED light, which turns on and stays on for the duration of the whole measurement process. Measuring the voltage at the current limiting resistor input side, we can see a 3.3V signal while the LED is off and a 0V signal while the LED is on. This shows that the other side of the LED is connected to the 3.3V and the logic is inverted.

As the voltage range of 0~3.3V perfectly fits the GPIO inputs on the Jetson Nano development board, we will use those to trigger our benchmark measurements for the validation.

A picture containing text, electronics, circuit

Description automatically generated

Figure 11:  
Jumper Wire Connected to the Microcontroller side of the Status LED Current Limiting Resistor



New Ideas

## New Benchmark Approach

Currently, the big emphasis on the benchmark procedures is placed on raw speed-based performance. Given that this work aims to evaluate the use of Jetson Nano in the Edge use case scenario, the emphasis is placed not only on the throughput performance but also on the energy consumption aspect.

By using the onboard power sensors, confirmed using the LynSyn Lite power measuring tool, this work serves as a practical demonstration of how very accurate power measuring benchmarks can be done on a budget.

## New Performance Findings

With the Jetson Nano and the whole world of embedded electronics based on computer vision gaining traction, the performance findings elaborated in this report will be useful for determining if the Jetson Nano is a practical solution for many budget-oriented projects.



INVESTIGATION

## Project Planning – Project Tune Down

Originally intended, the project was supposed to be an implementation project showing NVIDIA Jetson Nano in the Edge environment by creating all supporting hardware, ranging from 3D printed enclosure to custom power solution design. With no ability to obtain the external equipment outside the University, the project had to be tuned down excluding the above.

### Initial project plan

Graphical user interface

Description automatically generated with low confidence

Figure 12: Original project Gantt Chart

### Risk assessment

Taken from the initial Project Planning Document, the risk assessment table is included in the table below.

The risks no longer applicable are highlighted in the table.

Table 3: Initial Risk Assessment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Risk | Probability | Severity | Action |
| 1 | Chip shortage makes it impossible to obtain required computing boards and other hardware modules | Medium | High | Obtain all computer hardware at beginning of the project, from UK sources.  Limit the project scope to a single technology type (for example, do not try to implement both FPGA and GPU computing boards) |
| 2 | Computer hardware develops a fault due to ESD discharge | Medium | High | Follow the guidelines for working safely with ESD sensitive devices.  Use the ESD mat while working with the computer board.  While equipment is not in use, store it in the ESD bags that the boards came in. |
| 3 | Computer hardware develops physical failure | Low | High | Keep the equipment in the original boxes and foam packaging both for transporting and when not in use. |
| 4 | Unable to manufacture project enclosure using 3D printing due to printer malfunction | Medium | Medium | Adequate project enclosure can be purchased online. |
| 5 | Unable to manufacture project enclosure using 3D printing due to limited CAD knowledge | Medium | Medium | Adequate project enclosure can be purchased online. |
| 6 | Equipment gets destroyed due to catastrophic battery failure | Low | High | Get the battery from a reputable supplier in the UK. Make sure that the system has adequate battery protection features. Protect the batteries against short-circuit, under-discharge, over-voltage and over-temperature. |
| 7 | Unable to obtain image dataset for training demo object detection neural networks | Low | Low | As object-detection neural networks do not require a large dataset for successful training, a small dataset can be obtained manually using photo cameras and annotated using an open-source annotation tool. |
| 8 | Unable to train a neural network to produce the working project demos | Low | Low | The pre-trained neural networks can be found online. |
| 9 | Codebase loss due to personal computer damage or failure | Low | Low | Use platforms like GitHub or GitLab to store and work on code. |
| 10 | Codebase not completed by the project end date due to complexity issues | Low | Low | Use project planning features of GitHub or GitLab. |
| 11 | Loss of project documentation | Low | Low | Do not store documentation locally, use platforms like OneDrive, Google Docs or Overleaf to write project documentation and report. |
| 12 | Filesystem for the solution hardware gets corrupted or lost | Low | Low | Keep an image of the SD card media from which you could rebuild the system easily.  Be careful about updating system software on board. Read the update manifests first. |
| 13 | Project milestones are not met | Low | Medium | If not on time to meet the milestones, instead of pushing milestones further up the timeline try to go back and review/reduce the project features. |
| 14 | Unable to demonstrate the project | Medium | Medium | Try and create a good video demo that will highlight all aspects of the project. |

## Power-profiling YOLOv5 models running on Jetson Nano

### Measuring equipment

Aside from the LynSyn Lite device already mentioned we need some other miscellaneous cables and components that have not been mentioned yet. These components include:

1. System Power Supply
2. Ethernet switch or a router
3. 2x RJ45 Terminated Ethernet Cables
4. Laptop/Pc capable of sustaining SSH connection to Jetson Nano

A full list of components and tools used is available in Appendix F.

The overall measuring system hardware can be described in the following figure:

Diagram

Description automatically generated

Figure 13:  
Connection diagram for benchmark tests

### Measuring software

To capture the performance of the Jetson Nano board running diverse sizes of YOLOv5 Object Detection Neural Network, appropriate software had to be run on both Jetson Nano and the computer connected to the LynSyn Lite power measuring device.

List of software used:

* LynSyn Lite Viewer
* Performance Benchmark Program
* On-board power measuring Program

#### LynSyn Lite Viewer Program

LynSyn Lite Viewer is a support software used to interface LynSyn devices. It offers the users all power measuring capabilities in a nice and compact computer program from which the measuring operations can be achieved. In addition to power measuring, host-device applications can be inspected with a graphical view of the power utilised, measurements results can be saved and much more.

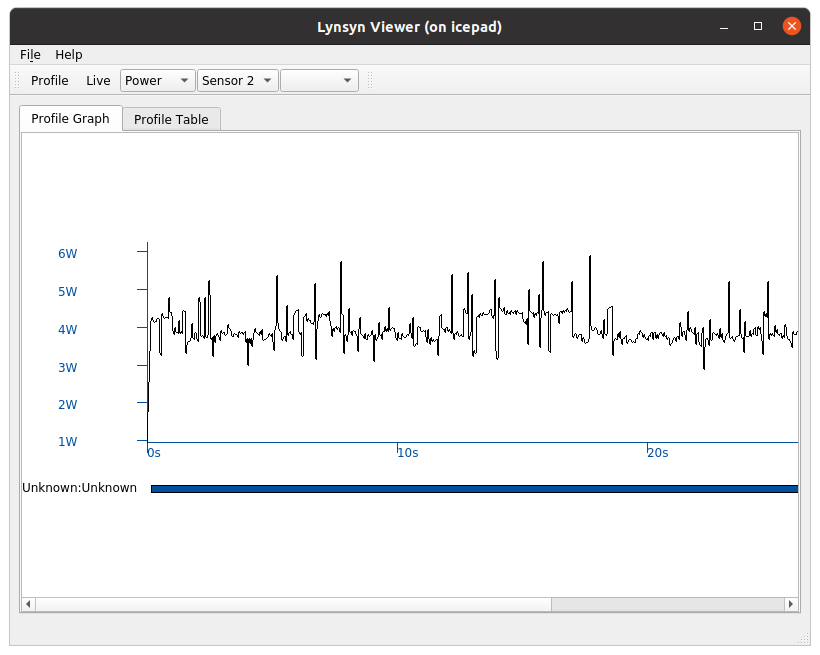


Figure 14:  
LynSyn Lite Viewer Software – Power measurement represented via power over time chart

Graphical user interface, text, application, email

Description automatically generated

Figure 15:  
LynSyn Lite Viewer Software – Power measurement as represented in a tabular way

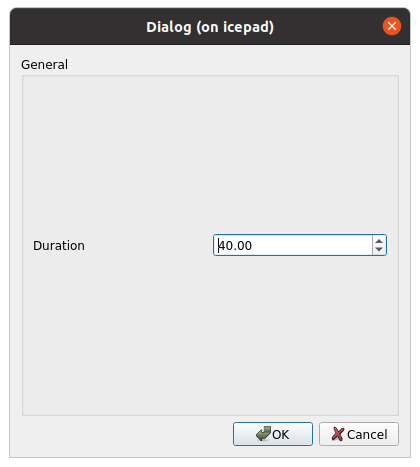


Figure 16:  
LynSyn Lite Viewer Software – Power Profiling starting form – user prompted to enter power profiling duration

The LynSyn software, together with all build and run instructions can be found at the Norwegian University of Science and Technology (NTNU) GitHub Repository: <https://github.com/EECS-NTNU/lynsyn-host-software>.

#### Performance Benchmark Program

To evaluate the performance of Jetson Nano SBC Computer, the accompanying benchmark program was created and included in this report. The benchmark program uses the object detection capabilities inherited from YOLOv5 supplied samples by including the YOLOv5 repository as a Python module and then writing a suitable interface wrapper around it.

The program works by loading the test data into the main memory to avoid any potential IO bottlenecks. This data consists of frames that will be passed to an object detection neural network for inference operations. Next, the neural network itself is loaded from the PyTorch model file and the model is ready to accept input images.

With the images and network being loaded, the program is now waiting for the input trigger via Jetson Nano GPIO pins. Once the GPIO is driven digitally LOW via the LynSyn line, the continuous inference operation is started, saving the number of processed frames and average frame times. At this stage, the Performance Benchmark program is starting the thread instance of the Onboard power measuring program for internal power measurement too. Once the measuring is done, LynSyn pulls the GPIO line back up to digitally HIGH and the measuring loop stops, displaying the performance metrics to the user on the standard output.

Different versions of this program exist for running the benchmark for both input types being images and video/camera.

#### Onboard power measuring Program

Observing the Jetson carrier board and technical documentation it was clear that the Jetson Nano Carrier board features INA3221 IC. INA3221 is 3 channel high-side shunt measuring device, which in Jetson’s implementation is used to watch over the various powerlines, including the main power line which is monitored via the LynSyn lite device.

Further investigating it was found that the power usage data can be simply obtained due to fact that the appropriate driver is built-in into the Jetpack OS image, by piping the output of the specific file. This meant that it would be easy to create a simple python program that would obtain the power usage from this file and keep it for logging purposes, displaying the results to the user when necessary.

By including this data, it is possible to further evaluate the results obtained with the LynSyn device with greater precision, as more data would be present and logged.

The channels/files available for measuring power this way is:

Table 4: Jetson Nano Power Measuring Files on JetPack Image

|  |  |
| --- | --- |
| File Path | Description |
| /sys/bus/i2c/drivers/ina3221x/6-0040/iio:device0/in\_power1\_input | SoC power utilisation |
| /sys/bus/i2c/drivers/ina3221x/6-0040/iio:device0/in\_power2\_input | CPU power utilisation |
| /sys/bus/i2c/drivers/ina3221x/6-0040/iio:device0/in\_power1\_input | GPU power utilisation |

In addition to the above, it is also possible to obtain more power-related details from the INA3221 sensor using filenames in the table below:

Table 5: Other power-related measuring files

|  |  |
| --- | --- |
| File Path | Description |
| /sys/bus/i2c/drivers/ina3221x/6-0040/iio:device0/in\_current1\_input | SoC current draw |
| /sys/bus/i2c/drivers/ina3221x/6-0040/iio:device0/in\_current2\_input | CPU current draw |
| /sys/bus/i2c/drivers/ina3221x/6-0040/iio:device0/in\_current1\_input | GPU current draw |
| /sys/bus/i2c/drivers/ina3221x/6-0040/iio:device0/in\_voltage1\_input | Voltage on SoC power-rail |
| /sys/bus/i2c/drivers/ina3221x/6-0040/iio:device0/in\_voltage2\_input | Voltage on CPU power-rail |
| /sys/bus/i2c/drivers/ina3221x/6-0040/iio:device0/in\_voltage1\_input | Voltage on GPU power-rail |

### Measuring approach

As we are trying to capture the power profile of an object-detection application running on Jetson Nano we are going to run all YOLOv5 model sizes with various input sources while capturing the total consumed power of the Jetson Nano development board by connecting it to a power supply via LynSyn Nano power-profiling tool and monitoring the onboard power measuring sensors.

To minimise measuring errors, all measurements will be repeated multiple times with different measuring durations, mainly:

* 20 seconds, synchronised via GPIO
* 40 seconds, synchronised via GPIO
* 60 seconds, synchronised via GPIO

Also, to provide more data for comparison between energy consumptions of different sized models, multiple input sources were used to feed the object detection neural network. These are:

* COCO2017 Test Dataset
* 640x480 Video File

The measuring process works by starting a benchmark program on the Jetson Nano via SSH interactive console. On the program initialization, image test data from COCO2017 is loaded into program memory to alleviate the potential bottlenecks which could occur while trying to read the test data directly from the SD card memory storage.

Afterwards, the YOLOv5 model is loaded, and a few images are passed to its input to alleviate any potential performance issues that could arise due to not initialized/default parameters.

Now that both images and the neural network model are loaded, the program waits for the signal on the GPIO line which is driven by the LynSyn Lite via the signal wire modification described in an earlier chapter. Once the signal is at the digital OFF position, the program in charge of monitoring the internal power consumption is started in a separate thread alongside the benchmark program. While the signal is at a digitally OFF position (0V, status LED light-up) the inference performance is measured, taking the frame count and time needed to process every frame. When the LynSyn tool has stopped taking measurements, the performance data is displayed in the program output together with onboard recorded power use and the entire process can be started again.

If we were to display our program with the state diagram, our program would look like this:

Diagram

Description automatically generated

Figure 17:  
Benchmark measuring program described via a state diagram showing a continuous benchmarking process



RESULTS / DISCUSSION

## Elaborating results

Using the tools and process described above, multiple measurements were taken for different each YOLOv5 Neural Network sized model, running the inference both from supplied images and test videos.

The results were logged in a spreadsheet (data available in a separate file and Appendix C).

### Accuracy

Due to having incoming power usage data from both LynSyn Lite and onboard power measuring sensors, the relative data inaccuracy against the average value was calculated using the following formula:

Formula 1: Relative inaccuracy calculation

Because measurements were not carried out with high-precision laboratory equipment, it is not possible to determine the absolute accuracy figure, but using two independent measuring devices, yielding low absolute inaccuracy ranging from 0.1 to 3% maximum, we can be confident that our measured results are representative of the real values being measured.

### Performance Figures – Frame Rate

From the measurements taken it can be seen, that at best, with the smallest YOLOv5 network size – YOLOv5n, the Jetson Nano board manages to achieve a throughput of 12 frames per second.

Chart

Description automatically generated

Figure 18:  
FPS Performance of various sized YOLOv5 models on Jetson Nano

Comparing that figure of 12fps with 25 frames per second inference speed from the benchmark in chapter 2 makes YOLOV3-Tiny at first look like the better alternative.

However, keeping in mind that YOLOV3-Tiny results in mean-average precision of just 33.1% compared to YOLOv5n’s 45.7% it might be more appropriate to use YOLOv5n dependent on the use-case scenario.

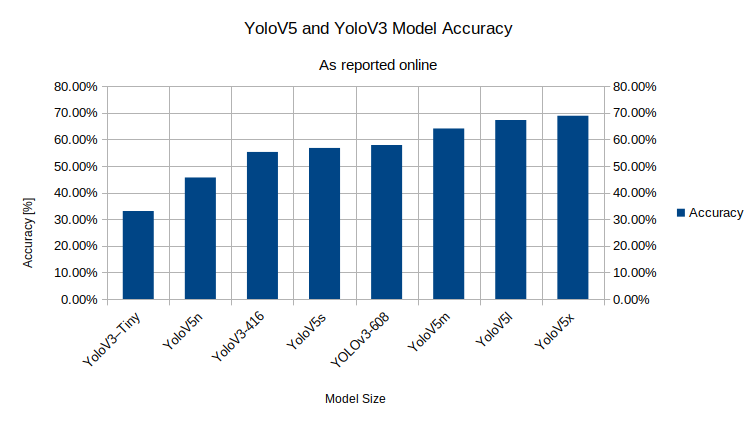


Figure 19:  
YOLOv5 and YOLOV3 Model accuracies – as reported online

For the biggest size YOLOv5 neural network – the YOLOv5x, the average inference speed was just shy of 1 processed frame per second. The performance of YOLOv5x on Jetson Nano is incredibly low but keeping in mind that the YOLOv5x manages accuracy of near 70%. For the tasks that need high detection accuracy without the need for real-time performance, YOLOv5x might be a suitable algorithm to use.

The network size with regards to the number of parameters can be described in the figure below:

Chart, bar chart

Description automatically generated

Figure 20:   
YOLOv5 Network Sizes

#### Jetson Nano in CPU Only Mode

With the Jetson Nano featuring a Quad-core ARM Cortex-A57 processor onboard, like the other close competing products like Raspberry PI 4 line-up, setting the YOLOv5 network to run inference on the CPU we can get a rough example of the performance we can expect.

From the benchmark results captured we can see that by just having a GPU device on the Edge, we can speed up our object detection tasks by 30 times (12 fps for YOLOv5n running in GPU mode vs only 0.4 FPS for the same size neural network running in CPU only mode).

Chart, bar chart

Description automatically generated

Figure 21:   
YOLOv5 Inference Performance Comparison between CPU and GPU mode on Jetson Nano

Chart, bar chart, waterfall chart

Description automatically generated

Figure 22:  
YOLOv5 Inference Performance in CPU mode on Jetson Nano

### Understanding Energy Consumption Figures

As shown above, the mWh/frame (milliwatt-hour per frame) figure was used to describe the power consumption of the whole NVIDIA Jetson Nano Development board while running inference operation on the input frames.

The mWh/frame figure describes how much energy is consumed for obtaining the inference results on a single frame. This figure is calculated from the electrical and performance measurements and is calculated with the following formula:

Formula 2: Energy consumption formula

Formula 3: Conversion formula between Wh and mWh per frame

The lower mWh/frame figure means that less power is consumed by running the inference operations. Taking the measurements from this work as an example, the YOLOv5x needed around 2.5 milliWatt-hours of energy to process one frame, while the smallest YOLOv5n only needed 0.15 mWh/frame, meaning that YOLOv5n is around 16 times more power efficient while running on Jetson Nano.

Chart, bar chart

Description automatically generated

Figure 23:  
Power consumption per processed frame in mWh/frame

The other advantage of having the GPU device on the Edge, besides the inference performance is the power profile of the inference operation. With the GPU accelerated inference being much faster the total energy consumption is much lower.

The mWh/frame figure for Jetson Nano running the inference with the GPU enabled acceleration was 0.15 mWh/frame. Compared to 2.18 mWh/frame for when the GPU was disabled, and 30 times speedup in processing speed we can see the benefit of having a GPU device on the Edge. Less energy will be consumed while delivering greater performance.

Chart, bar chart

Description automatically generated

Figure 24:  
Power consumption per processed frame in mWh/frame (YOLOv5n and YOLOv5

#### Potential powering configuration for Jetson Nano

Given the measured mWh/frame figures, it is possible to determine the potential power sources for running the Jetson Nano development board out on the Edge without mains connected power supply. Given its potential deployment environment, the power performance is one of the crucial factors that should be taken into consideration if the system is to be powered (or backed up) by a battery power source.

The table in Appendix D lists a few potential sources for powering the Jetson Nano on the Edge. The entries in the table assume that all energy of the storage device can be extracted without triggering Undervoltage protection or causing brownouts for the Jetson Nano.

Chart, bar chart

Description automatically generated

Figure 25:   
Potential Battery Powering Options for Jetson Nano

## Comparing Edge computing with a cloud solution

Motivated with low performance of Jetson Nano while running bigger YOLOv5 neural networks sizes such as YOLOv5x resulting in under one processed frame per second with around 2.5 mWh/frame energy consumption, the Azure Computer Vision service was utilised to evaluate the potential performance of the Jetson Nano sending the frames to the cloud for processing.

Azure Computer Vision service is Microsoft’s Cloud service capable of running object detection on the cloud, together with other functionalities like image classification, image description, crime detection from video/photo sources and many others.

To make sure there are no bottlenecks on the cloud, the paid Azure Computer Vision instance – Azure S1 was used, capable of receiving 10 frames per second, located in the geographically closest region at the time of writing, the UK (United Kingdom) SOUTH. The same functionality can be utilised via the free instance of the service above, but the access is limited to 20 frames per minute.

From the measurement figures, it can be observed that the cloud performance is limited mostly by network throughput. Achieving only ~2fps from the Azure S1, the performance is placed somewhere between locally running YOLOv5m and YOLOv5l.

Chart

Description automatically generated

Figure 26:   
Frame rate inference comparison between the Azure Object classification to the YOLOv5 running at the Edge

Regarding the power consumption, just like for the FPS performance, it is placed somewhere between locally running YOLOv5m and YOLOv5l. On average, the energy needed for sending one frame and obtaining the inference results from the cloud was around 2 mWh per frame.

Chart, bar chart

Description automatically generated

Figure 27:  
 Azure Object Detection Performance Compared to Edge Deployed YOLOv5 - mWh/frame

The area where Cloud Computer Services come to their best is for those devices that do not have any kind of GPU (or otherwise) accelerated capabilities on board.

When the measurements were compared to CPU only inference results on Jetson Nano, it is clear that utilising Azure S1 resulted in both better power and energy performance.

Chart, bar chart

Description automatically generated

Figure 28:   
Azure Object Detection Performance Compared to Edge Deployed YoloV5   
(Jetson in CPU only mode) – mWh/frame



Figure 29: Azure Object Detection Performance Compared to Edge Deployed YoloV5 (Jetson in CPU only mode) – FPS

## Results

The results of the measurements are included in the table below. More details on these can be found in Appendix C of this report.

Table 6: Measurement Results obtained via Jetson Nano

|  |  |  |
| --- | --- | --- |
| Neural Network | Average framerate [frames/second] | Average energy consumption per frame [mWh/frame] |
| YOLOv5n | 12.34 | 0.154 |
| YOLOv5s | 6.082 | 0.348 |
| YOLOv5m | 2.78 | 0.793 |
| YOLOv5l | 1.588 | 1.446 |
| YOLOv5x | 0.91 | 2.531 |



CONCLUSIONS / FUTURE WORK

## Conclusions

The goal of the project was to capture and analyse the performance of the NVIDIA Jetson Nano. Using the appropriate benchmark programs and tools this was achieved and documented in the section above. From the data, it is clear that the Jetson Nano thrives while using the lowest size YOLOv5 neural network, the YOLOv5n. With the YOLOv5n being the fastest, the power consumption per frame figure is also the best for YOLOv5n, making it the best choice to provide object detection capabilities on the Edge. In addition, doing the performance measurement of Azure Cloud Services, it is apparent that Edge computing can make use of Cloud-based products if appropriate hardware is not available onboard.

From the performance figures captured, it can be observed that Jetson Nano is a very capable board for running various computer vision operations on the Edge. Alone the fact that Jetson supplies a GPU acceleration device on a small SBC makes the Jetson Nano perform better from both performance and power perspectives. Having the GPU onboard the energy needed to process one frame running the YOLOv5 Inference was multiple times lower than when trying to do the same with the CPU only.

Jetson Nano also features plenty of processing power to provide image operations that could also be performed on the cloud (i.e., Azure), all without access to the internet or any other need for connection whatsoever. For the systems without the GPU devices cloud services are a valid (and in that case better) solution, but from the measurements taken Jetson Nano does not need those services to fulfil computer vision tasks.

Finally, having a GPU device on board that features full CUDA support is making Jetson Nano, and other boards in the NVIDIAs lineup an easy way to provide acceleration capabilities on the edge. With popular frameworks and libraries such as TensorFlow, Darknet and PyTorch being supported, the community around these products is thriving and is full of how-to knowledge and good tips and advice for many use cases and applications.

## Future work

### Try out smaller size neural networks

Due to limited GPU access, it was not possible to train smaller sized YOLOv5 neural networks to compare their performance. Pre-trained YOLOv5 neural networks come in 640x640 resolution. With the other standard sizes being 416x416 (for YOLOV3) and 224x224 (various Resnet versions), the YOLOv5 performance might be further improved by reducing the network size.

### Benchmark more devices

Because of the limited budget, it was not possible to benchmark more devices to compare them with Jetson Nano. The results in Appendix C could be expanded with more data from similar-priced devices such as Kria KV260 or Raspberry PI 4.

### Create more example-software

Originally intended to be a longer hardware-based project, the project scope had to be tuned down because very low budget which didn’t include any funds, only the pre-existing equipment at the university. With the limited scope of the project, the only software created was benchmark tools. This effectively converted this work into a research project.

## Legal, Social, Ethical and Professional Issues

### Legal issues

Due to a fact that this project deals with software published online using various licences when recreating and citing this work, great care must be taken to properly reference and acknowledge the required parts of the writing/code.

The datasets used to do the benchmarks for this work are available online, if the work was to be replicated using a custom dataset, the care should be taken to follow all data protection guidelines and guidelines from the BCS code of conduct.

### Social issues

Social issues of AI include those issues regarding job security and re-distribution of wealth if AI was to take over some if not all job positions currently filled in by human beings. Even with this work not directly affecting the popularity of AI, it could positively depict the Jetson Nano board, drawing more popularity to it and creating various products based on it, which could in turn compromise job security for some individuals or industries.

### Ethical issues

With the work being in the AI domain, all the ethical issues that arise with any AI projects are also applicable here. These issues include a lack of trust from the public and a lack of trust in technology and development. Great care should be taken that the work represents and follows the good ethical guidelines set by various institutions such as BCS, EU and others applicable.

### Professional issues

Because the content produced for this assignment is experimental student work, it might not be the best representative of the profession. It’s needed to make sure that the potential readers and reviewers outside the University are aware that this piece of work and all produced contents and materials are produced for the coursework. Nevertheless, all guidelines from the BCS code of conduct should be followed.

## Synoptic Reflections

Working on this project was a very satisfactory experience leading to many new skills being gained and earlier skillset further expanded and improved. Working with multiple fields such as software engineering, AI-related tasks, power profiling and basic electric engineering brought many new key points and valuable lessons to learn.

Working with open source YOLOv5 enabled interaction with the community surrounding the YOLO project. In addition to the YOLO community, the interaction with the community around Jetson Nano and other NVIDIA products was a good learning experience resulting in a confidence boost and motivation to do the best work to contribute back to the community itself.

Working on this project enabled combining all the skills mentioned above to better understand processes such as training neural networks, and a deeper understanding of neural network structures, together with Final Year Modules brought the understanding of neural networks and their principles even further to a next level.

The project also expanded current knowledge of how to write a good and well-documented code, together with all accompanying documentation.

And finally, this project proved to be a good exercise and learning experience on how to write an academic piece to elaborate on determined findings and will serve as a starting point to reflect for many new projects to come.

References

A. Nordrum, "The internet of fewer things [News]," in IEEE Spectrum, vol. 53, no. 10, pp. 12-13, October 2016, doi: 10.1109/MSPEC.2016.7572524.

C. Wisultschew, A. Otero, J. Portilla and E. de la Torre, "Artificial Vision on Edge IoT Devices: A Practical Case for 3D Data Classification," 2019 XXXIV Conference on Design of Circuits and Integrated Systems (DCIS), 2019, pp. 1-7, doi: 10.1109/DCIS201949030.2019.8959857.

Globaldata.com. 2020. *Computer vision will be worth $28bn by 2030*. [online] Available at: <https://www.globaldata.com/computer-vision-will-be-worth-28bn-by-2030/> [Accessed 9 March 2022].

Hackster.io. 2019. *Introducing the NVIDIA Jetson Nano*. [online] Available at: <https://www.hackster.io/news/introducing-the-nvidia-jetson-nano-aaa9738ef3ff> [Accessed 14 April 2022].

Jocher, G., 2021. *GitHub - ultralytics/yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite*. [online] GitHub. Available at: <https://github.com/ultralytics/yolov5> [Accessed 16 April 2022].

Lai, Z., 2021. *Storage for Edge Computing: Bringing Digital Transformation to the Edge - Phison Blog*. [online] Phison Blog. Available at: <https://phisonblog.com/storage-for-Edge-computing-bringing-digital-transformation-to-the-Edge/> [Accessed 9 March 2022].

N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob and M. Imran, "The Role of Edge Computing in Internet of Things," in IEEE Communications Magazine, vol. 56, no. 11, pp. 110-115, November 2018, doi: 10.1109/MCOM.2018.1700906

NVIDIA Developer. n.d. *Jetson Benchmarks*. [online] Available at: <https://developer.nvidia.com/embedded/jetson-benchmarks> [Accessed 14 April 2022].

NVIDIA Developer. n.d. *Jetson Nano Developer Kit*. [online] Available at: <https://developer.NVIDIA.com/embedded/jetson-nano-developer-kit> [Accessed 12 January 2022].

Odemakinde, E., 2022. *The Top 10 Computer Vision APIs in 2022 - viso.ai*. [online] viso.ai. Available at: <https://viso.ai/computer-vision/computer-vision-apis/> [Accessed 14 April 2022].

Oh, F., 2012. *What Is CUDA | NVIDIA Official Blog*. [online] NVIDIA Blog. Available at: <https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/> [Accessed 21 April 2022].

Pi Foundation, R., n.d. *Teach, Learn, and Make with Raspberry Pi*. [online] Raspberry Pi. Available at: <https://www.raspberrypi.org/> [Accessed 16 April 2022].

Bibliography

A. Nordrum, "The internet of fewer things [News]," in IEEE Spectrum, vol. 53, no. 10, pp. 12-13, October 2016, doi: 10.1109/MSPEC.2016.7572524.

C. Wisultschew, A. Otero, J. Portilla and E. de la Torre, "Artificial Vision on Edge IoT Devices: A Practical Case for 3D Data Classification," 2019 XXXIV Conference on Design of Circuits and Integrated Systems (DCIS), 2019, pp. 1-7, doi: 10.1109/DCIS201949030.2019.8959857.

CompTIA. 2015. *Sizing Up the Internet of Things | Internet of Things | CompTIA*. [online] Available at: <https://www.comptia.org/content/research/sizing-up-the-internet-of-things> [Accessed 14 April 2022].

Globaldata.com. 2020. *Computer vision will be worth $28bn by 2030*. [online] Available at: <https://www.globaldata.com/computer-vision-will-be-worth-28bn-by-2030/> [Accessed 9 March 2022].

Hackster.io. 2019. *Introducing the NVIDIA Jetson Nano*. [online] Available at: <https://www.hackster.io/news/introducing-the-nvidia-jetson-nano-aaa9738ef3ff> [Accessed 14 April 2022].

Jahre, M. and Djupdal, A., 2022. *GitHub - EECS-NTNU/lynsyn-host-software: Host software for the Lynsyn-family of non-intrusive power profilers*. [online] GitHub. Available at: <https://github.com/EECS-NTNU/lynsyn-host-software> [Accessed 16 April 2022].

Jocher, G., 2021. *GitHub - ultralytics/yolov5: YOLOv5 🚀 in PyTorch > ONNX > CoreML > TFLite*. [online] GitHub. Available at: <https://github.com/ultralytics/yolov5> [Accessed 16 April 2022].

Lai, Z., 2021. *Storage for Edge Computing: Bringing Digital Transformation to the Edge - Phison Blog*. [online] Phison Blog. Available at: <https://phisonblog.com/storage-for-Edge-computing-bringing-digital-transformation-to-the-Edge/> [Accessed 9 March 2022].

N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob and M. Imran, "The Role of Edge Computing in Internet of Things," in IEEE Communications Magazine, vol. 56, no. 11, pp. 110-115, November 2018, doi: 10.1109/MCOM.2018.1700906

NVIDIA Developer. n.d. *Jetson Benchmarks*. [online] Available at: <https://developer.nvidia.com/embedded/jetson-benchmarks> [Accessed 14 April 2022].

NVIDIA Developer. n.d. *Jetson Nano Developer Kit*. [online] Available at: <https://developer.NVIDIA.com/embedded/jetson-nano-developer-kit> [Accessed 12 January 2022].

Odemakinde, E., 2022. *The Top 10 Computer Vision APIs in 2022 - viso.ai*. [online] viso.ai. Available at: <https://viso.ai/computer-vision/computer-vision-apis/> [Accessed 14 April 2022].

Oh, F., 2012. *What Is CUDA | NVIDIA Official Blog*. [online] NVIDIA Blog. Available at: <https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/> [Accessed 21 April 2022].

Pi Foundation, R., n.d. *Teach, Learn, and Make with Raspberry Pi*. [online] Raspberry Pi. Available at: <https://www.raspberrypi.org/> [Accessed 16 April 2022].

PowerBank20. n.d. *What is the real output capacity of a Power* *bank?*. [online] Available at: <https://powerbank20.com/en/actual-output-capacity/> [Accessed 14 April 2022].

Prasa Infocom & Power Solutions Pvt. Ltd. 2020. *Data Center Energy Consumption* *vis-à-vis Rising Internet Usage - I*. [online] Available at: <https://prasa-pl.com/blog/data-center-energy-consumption-vis-a-vis-rising-internet-usage-i/> [Accessed 14 April 2022].

APPENDIX A. NVIDIA JETSON NANO TECHNICAL SPECIFICATION

A picture containing text

Description automatically generated

Available on 1st page of: [https://developer.NVIDIA.com/embedded/dlc/jetson-nano-system-module-datasheet](https://developer.nvidia.com/embedded/dlc/jetson-nano-system-module-datasheet)

APPENDIX B. NVIDIA JETSON NANO PERFORMANCE COMPARISON

| **Model** | **Application** | **Framework** | **NVIDIA Jetson Nano** | **Raspberry Pi 3** | **Raspberry Pi 3 + Intel Neural Compute Stick 2** | **Google Edge TPU Dev Board** |
| --- | --- | --- | --- | --- | --- | --- |
| **ResNet-50**  (224×224) | Classification | TensorFlow | 36 FPS | 1.4 FPS | 16 FPS | DNR |
| **MobileNet-v2**  (300×300) | Classification | TensorFlow | 64 FPS | 2.5 FPS | 30 FPS | 130 FPS |
| **SSD ResNet-18**  (960×544) | Object Detection | TensorFlow | 5 FPS | DNR | DNR | DNR |
| **SSD ResNet-18**  (480×272) | Object Detection | TensorFlow | 16 FPS | DNR | DNR | DNR |
| **SSD ResNet-18**  (300×300) | Object Detection | TensorFlow | 18 FPS | DNR | DNR | DNR |
| **SSD Mobilenet-V2**  (960×544) | Object Detection | TensorFlow | 8 FPS | DNR | 1.8 FPS | DNR |
| **SSD Mobilenet-V2**  (480×272) | Object Detection | TensorFlow | 27 FPS | DNR | 7 FPS | DNR |
| **SSD Mobilenet-V2**  (300×300) | Object Detection | TensorFlow | 39 FPS | 1 FPS | 11 FPS | 48 FPS |
| **Inception V4**  (299×299) | Classification | PyTorch | 11 FPS | DNR | DNR | 9 FPS |
| **Tiny YOLO V3**  (416×416) | Object Detection | Darknet | 25 FPS | 0.5 FPS | DNR | DNR |
| **OpenPose**  (256×256) | Pose Estimation | Caffe | 14 FPS | DNR | 5 FPS | DNR |
| **VGG-19**  (224×224) | Classification | MXNet | 10 FPS | 0.5 FPS | 5 FPS | DNR |
| **Super Resolution**  (481×321) | Image Processing | PyTorch | 15 FPS | DNR | 0.6 FPS | DNR |
| **Unet**  (1x512x512) | Segmentation | Caffe | 18 FPS | DNR | 5 FPS | DNR |

APPENDIX C. POWER AND PERFORMANCE MEASUREMENT RESULTS

Chart

Description automatically generated

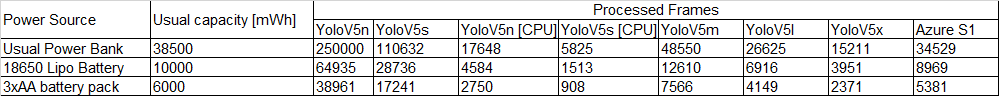
A picture containing diagram

Description automatically generated

A picture containing text, cabinet

Description automatically generated

APPENDIX D. CALCULATION FOR POTENTIAL BATTERY RUNTIME



# APPENDIX E. Initial project Gannt Chart

Graphical user interface

Description automatically generated with low confidence

# APPENDIX F. Equipment used

* NVIDIA Jetson Nano
* Lenovo ThinkPad L480
* Lenovo ThinkPad X240
* Mikrotik RB941 hAP Lite
* Opus DCX2.180/240 (180W/240W) Power Supply
* LynSyn Lite
* See3CAM USB camera
* 2x RJ45 Patch Cables
* 1x MicroUSB Cable
* 64GB MicroSD Memory Card

1. Available online <<https://cocodataset.org>>, last accessed <21.04.2022> [↑](#footnote-ref-2)
2. Company website: <https://ultralytics.com/> [↑](#footnote-ref-3)