# Estimating the Power Consumption of Heterogeneous Devices When Performing AI Inference

Ivica Matic, Francisco de Lemos, Pedro Machado,
Isibor Kennedy Ihianle, David Ada Adama, Jordan J. Bird
*Department of Computer Science*
School of Science and Technology
Nottingham Trent University
Nottingham, UK
Emails: ivica.matic2018@my.ntu.ac.uk,
{francisco.lemos,pedro.machado,isibor.ihianle,david.adama,jordan.bird}@ntu.ac.uk

***Abstract*—Modern-day life is driven by electronic devices connected to the internet. The emerging research field of the Internet-of-Things (IoT) has become popular, just as there has been a steady increase in the number of connected devices. Although these devices are utilised to perform Computer Vision (CV) tasks, it is essential to understand their power consumption against performance. We report the power consumption profile and analysis of the NVIDIA Jetson Nano board while performing object classification. The authors present an extensive analysis regarding power consumption per frame and the output in frames per second using YOLOv5 models. The results show that the YOLOv5n outperforms other YOLOV5 variants in terms of throughput (i.e. 12.34 fps) and low power consumption (i.e. 0.154 mWh/frame).**

***Index Terms*—Internet of things, edge computing, NVIDIA Jetson Nano, Power consumption, Deep learning inference**

## I. Introduction

The rise of Artificial Intelligence (AI) and the continuous generation of Big Data are creating computational challenges. Central Processor Unitss (CPUs) are not enough to efficiently run state-of-the-art AI algorithms or process all the data generated by a wide range of sensors. World-leading processing technology companies (such as NVIDIA, AMD, Intel, ARM and Xilinx) have been looking closely into the new requirements. They have been pushing the boundaries of technology for delivering efficient and flexible processing solutions.

Heterogeneous computing refers to the use of different types of processor systems in a given scientific computing challenge. Heterogeneous platforms are composed of different types of computational units and technologies. Such media can be composed of multi-core CPUs, Graphics Processor Unitss (GPUs) and Field Programmable Gate Arrays (FPGAs) acting as computational units and offering flexibility and adaptability demanded by a wide range of application domains [1]. These computational units can significantly increase the overall system performance and reduce power consumption by parallelising concurrent operations that require substantial CPU resources over long periods.

Accelerators like GPUs and FPGAs are massive parallel processing systems that enable accelerating portions of code that are parallelisable. Combining CPUs with GPUs and FPGAs helps improve the performance by assigning different computational tasks to specialised processing systems. GPUs are optimised to perform matrix multiplications in parallel, which is the major bottleneck in video processing and computer graphics. Normally, images are captured by cameras and the image stream is sent back to the cloud for processing using AI algorithms. This process consists of visual data to a remote endpoint on the internet and obtaining the response, in which inference results are included. Generally, due to the initial hardware cost of obtaining the appropriate equipment and associated energy costs of running inference equipment, it is cost-effective to employ cloud-based CV services than hosting them locally, provided there is internet access. Moreover, many cloud providers, including Amazon Web Services (AWS), Azure, and Google Cloud offer free tier CV services, lowering the initial costs, while allowing users to run their experiments and optimise their applications.

For some small-scale projects, free tier service might be sufficient. The goal of this work is to improve the quality of service for the existing and new embedded devices that require the use of embedded CV capabilities, which are normally provided by the cloud. Bridging the gap between edge deployed Internet of Things (IoT) devices and their cloud-based processing endpoint, has the following advantages: 1) throughput maximisation due to lower latency. Since the processing endpoint is placed much closer to the execution device, the latency is significantly lower; 2) providing CV capabilities in places with no internet access. Since the endpoint devices are internet independent; and 3) reduces the power consumption and data storage needs because the data is processed locally

The authors would like to thank Mr Flemming Christensen and Sundance Multiprocessor Ltd. for providing the LynSyn Lite board, which was crucial for obtaining the results reported in this paper.

and the volume of post-processed data to be transmitted to the cloud is substantially less than raw data that is streamed directly from cameras.

The NVIDIA Jetson Nano (NJN)[1] was selected over other heterogeneous Single Board Cumputers (SBCs) because it was available in the laboratory where the experiences were performed. The research aimed at evaluating the NJN in terms of power consumption and CV throughput. The remainder of this paper includes the literature review in Section II review, the description of the CV hardware and software used in this work in Section III, the description of the measurement procedure in Section IV, the presentation of the results obtained and their evaluation in Section V, and finally the conclusions that and future work in Section VI

## II. Literature review

The number of internet-connected devices being used for customised applications worldwide is rapidly increasing as technology becomes more accessible and simple to use. Although the estimate of 50 billion internet-connected devices by 2020 was contested back in 2016, it must be acknowledged that 46 billion linked devices by the end of 2021 represent a remarkably sizable quantity [2].

The first generation IoT devices can only send and collect data for analysis for further processing. Advances in the later generations of IoT are remarkable considering the heterogeneous Multi-Processor system-on-Chips (MPSoCs) [3] and Adaptive Compute Acceleration Platforms (ACAPs) [4] which enable users to perform more complex operations such as running state-of-the-artAI algorithms at the edge [5].

The integration of AI with IoT, also known as Artificial Intelligence of Things (AIoT), can have several advantages in various domains and form ubiquitous ecosystems of intelligent devices working together. Nevertheless, there are significant challenges that must be overcome before the full realisation of its potential. AIoTs help to improve our lives in a variety of ways, from making daily jobs simpler to enhancing our health and happiness [6]. Efforts are being made to develop model compression and acceleration approaches to deploy Deep Learning (DL) algorithms on mobile and embedded devices to better satisfy the real-time application constraints and user privacy protection [7]–[9].

IoT networks could experience failure as a result of increased data traffic brought on by more IoT nodes [10]. The biggest problem with IoT networks is that they might not have enough memory to manage all the transaction data they need to handle. The solution to this challenge is data compression, a process that reduces the number of bits required to represent data [11]. Data compression can reduce network bandwidth requirements, increase speed file transfers, and conserve space on storage systems [12]. With a minor accuracy loss, model compression reduces the complexity and resources required to compress those models. Model compression techniques include parameter reduction, encoding, encryption, and quantisation [13], [14].

You Only Look Once (YOLO) [15] is a state-of-the-art Convolutional Neural Network (CNN) that accurately detects objects in real-time. This method processes the entire image using a single neural network, then divides it into parts and forecasts bounding boxes and probabilities for each component. The predicted probability weighs these bounding boxes. The technique "only looks once" at the image, since it only does one forward propagation loop through the neural network before making predictions.

The classified objects are displayed after non-max suppression to ensure that each object is only identified once. Yang et al. [16] propose a method based on YOLOv5 [17] to recognise faces wearing masks. When people entered a store, they had to stand in front of a camera, and if recognition succeeded, they could enter through the gate. Additionally, the network was able to classify even when a mask is worn but does not cover the nose.

In general, computer tasks consume variable amounts of energy, and the more processing, the more energy is consumed. Yu et al. [18] proposes the use of the LynSyn Lite Board (LSLB) for measuring power consumption and the Power Measurement Utility Reinforcement Learning (PMU-RL) algorithm to dynamically adjust the resource utilisation of heterogeneous platforms in order to minimise power consumption. The PMU-RL algorithm learns from the power consumption patterns and measurements and controls the programmable logic clock states. Each estimated state of the clock is rewarded when the power consumption is decreased without deteriorating the application performance.

The revised literature shows that it is important to better understand the impact of running state-of-the-art AI algorithms at the edge. Therefore, the authors benchmarked five variants of YOLOv5 on the NJN and the LSLB for measuring power consumption for each YOLOv5 variant and their outputs in frames per second (fps).

## III. Methodology

The utilised methods and hardware platform are discussed in this section.

### A. Hardware Platform

The NJN is a heterogeneous platform which was designed to run efficient state-of-the-art AI applications. The NVIDIA Jetpack Software Developer Kit (SDK) features a full set of libraries for GPU-accelerated computing, Linux drivers, and the Ubuntu operating system. The on-chip GPU can be programmed using NVIDIA's Compute Unified Device Architecture (CUDA) for accelerating complex and parallelisable algorithms. CUDA cores are Nvidia's GPU equivalents of CPU cores. They are built to handle several calculations at once, which is an important feature for accelerating DL algorithms.

Nowadays, AI frameworks such as PyTorch and TensorFlow are already integrated with the high-performance NVIDIA

[1]Available online, https://developer.nvidia.com/embedded/jetson-nano-developer-kit, last accessed 10/09/2023

libraries to abstract the AI developers from the GPU hardware complexity. Therefore, the NJN was selected over other hardware platforms to benchmark the different YOLOv5 architectures. The LSLB was connected to the NJN power rails for measuring the power consumption during the AI inference. Finally, the LSLB profiling tool and the onboard power sensors were used to measure the power consumption whilst running each of the YOLOv5 variants.

### *B. Deep Learning Architecture*

YOLOv5 was selected over other DL methods because it is composed of five variants[2], namely the nano, small, medium, large, and extra-large corresponding to YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x respectively. The YOLOv5 was implemented using the Pytorch framework [3] for delivering a user-friendly environment and optimised to leverage from the GPU when required. The different YOLOv5 variants were benchmarked using the COCO2017 dataset and the performance of YOLOv5 was performed using the Ultralytics[4] which is the recommended AI and deployment platform. The COCO2017 pre-trained weights were used because the objective of this work is to estimate the power consumption per neural network variant.

### *C. LynSyn Lite Board*

LSLB[5] is a power-profiling tool that was designed for monitoring the power consumption of SBCs. Moreover, the LSLB can measure both voltages and current from the power rails in the target device under test and is utilised to extract the power profile for that device. The LSLB can be connected to the NJN through the Joint Test Action Group (JTAG), but the NJN currently lacks JTAG debug capabilities, and therefore the measurement will be done by connecting the NJN General Purpose Input/Output (GPIO) pins.

Although the LSLB also serves the purpose of a JTAG programming tool, it was specifically developed for power-profiling Application Response Measurement (ARM) on Xilinx FPGA devices.

## IV. Measurements procedure

The LSLB viewer application is used to capture the power usage of the NJN board running the different variants of the YOLOv5. Figure 1 shows the hardware setup where it can be seen that the power supply powers NJN through the power rails of the LSLB. The LSLB is also wired to the NJN through the GPIO of both boards. A laptop is connected to the NJN using the ethernet port and to the LSLB through the USB bus.

Additionally, a custom Performance Benchmark Program (PBP) application was designed to benchmark the performance of NJN and collect the power measurements from the LSLB while each of the YOLOv5 variants runs on the NJN. Figure 2 depicts a representation of the different states of the PBP algorithm.

The PBP was initialised on the NJN using ssh. The PBP loads the COCO2017 test dataset and the target YOLOv5 variants. The test images were exposed to the target YOLOv5 variants while the PBP monitors the signal events exchanged through GPIO circuit between the NJN and LSLB. The PBP starts a thread alongside the target YOLOv5 variants while the GPIO is asserted low. The power measures, frame count, and time needed to process each frame are recorded while the signal is asserted low. The performance data is displayed on the screen and the measurements are stored once the process is completed (i.e. once the signal is asserted high). To minimise measurement errors, all measurements were synchronised using GPIO and repeated for intervals of 20, 40 and 60 seconds. To provide more data for comparison between the power consumption of different YOLOv5 variants, using the COCO2017 test dataset and a simple $640x480$ video were utilised.

The power consumption was computed using both the LSLB and the built-in power sensor data. The relative error $R_{error}$ was computed as function of the instaneous power measure $p_{measured}$ and the average powera for all the measurements $p_{measured}$ using Equation 1 [19]:

$$R_{error}[\%] = \left| 1 - \frac{\Delta(p_{meaured} - p_{average})}{p_{average}} \right| .100 \qquad (1)$$

## V. Results and Evaluations

The power measurement was performed using the procedure described in the previous section using Equation 1. Multiple measurements were taken per each YOLOv5 variant when running the AI inference on the test images and video. The measurements were carried out in the laboratory with limited precision and therefore is not possible to determine the absolute error.

### *A. Performance*

Figure 3 shows the that the YOLOv5n has an average throughput of 12 fps when running on the NJN.

The pre-existing trained weights were used as part of the implementation of YOLOv5 variants to determine the accuracies. Figure 4 shows YOLOv5x has the best accuracy while the YOLOv5n has the worst accuracy when tested against the COCO2017 dataset.

The average inference speed is 1 fps for the YOLOv5x. The performance of YOLOv5x on NJN is substantially lower than the other 4 variants. Nevertheless, the YOLOv5x achieves an accuracy of 70% when tested against the COCO2007 dataset. The goal of this work is to measure the power consumption and not the issues related to the accuracy of the YOLOv5 variants. YOLOv5x is a suitable variant to use when considering high detection accuracies without the need for real-time performance. The number of parametes per YOLOv5 varant is depicted in Figure 5.

[2]Available online, https://github.com/ultralytics/YOLOv5, last accessed 01/07/2022

[3]Available online, https://pytorch.org/, last accessed 01/07/2022

[4]Available online, https://ultralytics.com/, last accessed 01/07/2022

[5]Available online, https://store.sundance.com/product/lynsyn-lite/, last accessed 01/07/2022

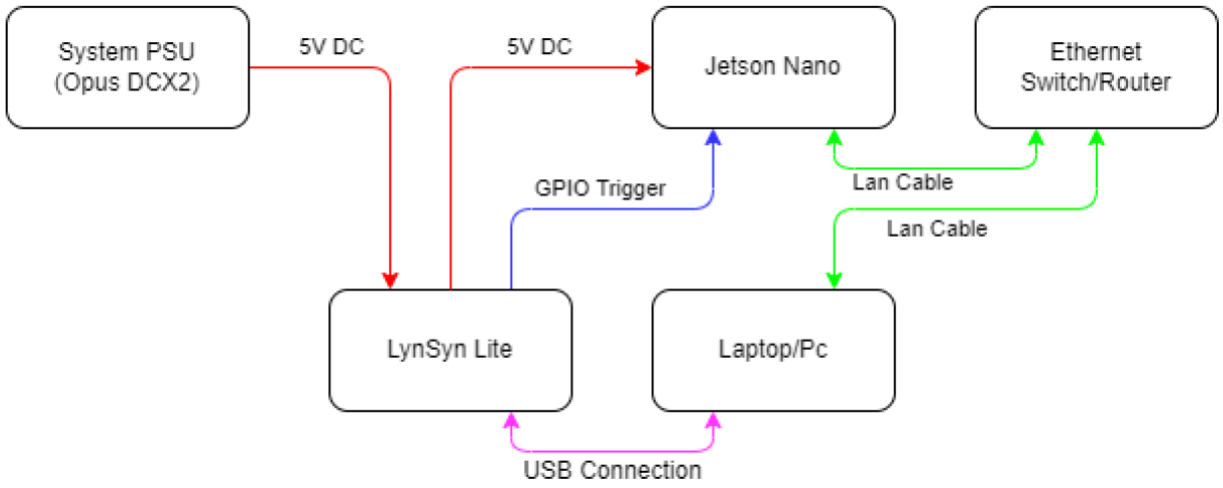


Fig. 1. Hardware setup. The power supply is wired to NJN through the power rails in the LSLB. The LSLB is connected to the NJN via the GPIO headers of both boards. A laptop is used to collect the power measurements from the LSLB using the USB port and from NJN through the ethernet

### B. Jetson Nano in CPU Only Mode

The NJN features a Quad-core ARM Cortex-A57 processor onboard, so setting the YOLOv5 network to run inference on the CPU allows the acquisition of the performance expected. Figure 6 depicts the AI inference performance and that combining the CPU and GPU devices (i.e. NJN has a throughput of 12 fps for YOLOv5n) outperforming it by a factor of 30 CPU inference - NJN (CPU) has a throughput of 0.4 fps for YOLOv5n.

### C. Power Consumption

Milliwatt-hour per frame (mWh/frame) is the measurement unit for power consumption per each processed image frame. This value is calculated according to Equation 2, where $P_{frame}$ is the consumed power per image frame, $t_s$ the sampling period in seconds, $P_{avg}$ the average power during measurement, $n_{frames}$ the number of frames and constant $K_c$ = 3600 . 1000 used to convert from mWs to Wh.

$$P_{frame} = \frac{t_s.P_{avg}}{n_{frames}.K_c} \quad (2)$$

Lower mWh/frame values mean less power is required to perform inference per image frame. The results show that YOLOv5x required 2.5 mWh/frame while the smallest YOLOv5n only required 0.15 mWh/frame to perform the same tasks, meaning that YOLOv5n is approximately 16 times more power efficient while running on the NJN (see Figure 7).

The inference using the gls*cpu vs. CPU and GPU reduced the power consumption from 2.18 mWh/frame to 0.15 mWh/frame which represents a reduction of 2.03 mWh/frame for the YOLOv5n. This is the benefit of combining heterogeneous platforms at the edge (see Figure 8).

The power performance is a crucial factor that should be taken into consideration if the system is to be powered by a battery power source.

The overall results of the power consumption are listed in Table I

TABLE I
OVERALL RESULTS

| Model | Dataset | Measuring device | Device(s) under test | fps | Average Power Consumption (APC) [mWh/frame] |
|---|---|---|---|---|---|
| YOLOv5n | COCO2017 | LSLB | CPU + GPU | 11.9 | 159 |
| YOLOv5n | COCO2017 | LSLB | CPU only | 0.4 | 2183 |
| YOLOv5n | COCO2017 | NJN Power Sensors | CPU + GPU | 11.7 | 161 |
| YOLOv5n | Test video | NJN Power Sensors | CPU + GPU | 13.5 | 143 |
| YOLOv5s | COCO2017 | LSLB | CPU + GPU | 5.9 | 354 |
| YOLOv5s | COCO2017 | LSLB | CPU only | 0.1 | 6610 |
| YOLOv5s | COCO2017 | NJN Power Sensors | CPU + GPU | 5.8 | 363 |
| YOLOv5s | Test video | NJN Power Sensors | CPU + GPU | 6.5 | 328 |
| YOLOv5m | COCO2017 | LSLB | CPU + GPU | 2.7 | 804 |
| YOLOv5m | COCO2017 | NJN Power Sensors | CPU + GPU | 2.7 | 822 |
| YOLOv5m | Test Video | NJN Power Sensors | CPU + GPU | 2.9 | 754 |
| YOLOv5l | COCO2017 | LSLB | CPU + GPU | 1.6 | 1474 |
| YOLOv5l | COCO2017 | NJN Power Sensors | CPU + GPU | 1.5 | 1501 |
| YOLOv5l | Test Video | NJN Power Sensors | CPU + GPU | 1.7 | 1364 |
| YOLOv5x | COCO2017 | LSLB | CPU + GPU | 0.9 | 2562 |
| YOLOv5x | COCO2017 | NJN Power Sensors | CPU + GPU | 0.9 | 2580 |
| YOLOv5x | Test Video | NJN Power Sensors | CPU + GPU | 1.0 | 2452 |

### D. Comparing edge Computing With a Cloud Solution

Motivated by the low performance of the NJN while running bigger YOLOv5 variant networks sizes such as YOLOv5x the Azure CV service was utilised to evaluate the potential performance of the NJN sending the frames to the cloud for processing. A batch of measurements used an Azure S1 cloud instance to ensure that there were no bottlenecks in the cloud. The latter is capable of processing 10 frames per second, and the chosen location in the geographically closest region at the time of writing was the UK-SOUTH. The cloud performance is limited mostly by network throughput, achieving only roughly 2 fps. This performance is equivalent to locally running YOLOv5m and YOLOv5l.

Similarly, the power consumption is also equivalent to that of locally running YOLOv5m and YOLOv5l. The APC needed for sending one frame and obtaining the inference results from the cloud is around 2.5 mWh/frame.

## VI. Conclusions and Future Work

The subject covered in this paper has several impacts on ethical intelligent decision-making. By analysing the power

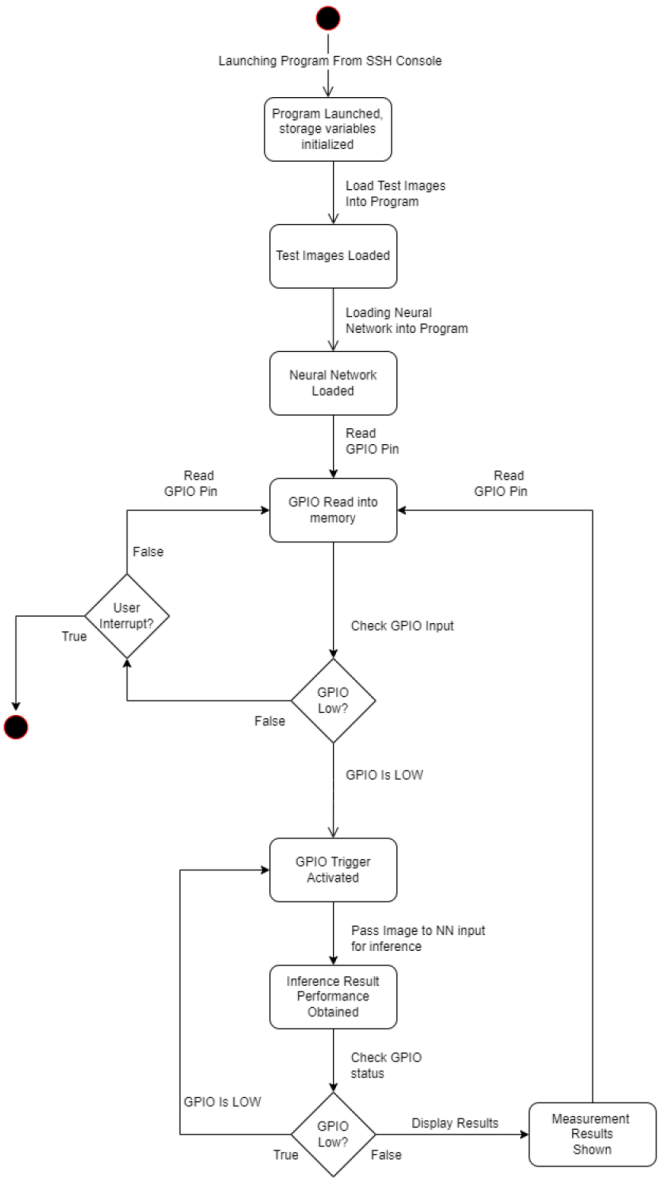


Fig. 2. PBP flow diagram.

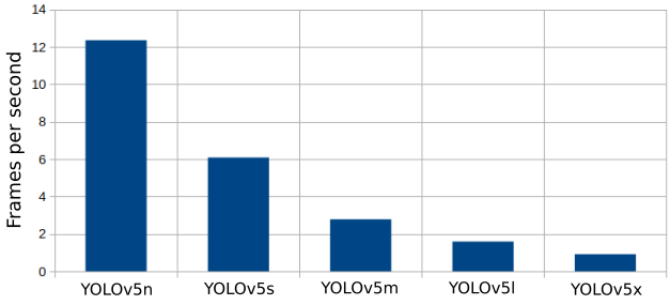


Fig. 3. Performance of various sized YOLOv5 models running on the NJN

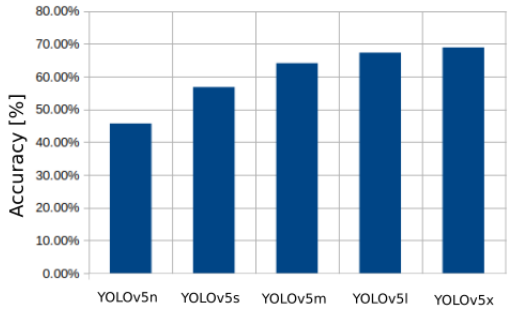


Fig. 4. Accuracies of the YOLOv5 variants

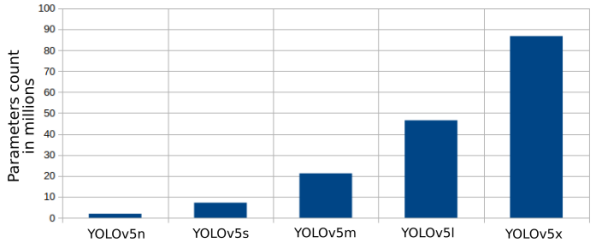


Fig. 5. Number of parameters per YOLOv5 variants

consumption and data throughput of the NJN when running different variants of the YOLOv5 model, the authors provide valuable insights into the energy efficiency and performance trade-offs in AI inference at the edge. This information is essential for making informed decisions regarding the deployment of AI systems in resource-constrained environments. Ethical decision-making in this context involves considering the environmental impact and sustainability of AI systems, ensuring they operate efficiently without unnecessarily depleting resources.

Moreover, the study demonstrates that combining both CPU and GPU resources on the NJN outperforms using the CPU alone for YOLOv5 inference. This finding highlights the

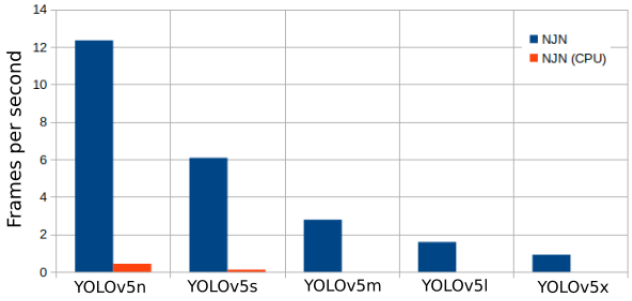


Fig. 6. YOLOv5 Inference Performance Comparison between CPU and GPU mode on Jetson Nano

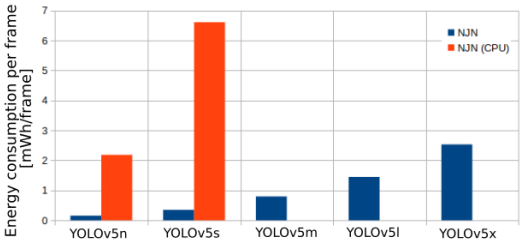


Fig. 7. Power consumption per processed frame in mWh/frame

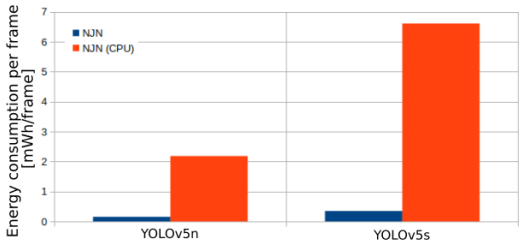


Fig. 8. Power consumption per processed frame in mWh/frame (YOLOv5n and YOLOv5s)

importance of heterogeneity in AI systems for achieving optimal performance. Ethical intelligent decision-making involves considering the most efficient allocation of computational resources to minimising energy consumption and maximising performance while adhering to any constraints or limitations.

From a business perspective, the project opens up future exploitable application areas. The authors show that the NJN offers lower power consumption and higher data throughput compared to combining the CPU performance with cloud services like Azure. This finding suggests that the NJN can be a cost-effective solution for AI inference at the edge, where real-time processing and low latency are crucial. Businesses can leverage the NJN to develop AI applications that require on-device processing, such as autonomous vehicles, surveillance systems, industrial automation, and robotics.

Furthermore, the authors highlight the ease of use and AI acceleration capabilities of the NJN and similar NVIDIA boards, thanks to full CUDA support. This accessibility and compatibility with popular frameworks and libraries like TensorFlow, Darknet, and PyTorch make it easier for businesses to adopt and integrate AI technologies into their existing workflows. It creates a thriving community with abundant knowledge, tips, and advice, facilitating the development and deployment of AI applications.

For future work, the authors will expand this research to other DL algorithms suitable to be used in edge devices. The authors will also explore the use of Machine Learning (ML) for estimation of power consumption and reducing power consumption in real-time. Finally, it is also intended to extend this work by performing the same tests using the same methodology on MPSoC, ACAP and other commercial-off-the-shelf heterogeneous platforms.

## References


[1] H. S. de Andrade, "Software concerns for execution on heterogeneous platforms," Ph.D. dissertation, Chalmers University of Technology, 2018.

[2] A. Nordrum, "The internet of fewer things [news]," *IEEE Spectrum*, vol. 53, no. 10, pp. 12–13, 2016.

[3] W. Wolf, "Multiprocessor system-on-chip technology," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 50–54, 2009.

[4] B. Gaide, D. Gaitonde, C. Ravishankar, and T. Bauer, "Xilinx adaptive compute acceleration platform: Versaltm architecture," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2019, pp. 84–93.

[5] N. Hassan, S. Gillani, E. Ahmed, I. Yaqoob, and M. Imran, "The role of edge computing in internet of things," *IEEE communications magazine*, vol. 56, no. 11, pp. 110–115, 2018.

[6] A. Ghosh, D. Chakraborty, and A. Law, "Artificial intelligence in internet of things," *CAAI Transactions on Intelligence Technology*, vol. 3, no. 4, pp. 208–218, 2018.

[7] N. D. Lane, S. Bhattacharya, A. Mathur, P. Georgiev, C. Forlivesi, and F. Kawsar, "Squeezing deep learning into mobile and embedded devices," *IEEE Pervasive Computing*, vol. 16, no. 3, pp. 82–88, 2017.

[8] Y. Deng, "Deep learning on mobile devices: a review," in *Mobile Multimedia/Image Processing, Security, and Applications 2019*, vol. 10993. SPIE, 2019, pp. 52–66.

[9] Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, and Q. Zhang, "Deep learning on mobile and embedded devices: State-of-the-art, challenges, and future directions," *ACM Computing Surveys (CSUR)*, vol. 53, no. 4, pp. 1–37, 2020.

[10] A. Nasif, Z. A. Othman, and N. S. Sani, "The deep learning solutions on lossless compression methods for alleviating data load on iot nodes in smart cities," *Sensors*, vol. 21, no. 12, p. 4223, 2021.

[11] S. Han, H. Mao, and W. Dally, "Compressing deep neural networks with pruning, trained quantization and huffman coding. arxiv 2015," *arXiv preprint arXiv:1510.00149*.

[12] X. Liu, J. Pool, S. Han, and W. J. Dally, "Efficient sparse-winograd convolutional neural networks," *arXiv preprint arXiv:1802.06367*, 2018.

[13] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song *et al.*, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA international symposium on field-programmable gate arrays*, 2016, pp. 26–35.

[14] R. Zhao, W. Song, W. Zhang, T. Xing, J.-H. Lin, M. Srivastava, R. Gupta, and Z. Zhang, "Accelerating binarized convolutional neural networks with software-programmable fpgas," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 15–24.

[15] M. J. Shafiee, B. Chywl, F. Li, and A. Wong, "Fast yolo: A fast you only look once system for real-time embedded object detection in video," *arXiv preprint arXiv:1709.05943*, 2017.

[16] G. Yang, W. Feng, J. Jin, Q. Lei, X. Li, G. Gui, and W. Wang, "Face mask recognition system with yolov5 based on image recognition," in *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, 2020, pp. 1398–1404.

[17] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, J. Fang, imyhxy, K. Michael, Lorna, A. V, D. Montes, J. Nadar, Laughing, tkianai, yxNONG, P. Skalski, Z. Wang, A. Hogan, C. Fati, L. Mammana, AlexWang1900, D. Patel, D. Yiwei, F. You, J. Hajek, L. Diaconu, and M. T. Minh, "ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference," Feb. 2022, https://doi.org/10.5281/zenodo.6222936. [Online]. Available: https://doi.org/10.5281/zenodo.6222936

[18] Z. Yu, P. Machado, A. Zahid, A. M. Abdulghani, K. Dashtipour, H. Heidari, M. A. Imran, and Q. H. Abbasi, "Energy and performance trade-off optimization in heterogeneous computing via reinforcement learning," *Electronics*, vol. 9, no. 11, p. 1812, nov 2020.

[19] A. D. Helfrick and W. D. Cooper, *Modern electronic instrumentation and measurement techniques*. Prentice Hall Englewood Cliffs, NJ, 1990.