CSC8499 Individual Project: Benchmarking SOTA Deep Learning Models on Jetson nano

Xinquan, Li

MSc in Advanced Computer Science,

School of Computing Science, Newcastle University.

b9033698@newcastle.ac.uk

Abstract. Edge computing play an import role in Internet of Things. One of most difficult task is image processing[2]. Deep learning techniques have made great success in the field of computer vision. The report run the object detection tasks on Jetson nano. The SOTA deep leaning models will read the raw images for inference and run on VOC validation dataset. During the process, the report will compare the influence of hybridised programming, and the total inference time of cpu and GPU. When the SOTA deep learning models run VOC validation dataset, the information of system performance, such as the change of GPU temperature and swapfile, and performance of object detection models will be collected. This will help users better understand the capability of Jetson nano and compare the models when deploying.

1. Introduction

The modern society are generating more and more data, such as photos, audios and videos. The types of data are diverse. And the source of the data is also different. Not only the data from individuals, but also from devices, sensors and systems. For example, smart cars, sensors in smart devices and traffic control system.

There is a predict of International Data Corporation, the sum of the data in the world will grow from 33 zettabytes(2018) to 175(2025), and data real-time processing will be a significant part[9]. When processing the data, first data will be saved in datacenter, then data will send to clouds. At last, the software engineers and data scientists will analyse the data by using some techniques, such as machine learning and deep

learning methods. But this kind of solution is not suitable for every tasks, because the capacity and speed of networks are limited sometimes. In this situation, the real-time tasks will fail because of the delay of networks.

Edge computing is a distributed computing architecture. It moves the computing of application, data and services from the central node of the network to the logical edge node of the network for processing the data[5]. The central nodes are usually the clouds, the edge nodes are usually edge computing devices, such as Jetson nano, Raspberry and Coral. Edge nodes are closer to user terminal devices, which can speed up data processing and reduce delays. Under this architecture, the analysis of the data are closer to the source of data, so it is more suitable for processing data. In this article, the tasks will run on Jetson nano. Jetson nano is small edge device, it can run embedded and AI applications.

The most difficult tasks in edge computing architecture is dealing with audio, video, and images. But deep learning technology is successful in overcoming these difficulties. Deep learning is a branch of machine learning. It is an algorithm that attempts to use multiple processing layers consisting of complex structures or multiple nonlinear transformations to abstract data at a high level. But the edge computing devices is not very powerful, the limitation of edge computing devices in memory and calculation speed is obvious. So it is import to find a balance between deep learning models and edge computing devices.

In order to help make decisions when deploy SOTA deep learning models on edge computing devices, this article benchmark object detection models on Jetson nano. In order to get direct impression of Jetson nano and object detection models, the information of system performance and model evaluation will be collected. For example, GPU and memory data when running the models on Jetson nano will be collected for system performance. And throughput, accuracy and mAP will be collected for model evaluation.

The background and basic concepts of deep learning, object detection and Jetson nano will be discussed in section 2. The method of benchmarking will be discussed in section 3. The results will show in section 4. The aim and objectives will show in section 2.1.

2. Background

2.1. Objectives

The aim of the report is to benchmark SOTA deep learning models on edge computing device(Jetson nano). It will run some object detection tasks on Jetson nano, which will help better understand theses SOTA deep learning models and edge computing device.

Table 1. The specific objectives of the report

List	Specific Objectives
1: Explore	Explore basic concepts of deep learning and the edge computing devices(Jetson nano)
2: Design and Apply	Design object detection tasks and run the tasks on Jetson nano
3: Benchmarking	Collect and analysis the information of object detection models and Jetson nano performance.

2.2. Artificial Intelligence and Deep Learning

The term artificial intelligence has existed a long time. But It has been talked about by the public in recent years. This is closely related to the rise of technologies such as machine learning and deep learning.

Machine learning is a subject that discusses various functional forms suitable for different problems and how to use data effectively obtain the specific values of function parameters. Deep learning is a branch of machine learning, deep learning is a class of functions in machine learning. The form of deep learning is usually a multilayer neural networks.

The neural network model and the core idea of programming with data have been studied for hundreds years. The computation theory of Alan Turing have a profound impact on machine learning. In the famous paper “Computer Machines and Intelligence”[2], Turing raised the a famous question “ Can machine think?” and described “ Turing test”. “ Turing test” is a person cannot distinguish whether his conversation object is a human or a machine when have a text interaction.

Nowadays machine learning has get into every aspects of work and life, because machine learning make breakthroughs in many issues that thought to be unsolved. Deep learning play a significant part in this process and it gradually get evolved into a universal tool for engineers and scientists.

2.3. Neural Networks

The research of neural network can be traced more than a century ago. Researchers tried to build computational circuits that mimic the interaction of neurones. Nowadays , the biological explanation of neural networks was replaced, but the name for Neural networks was still kept. The report will show two simple single layer neural network as introduction.

Linear regression model try to learn a function that predicts through linear combination of attributes. The function is:

$$f(x) = w^T x + b \tag{1}$$

The linear model is simple in form, and many more powerful nonlinear models can be obtained by introducing hierarchical structure or high dimensional mapping on the basis of linear models. In addition, w directly express the importance of each attribute in prediction.

In deep learning, the neural network diagrams can intuitively express the model structure. In order to show the structure of linear regression model, Figure 1 will use a neural network diagram to represent the linear regression model. The neural network diagram hide the model parameters weights and deviations. In linear regression model, the output layer are completely connected to the inputs layer. Therefore, the output layer is also called fully connected layer or dense layer.

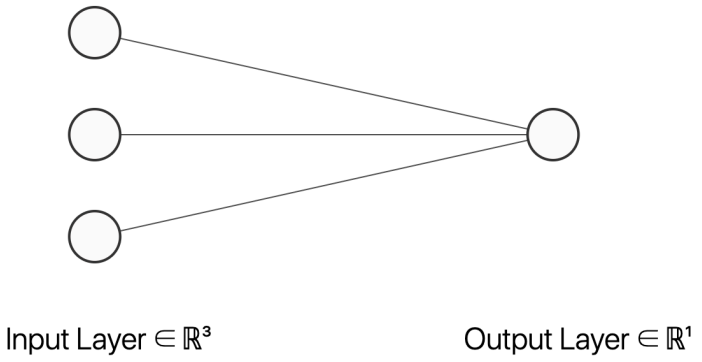

Figure 1. Linear regression model, single layer neural network

Another single layer neural network is SoftMax regression. Softmax regression is suitable for the scenario that the outputs of the model are discrete values. For example, image category. So for such discrete value prediction problems, the classification models such as softmax regression will be used. As comparison, unlike the linear regression model, the output unit of softmax regression model has changed from one to multiple. Figure 2 is softmax regression model. The number of outputs in

softmax regression model is equal to the number of categories in classification problem.

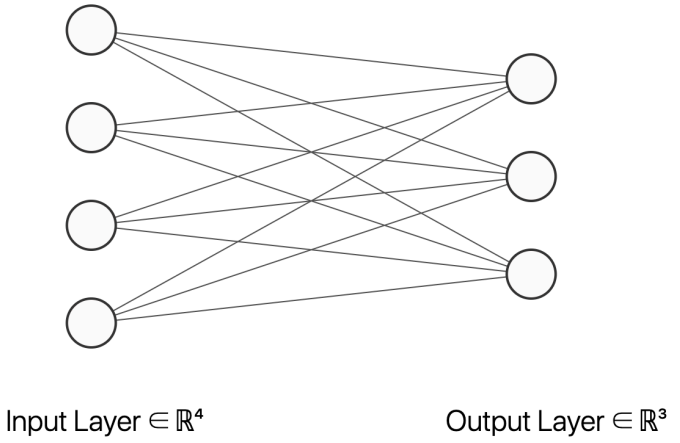

Figure 2. Softmax regression model, single layer neural network

Softmax regression is suitable for classification problems, beacaues it uses softmax to calculate the probability distribution of the output. There is the formula of softmax.

$$softmax(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{2}$$

2.3.1. Multilayer Perceptron

The multilayer perceptron add one or more hidden layer on the basis of a single layer neural network. The hidden layer is located between input layer and output layer. Figure 3 shows a multilayer perceptron.

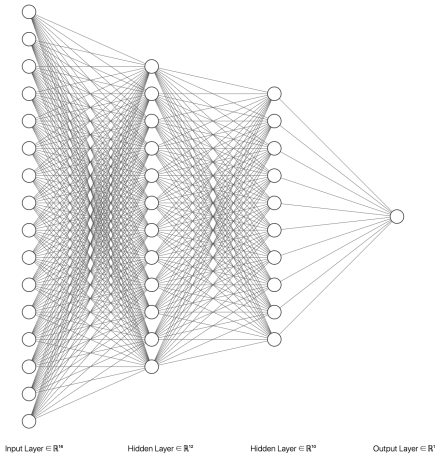

Figure 3. Multilayer Perceptron Model

2.3.2. Activation Functions

Although the neural network add hidden layers, but actually it is still equal to a single layer neural network. The problem is that the fully connected layer only performs affine transformation on the data. One way to solve the problem is to introduce a nonlinear transformation, such as transforming hidden variables using nonlinear functions and use them as the input of next fully connected layer. This nonlinear function is called the activation function. There are some common activation functions.

Rectified linear function(ReLU) is a simple nonlinear transformation. Given variable x, the function is defined as :

$$ReLU(x) = max(x,0). \tag{3}$$

It can be seen that the ReLu function only keep positive elements and clears negative elements.

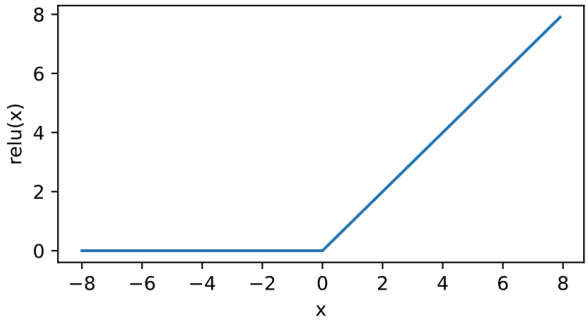

Figure 4. ReLU Function diagram, activation function.

The sigmoid function was more common in neural networks a few years ago, but it is gradually replaced by the ReLU function. The sigmoid function can transform the value of element to between 0 and 1. There is the sigmoid function:

$$sigmoid(x) = 1/(1 + exp(-x)). \tag{4}$$

The sigmoid function is draw below. When the input is close to 0, the sigmoid function is close to linear transformation.

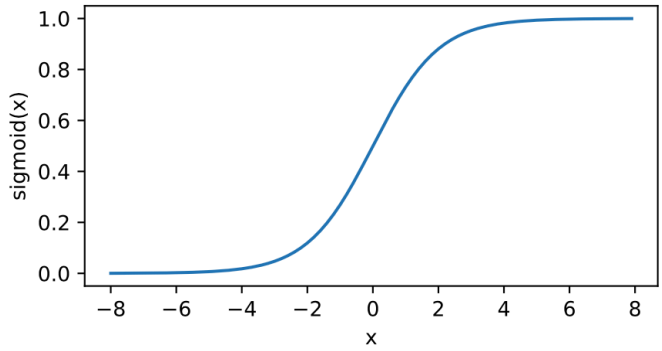

Figure 5. Sigmoid Function diagram, activation function.

The *tanh* function can transform the value of an element between -1 and 1.There is the *tanh* function:

$$tanh(x) = (1 - exp(-2x))/(1 + exp(-2x)). \tag{5}$$

When the input is close to 0, the *tanh* is close to 0, the *tanh* is close to linear transformation. Although the shape of the function is very similar the shape of the sigmoid function, the *tanh* is symmetric at the origin of the coordinate system.

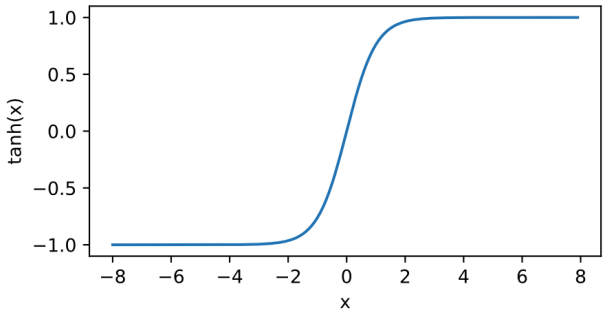

Figure 6. Tanh Function diagram, activation function.

Commonly used activation functions include ReLu, sigmoid and tanh function. Sigmoid is suitable for binary classification tasks and rarely used in other situation. In most cases, the ReLU function is the default activation function.

2.4. CNN

Convolutional neural networks(CNN) is the star network architecture in deep learning neural network. Convolutional neural network is cornerstone in the filed of computer vision in recent years. A standard convolutional network architecture is mainly composed of core layers such as convolutional layer, pooling layer and fully connected layer.

The report will describe the working principle of convolutional layer, pooling layer and fully connected layer, and explain the meaning of padding, stride, input channel and output channel.

The main function of the convolutional layer is to extract features of the input data, and the convolution kernel in convolutional layer will complete this function. The convolutional kernel is regarded as a scanner with a specified a window size. The scanner scans the input data again and again to extract the features. If the input is image data, after processing by the convolutional kernel, the import features of the image can be identified.

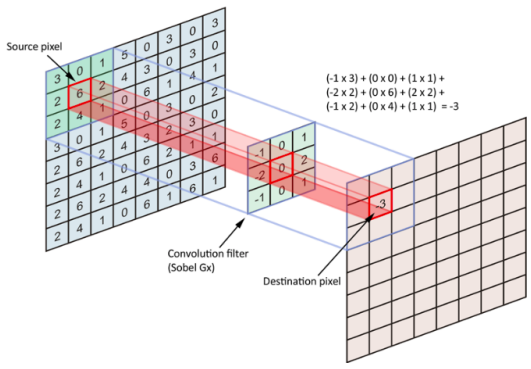

Figure 7. Convolutional Layer, the kernel extract the features of the input data.

There are two important hyper-parameters: padding and stride. The two hyper-parameters can change the output shape(given input shape and convolutional layer).Padding can increase the width and height of the output. This is often used to make the output have same width and height as the input. The stride can reduce the height and width. Given an input image, the convolutional layer with different kernels can perform many operations. For example, edge detection and blur[1].

In practical, input images with only one colour channel are rare. Because the real data have higher dimension. Suppose there are RGB three colours channel. The three-channel convolution process can be regarded as three independent single-channel

convolution processes. The three independent single-channel convolution processes are added to get the final output.

The pooling layer in the convolutional neural network can be regarded as the a method that extract the core features of the input data in the convolutional neural network. The pooling layer not only achieves the compression of data, but also has a significant impact on reducing the parameters involved in model calculation. So pooling will improve the calculation efficiency.

The most commonly used pooling layer methods are average pooling layer and maximum pooling layer.

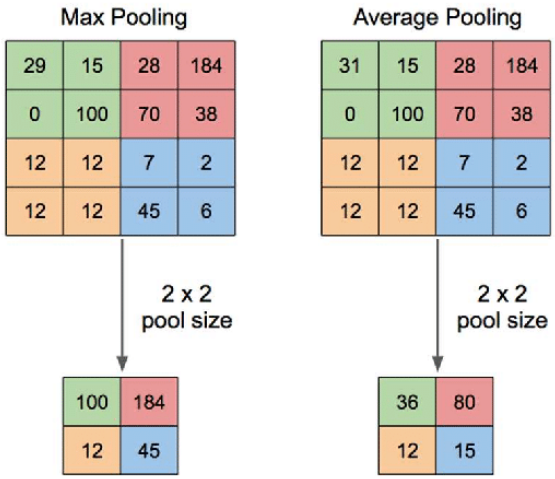

Figure 8. Pooling layer methods, Max pooling and average pooling.

The main function of the fully connected layer is to compress the features that extracted from the input image after the convolutional layer and pooling layer. According to the compressed features the pooling layer will complete the classification function of the model.

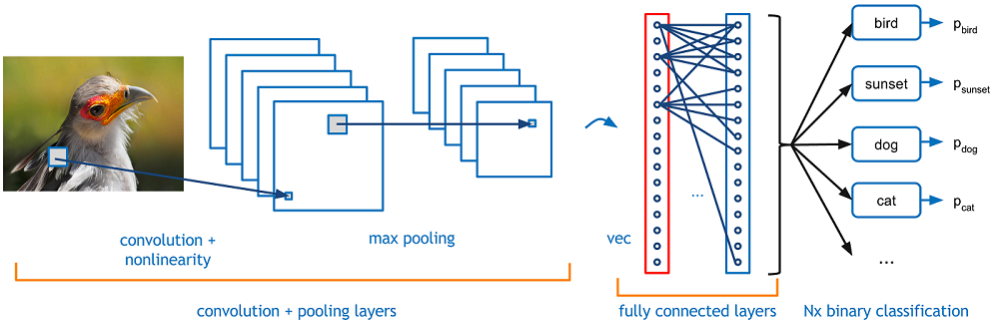

Figure 9. CNN architecture, the fully connected layer will complete classification function.

In image classification, the input refers to the input of image, and the output is the possible category or classification to be recognised by the training model. Users can select 1000 different objects or just two. After training, the model can process real-time data and provide real-time results. This process is called inference.

Deep learning relies on convolutional neural network (CNN) models to convert images into predictive classification. CNN is a type of artificial neural network. It uses convolutional layers to select useful information from the input and is the preferred network for image processing.

Convolutional neural networks are networks with convolutional layers. There are some famous CNN models:

• LeNet, an early neural network used to recognise handwritten digital images[8]. LeNet demonstrated that training a neural network through gradient descent can achieve the most advanced results of handwritten digit recognition at that time. This is first time that CNN is known to the world.

• AlexNet, the name of the model comes from the author of the paper. AlexNet uses 8-layer convolutional layer and won the ImageNet 2012 Image Recognition Challenge with a great advantage. It proved for the first time that the learned features can surpass the manually designed features, thus breaking the previous state of computer vision research[11]. There are some new features of the model, for example Alex changed the sigmoid activation function to ReLU activation function.

• VGG model proposed the idea of building a deep model by reusing simple basic blocks[21].

2.5. Object Detection

How to get the information from the image that can be understood by the computer is the most important problem in the field of computer vision. Deep learning techniques become a hot direction in computer vision because their powerful capabilities.

Object detection is a way to understand an image. In the image classification task, assume that there is only one subject target in the image and pay attention to how to identify the target category. However, many times there are multiple objects of interest in the image, and researchers not only want to know their categories, but also their specific positions in the image. In computer vision, this type of task is called

object detection. Object detection is widely used in many fields, for example, robots or sensors use object detection task to detect objects of interest.

2.5.1. Bounding Box and Anchor Box

Object detection is the problem of two steps. First, the object detection model should locate the object or several objects in the image. Then it will give a prediction of the classes of objects. In object detection tasks, researchers usually use a bounding box to describe the object location. The bounding is a rectangular box, which can be defined by the upper left corner of rectangle (*x, y*) and the lower right corner(*x, y*).

Figure 10. The bounding box in object detection.

Object detection algorithms usually sample a large number of regions in the input image, then it determine wether these regions contain the objects of interests. And it will adjust the edge of the region, so it is more close to the ground-truth bounding box of the targets. Different may use different regional sampling methods, the report will introduce three object detection models in next section(SSD, faster-RCNN, yolo-v3) for benchmarking.

Anchor box is that object detection model generate multiple bounding boxes with different size and ratio based on each pixel of the input image. In MXNET deep learning frame, the method of generating the anchor box has been implemented in the MultiBoxPrior function. According to the input, sizes and ratios, the function will return all anchor boxes of the input image. Here is the api link:*http://mxnet.incubator.apache.org/versions/1.1.0/api/python/symbol/contrib.html*.

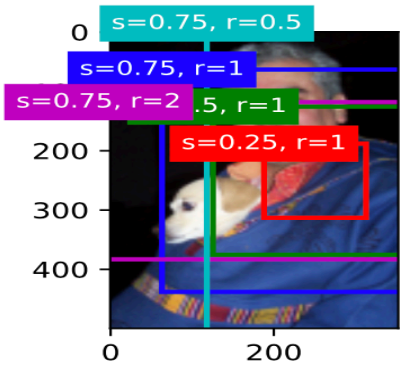

Figure 11. Anchor Box in Object detection

2.5.2. IoU

IoU(intersection over union) is a method to measure the similarity between the anchor box and the ground-truth bounding box. The meaning of IoU is the ratio of the intersection area of two bounding boxes to the combined area. The value range of IoU is between 0 and 1, 0 means that the two bounding boxes have no overlapping pixels, 1 means that the two bounding boxes are equal.

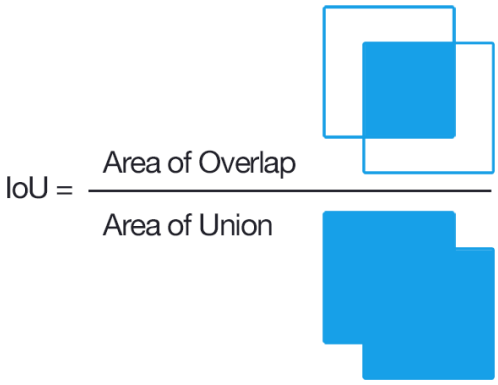

Figure 12. IoU, intersection over union. measured the similarity of bounding boxes

In the model prediction stage, the model will generate the prediction of category and offset of anchor boxes. When the number of anchor boxes is too large, NMS(non-maximum suppression) can be used to remove similar predicted bounding boxes. In MXNET, the MultiBoxDetection perform the non-maximum suppression.

2.5.3. mAP

The object detection model can be described in the framework of statistical estimation. Researchers pay attention to the probability of making the first type of error and the second type of error, which are usually described by accuracy and recall. The accuracy rate describes how accurate the model is, that is, how many of the results that are predicted to be positive examples are real cases; the recall rate describes how complete the model is, that is, how many of the true samples are predicted by the model Is a positive example. Different tasks have different preferences for the two types of errors, and often try to reduce another type of error when the error of some type is not more than a certain threshold. In object detection, mAP (Mean Average Precision) takes these two errors into consideration as a unified indicator.

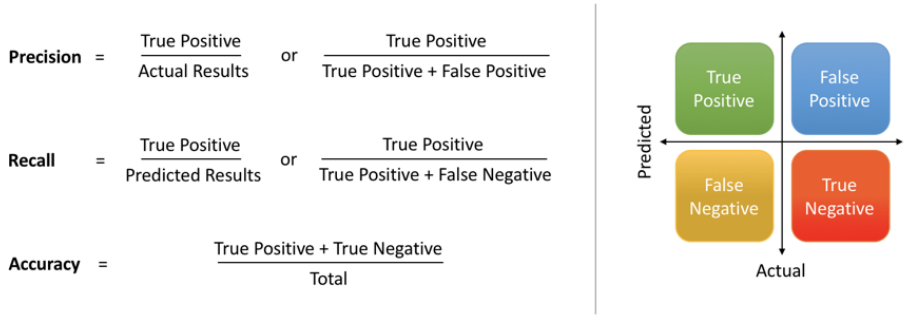

Figure 13. Recall, precision and Accuracy

Specifically, for each picture, the object detection model gives multiple prediction frames (often far more than the number of real frames). IoU (Intersection Over Union) is to mark whether the prediction frame is correct. After the marking is completed, as the prediction frame increases, the recall rate will always increase, and the accuracy rate will be averaged under different recall rate levels to get the AP, and finally all categories will be averaged according to their proportions to get the mAP .

2.5.4. Multiscale Object Detection

Anchor boxes are generated with each pixel of the input image. These anchor boxes are samples of different areas of the input image. However, if the anchor frame is generated with each pixel of the image, it is easy to generate too many anchor frames and cause too much calculation in training and inference. A simple method is

to uniformly sample a small part of the pixels in the input image and generate an anchor frame with the sampled pixels. In addition, under different scales, the object detection model can generate different numbers and different sizes of anchor boxes.

The output of a two-dimensional array of convolutional neural networks is called the feature map. According the shape of the feature map, researchers can determine the uniformly sampled anchor box centre on the image.

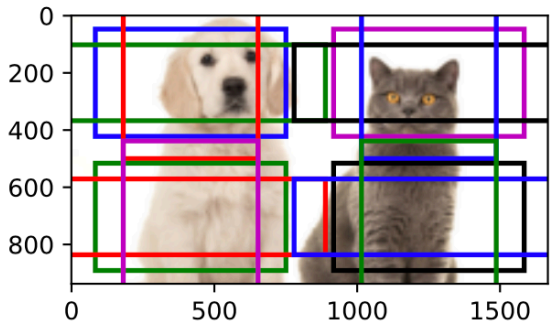

Figure 14. Uniformly sampled on the image, MultiScale object detection.

When feature maps of different layers have different receptive fields on the input image, this will be used to detect targets of different sizes. That is a basic idea of SSD(One stage object detection model)[20].

2.6. Jetson Nano

Jetson nano is a small embedded computer devices made by NVDIA, and Jetson nano is also a development board for deep learning or computer vision. The development kit of Jetson nano allows hobbyists and programmers have the opportunity to play deep learning and neural networks at an affordable price.

Any application related to smart computing, including smart cities, smart industry, smart driving[14], Internet of Things and remote medical analysis[3], they are all closely related to data, algorithms, and computing performance. With the participation of more application scenarios, people hope to implant more applications into a device that is smaller in size, lower in power consumption, more suitable for cost, and at the same time can meet the requirements of a complete artificial intelligence system. The data of higher resolution sensors, the inference function of running multiple neural networks, supports continuous software upgrades, and can add and update AI network and framework support at will.Jetson nano is a highly versatile embedded high-

performance device, suitable for most application scenarios, and based on NVIDIA's CUDA unified architecture.

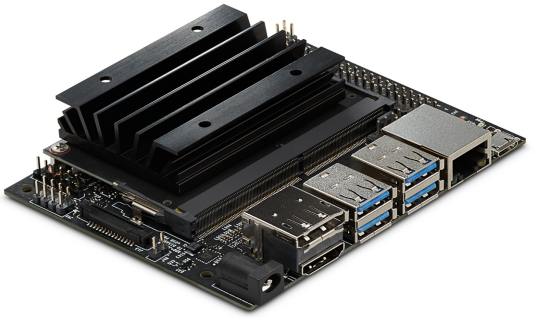

Figure 15. Jetson nano Developer Kit

Jetson nano is equipped with quad-core Cortex-A57 MPCore processor and 128-core Maxwell™ GPU. Support JetPack SDK. Support mainstream AI frameworks and algorithms, such as TensorFlow, PyTorch, Caffe/Caffe2, Keras, MXNet, etc. Supports applications such as face recognition, object recognition tracking, object detection and positioning.

GPU	128-core Maxwell
CPU	Quad-core ARM A57 @ 1.43 GHz
Memory	4 GB 64-bit LPDDR4 25.6 GB/s
Storage	microSD (not included)
Video Encode	4K @ 30 | 4x 1080p @ 30 | 9x 720p @ 30 (H.264/H.265)
Video Decode	4K @ 60 | 2x 4K @ 30 | 8x 1080p @ 30 | 18x 720p @ 30 (H.264/H.265)
Camera	2x MIPI CSI-2 DPHY lanes
Connectivity	Gigabit Ethernet, M.2 Key E
Display	HDMI and display port
USB	4x USB 3.0, USB 2.0 Micro-B
Others	GPIO, I^2C, I^2S, SPI, UART
Mechanical	69 mm x 45 mm, 260-pin edge connector

Figure 16. Jetson nano Hardware

Jetson platform is supported by JetPack SDK, including board support package (BSP), Linux operating system, NVIDIA CUDA, and compatible with third-party platforms. JetPack can provide libraries for deep learning, computer vision, accelerated computing and multimedia, and supports drivers for various sensors. It provides detailed information about application and system power and performance, and therefore helps developers quickly optimise and adjust the code.

There is a tutorial from NAVIDA, which will help new developers quickly start[7]. This tutorial will give an introduction about Jetson nano, and setup the devices and download the Jetson nano developer kit SD card Image. After logging, users can check the installed system components.

Table 2. Jetson Nano developer kit Pre-installed system components and locations

System components	Location
TensorRT	/usr/src/tensorrt/samples/
CUDA	/usr/local/cuda-/samples/
cuDNN	/usr/src/cudnn_samples_v7/
Multimedia API	/usr/src/tegra_multimedia_api/
VisionWorks	/usr/share/visionworks/sources/samples/ /usr/share/visionworks-tracking/sources/samples/ /usr/share/visionworks-sfm/sources/samples/
OpenCV	/usr/share/OpenCV/samples/

NVP model is to set different clocks to achieve different power consumption modes of the chip. NVIDIA Jetson nano platform has a set of custom modes of operation, but default mode of operation is MAXN.All the tasks in this report will run under MAXN mode.

2.7. Experimental Setup

JetPack SDK is a good tool when developers want to build their deep learning applications or other AI applications. The Jetpack SDK is the solution made by NAVIDA. But it is a confusing task to run theses applications sometimes. There are

many Jetpack SDK versions, sometimes the deep learning frame that the users want to use is not compatible with Jetpack SDK because of versions. For example, there is a version of Jetpack SDK(NAVIDA DLI Jetson nano SDK) that pre-installed PyTorch and Tensorflow, but when users want to install MXNET by the wheel(provided by NAVIDA). Users may fail sometimes, because the version of JetPack SDK is not supported by the MXNET wheel that provided by NAVIDA. So it is necessary to have an introduction about experimental setup of the project.

2.7.1. MXNET

MXNet is a deep learning framework designed to improve efficiency and flexibility. It allows you to mix symbolic and hybridise programming to maximise efficiency and productivity. The core of MXNet is a dynamically dependent scheduler that can dynamically and automatically parallelise symbol and command operations. The most important graphics optimisation layer makes symbol execution faster and memory efficient. MXNet is portable and lightweight, and can be effectively extended to multiple GPUs and multiple machines.

The project install MXNET v1.6 and the dependencies for it on Jetson nano. Jetpack4.4 is necessarily needed. It can download from the tutorial(Getting started with Jetson nano)[7]. There are some necessary steps to install MXNET on Jetson nano:

- Increase the swapfile. It is necessary to increase the swapfile in order to get more memory on Jetson nano. Without enough memory the inference task may be killed. Here is a good tool to increase the swapfiles, *https://github.com/JetsonHacksNano/installSwapfile.git*[13]. Use this tool, it can increase the swapfile on Jetson nano.

- Install MXNET dependencies. These command lines will be used:
  sudo apt-get update
  sudo apt-get install -y git build-essential libopenblas-dev python3-pip
  sudo pip3 install -U pip

- Install MXNET v1.6. There are several ways to install MXNET on Jetson nano. One is to download from source code and setup. Another is highly recommended, download and install MXNET wheel with NAVIDA support. Here are command lines that will be used:
  wget https://mxnet-public.s3.us-east-2.amazonaws.com/install/jetson/1.6.0/mxnet_cu102-1.6.0-py2.py3-none-linux_aarch64.whl
  sudo pip3 install mxnet_cu102-1.6.0-py2.py3-none-linux_aarch64.whl

2.7.2. GluonCV

GluonCV is a deep learning computer vision toolbox, which provides the top-level algorithm implementation and basic operations of computer vision. GluonCV provides the implementation of top deep learning algorithms in the field of computer vision. GluonCV is simple and easy to use, there are many trained models, which can be downloaded and used with one line of code, which is very convenient. These models can be download from GluonCV model zoo.

Follow this instruction, it is convenient to install GluonCV computer vision toolkit. Just use the pip and download.*https://gluon-cv.mxnet.io/install.html*[12]

2.7.3. Why MXNET

In computer vision, when researchers want to use the state-of-art deep learning models, repeating the experimental results in the paper is the most difficult thing.

Fortunately, in recent years, the open source has been deeply popular. Many papers can find open source implementations. Most users do not need to implement the article from scratch, just go to Github to find an implementation. But this does not solve all problems. For example, online implementations are mixed, and many of them are practiced by newcomers. It may not be possible to repeat the results of the paper.

This toolkit GluonCV provides many functions, such as providing pre-trained models that can be used directly and each model implementation and interface will be as consistent as possible to reduce the learning threshold for using new models. Compared to TorchVision(computer vision toolkit of PyTorch), GluonCV have more object detection models and more easy to use.

3. Benchmark Method

Benchmark is an import method in computer vision. Benchmark is to compare the differences between different approaches or deep learning models and not focus on comparing which approaches or models are better than others. It can be seen as a simple heuristic used as a reference when comparing deep learning models.

Benchmark will help developers quantify the lowest expected effects for specific problems. For example, The most successful application of Benchmark in the computer field is performance testing, which mainly tests load execution time, transmission speed, throughput, and resource occupancy rate.
Benchmark is a process and it consists of three steps:

- Setup, Set up according to the purpose of the experiment, which is usually the experimental setting to be explained before the experimental results of the paper. Choose the appropriate data set, algorithm, comparison algorithm, comparison parameter, etc.
- Execution: This part is to experiment according to the settings in the previous step.
- Analysis: Analyse the experimental results obtained in the previous step through various analysis methods to support the proposed algorithm or hypothesis.

The aim of project is to benchmark the SOTA deep earning models on Jetson nano. It will run object detection tasks on Jetson nano. SOTA means state-of-the-art, The algorithm that can be called SOTA shows that its performance is currently the best performance. There are one-stage object detection models and two-stage object detection model in computer vision. The two-stage models are named for their two-stage processing of pictures. For example, R-CNN abstracts detection into two processes. One is to propose several regions that contain objects based on the input image, in the article, the selective search algorithm is used. Two is to propose the best performance classification network(AlexNet) on the ares to get the category of the objects in each area[19]. The Faster-RCNN is the foundational work of 2-stage method, the Faster-RCNN model use RPN network replaces the selective search algorithm so that the object detection task can be complete by the neural network end-to-end[6]. The report will talk the Faster-RCNN model later. The 1-stage model does not have an intermediate region detection process, and obtain the prediction result directly from the input image. So 1-stage model is also called a Region-Free method. For example, YOLO and SSD model is 1-stage model and the report will discussed YOLO and SSD model in detail later. So the project will benchmark three object detection models, they are Faster-RCNN, SSD and YOLO-V3.

Table 3. Object detection models for benchmarking in the experiments.

Model	Dataset	Stage
ssd_512_resnet50	VOC Dataset	1-stage
faster_rcnn_resnet50	VOC Dataset	2-stage
yolo3_darknet53	VOC Dataset	1-stage

The deep learning model consists of a neural network and a series of internal parameters (or weights). After training, the model can process real-time data and provide real-time results, this process is called inference.

There is a report that use batch size 1 image classification task to run on different edge computing devices and use different deep learning frame. This report measure the real-time inference through the image classification task, and get an approximate number of processed frames per second(FPS)[17].

There is an inference performance of Jetson nano from NAVIDA, the inferencing used batch size 1 and FP16 precision[15].

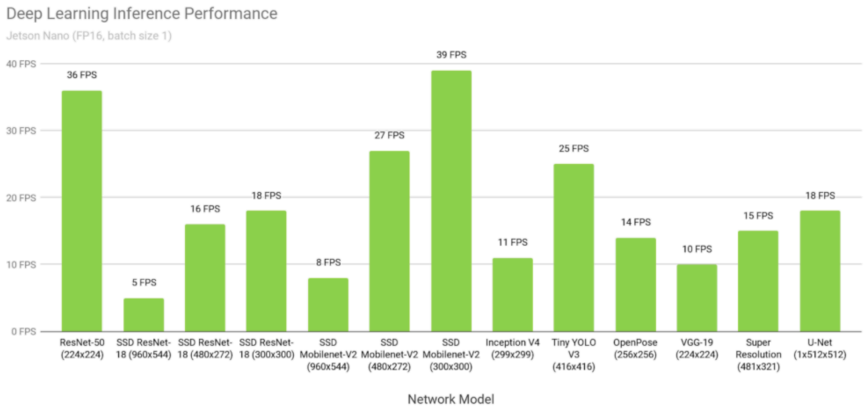

Figure 17. Jetson nano inference results from NAVIDA

This report will run pre-trained object detection models that from GluonCV model zoo on Jetson nano. It will use VOC dataset(validation dataset), and inference these input image with different batch size, not only batch size 1. During the experimental, in order to test the capabilities of Jetson nano, it will collect and compare the GPU temperature and swapfile changes of Jetson nano. As for these object detection models(Table 3), the inference, throughput, mAP and accuracy in validation dataset will be collected. These measures will be discussed in detail later.

Table 4. Elements for benchmark that use object detection task run on Jetson nano

Components	Measured
GPU temperature	inference task, run all VOC validation dataset, compare the change of GPU temperature.
Memory	inference task, run all VOC validation dataset, compare the change of swapfile.
FPS	frames per second
Inference time	total inference time on VOC validation dataset
mAP	measure object detection model(discussed in previous section)

3.1. Dataset

Deep neural networks is successful in application. An import reason is the large scale dataset. In object detection, it is not only necessary to find all the objects of interests in the input image, but also to the positions of the objects. The position is generally represented by a rectangular bounding box. There is no small dataset in the field of object detection. The report will give a quick introduction for VOC dataset.

3.1.1. VOC dataset

The PASCAL VOC Challenge (PASCAL Visual Object Classes) is a world-class computer vision challenge. The full name of PASCAL is : Pattern Analysis, Static Modelling and Computational Learning, is a organisation funded by the European Union.

Many excellent computer vision models such as classification, object detection, segmentation, action recognition and other models are based on the PASCAL VOC challenge and dataset, especially some object detection models. For example, Faster-RCNN, yolo and SSD.

For researchers, PASCAL VOC 2007 and PASCAL VOC 2012 are two import object detection datasets. The PASCAL VOC dataset consists of 5 parts: JPEGImages, Annotations, ImageSets, SegmentationClass and SegmentationObject.

- JPEGImages: store all images of training and testing.

- Annotations: Store the XML file corresponding to each picture after tagging

- ImageSets: There are main text files store in the folder, test.txt, train.txt, trainVal.txt and val.txt. The four files store the name of images. For example, the test.txt store the name of pictures in the test dates.

- SegmentationClass and SegmentationObject: store the image segmentation results, which is useless for object detection tasks.

PASCAL VOC dataset provides 20 types of objects and four main classes. Figure 18 show these objects.

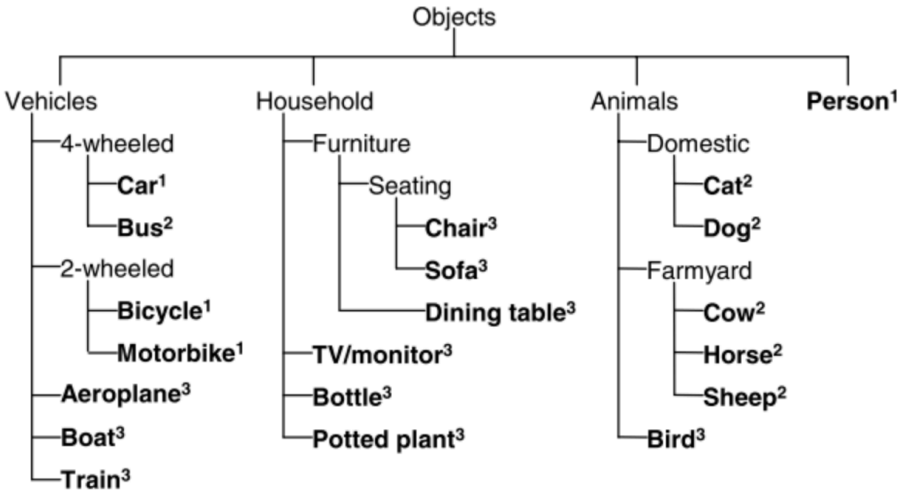

Figure 18. Type of objects in PASCAL VOC dataset.

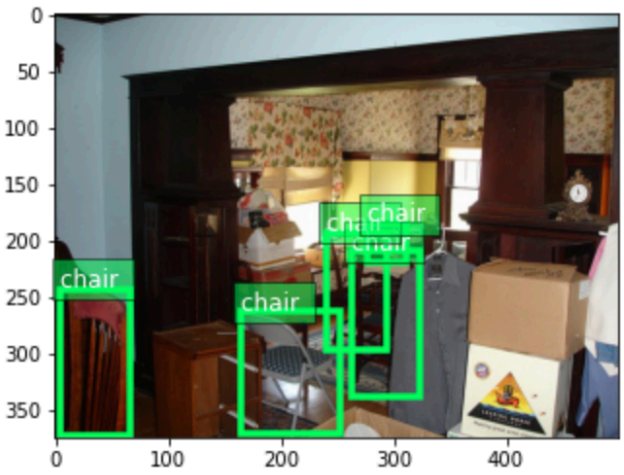

Figure 19. Visualise one example of PASCAL VOC dataset.

3.1.2. PIKAQIU dataset

There is no small dataset in the filed of object detection, which is unlike image classification. In the book " Dive into deep learning " the author make a small dataset named PIKAQIU dataset. This dataset include 1000 images with different sizes. MXNET provide the im2rec[10] tool that can convert image into binary format. This format will reduce the cost on memory and increase the efficiency of loading data.

Figure 20. Examples of PIKAQIU dataset.

3.2. Models

For better understand the experiments, this section will give describe the three object detection models(Faster-RCNN, yolo-v3 and SSD) in details.

3.2.1. SSD

The full name of SSD is Single Shot MultiBox Detector, single shot means that SSD is one-stage method and multi box indicates that SSD is multi-box prediction. SSD uses CNN neural network directly for object detection. There are two import points for SSD. One is SSD extracts feature maps of different scales for detection. Large-scale feature maps (the earlier feature maps) can be used to detect small objects, and the small-scale feature maps (the lower feature maps) can be used to detect large objects. Two is that the SSD uses a priori boxes of different scales and aspect ratios[20].

SSD use a CNN network for detection and use multi-scale feature map. The basic architecture is shown in Figure 21.

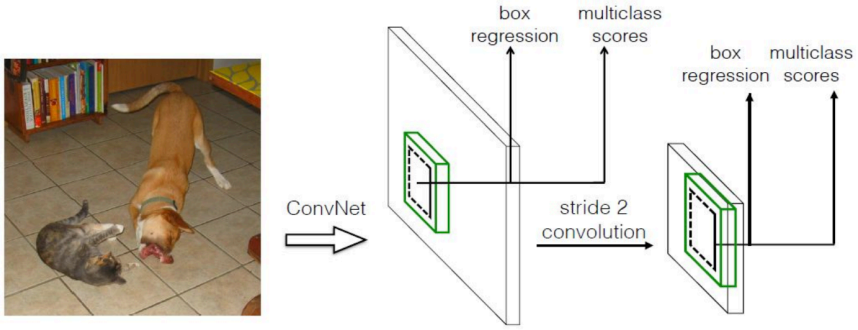

Figure 21 SSD basic architecture.

As for point one, SSD use multi-scale feature maps for detection. In CNN network, the front will have larger feature map, and then will gradually use stride=2 (In CNN section, stride=2 will half the width and height of the feature map) convolution or pool to reduce the size of the feature map. This is shown in Figure 22.

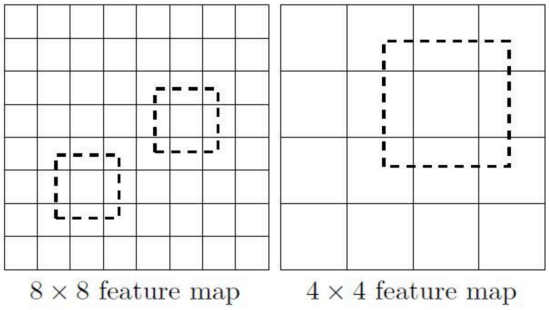

Figure 22 Multi-scale feature maps.

SSD also use prior boxes. SSD set each unit a priori boxes with different scales or aspect ratios, and the predicted bounding boxes are based on these a priori boxes. As shown in Figure 23, each unit uses 4 different a priori boxes. In the picture, cats and dogs use the best a priori boxes for their shapes for training.

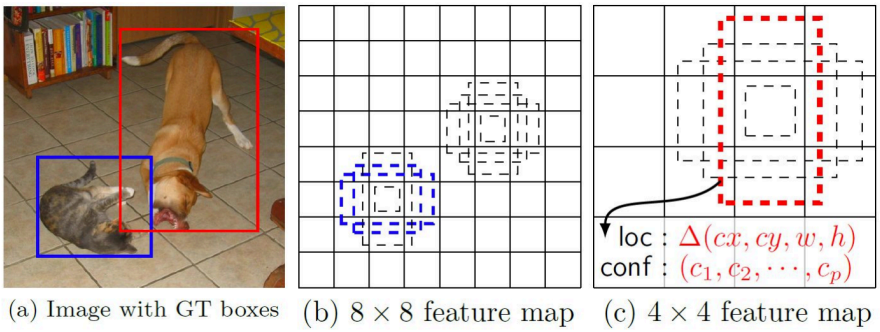

Figure 23 SSD prior boxes.

3.2.2. Faster-RCNN

Faster-RCNN is a state-of-art deep learning model in accuracy. As mentioned, it is a 2-stage model. Compare to RCNN and Fast-RCNN, Faster-RCNN replaces the traditional Selective Search method of extracting targets with training networks, which greatly improves the detection and classification speed of the whole process.Here is the basic architecture of Faster-RCNN[6].

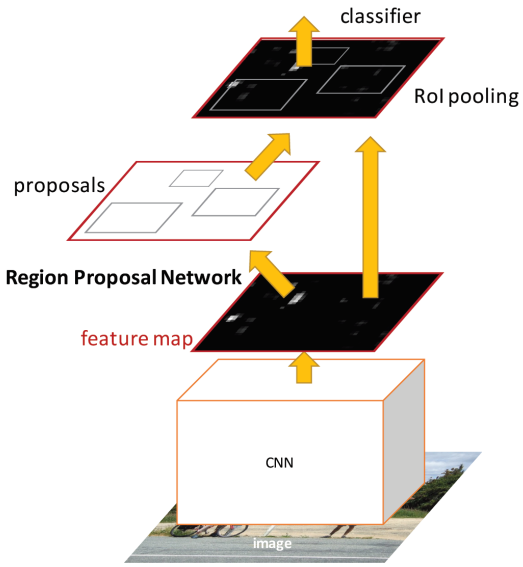

Figure 24 Faster-RCNN architecture.

Faster-RCNN can be divided into 4 part:

1. Convolutional layers, Faster RCNN first uses a set of basic conv+relu+pooling layers to extract image feature maps. The feature maps are shared for the subsequent RPN layer and fully connected layer.

2. Region Proposal Networks, The RPN network is used to generate region proposals. Softmax judges that where anchors belong and then this layer uses bounding box regression to correct anchors.

3. RoI(region of interests) Pooling, This layer collects the input feature maps and proposals, extracts the proposal feature maps and sends them to the fully connected layer to determine the object category.

4. Classification, Use the feature maps to calculate the category, and again bounding box regression to get the final position of the bounding box.

3.2.3. yolo-v3

Yolo-v3 is just to divide a picture into different grids, each grid is responsible for the prediction of the region. As long as the centre of object is located in this region, the object is determined by this grid. The prior detection of yolo-v3 reuses the classifier or locator for object detection tasks. It will use multi scales of image and those areas with higher scores can be regarded as the test results.

Yolo-v3 use multi-scale prediction, better basic CNN network(residual network) and binary cross-entropy loss for classification loss.

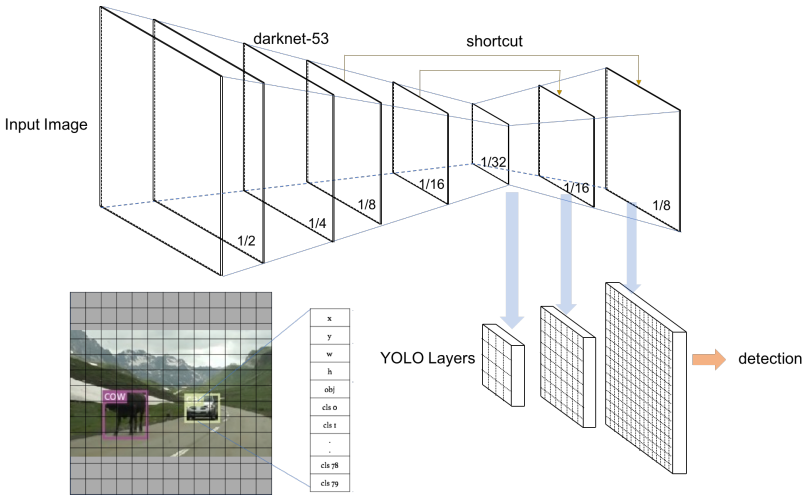

Figure 25 Yolo-v3 architecture.

3.3. Measures

This section will explain the methods that used in the experiments. The aim of the project is to benchmark SOTA deep learning models on Jetson nano. As mentioned, SSD, Faster-RCNN and yolo-v3 will be benchmarked. During the process, it will collect the elements in Table 4 in section 3. These elements can compare the differences of the models and the capability of Jetson nano.

MXNET combines imperative programming and symbolic programming, the report will compare the influence of hybridisation by using inference task.

The report will also compare the cpu and GPU performance by using inference task.

3.3.1. Hybridisation

MXNET combines imperative programming and symbolic programming. Imperative programming is more convenient when use python, most of codes is easy to write and understand. Symbolic programming is more efficient.

In the experiment, it choose 100 raw images, and object detection models read the images for inference. It will compare the calculation time before hybridisation and after hybridisation.

3.3.2. CPU and GPU

Graphic Processing Unit(GPU) have significant influence on deep learning. The most suitable type of calculation for GPU is the calculation that can be in parallel. This is reason that GPU is faster than cpu in deep learning.

The report use 100 raw images, and the object detection models will read the images for inference. It will compare the inference time of using cpu and GPU.

3.3.3. VOC dataset

The main experiment will benchmark three pre-trained object detection models(SSD, Faster-RCNN and yolo-v3). Theses pre-trained models will run on VOC validation dataset, the number of pictures in the validation dataset is 4952. These models will run inference tasks, the batch size in the experiments is 8. The setup and environments has talked in previous section. when the models finish tasks, the change of GPU temperature and swapfile will be collected in order to realise the capabilities of Jetson nano, the total inference time, FPS and mAP also will be collected in order to compare different models.

The meaning of mAP has discussed in section 2.5.3. FPS(frames per second) is the number of frames that can be detected per second, which represents the running speed of deep learning model. FPS also measure the real-time performance of model.

4. Results

This section will show the experiment results. All the measures that used in benchmarking deep leaning models in section 3. By running object detection tasks on Jetson nano, it will help understand and identify the capabilities of Jetson nano and the performance of deep learning models. The default SSD is SSD_512_resnet50, the default yolo-v3 is yolo3-darknet53 and default Faster-RCNN is faster_rcnn_resnet50. If there is no special instructions, it is the default. Default iou_thresh is 0.5.

4.1. Hybridisation

MXNET provide the style of hybridised programming. In MXNET users can use HybridBlock class or HybridSequential class build deep learning model. When use the function hybridise, MXNET will convert to execute in the way of symbolic programming. As a matter of fact, only the layers that inherit the HybridBlock class will be optimised. The source code of object detection models is in this link[18]. Most models can execute in the style of hybridised programming.

In this experiment, the pre-trained object detection models will read 100 raw images for inference, batch size is 1. The size of images is 512 * 512. The aim of this experiment is to compare the influence of hybridised programming on calculation performance.

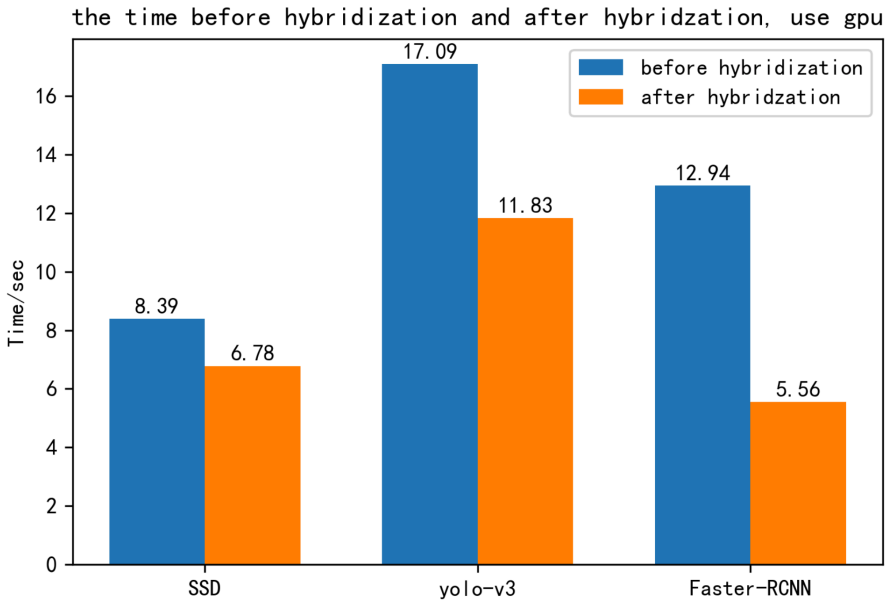

Figure 26 Total inference time before hybridisation and after hybridisation, use gpu.

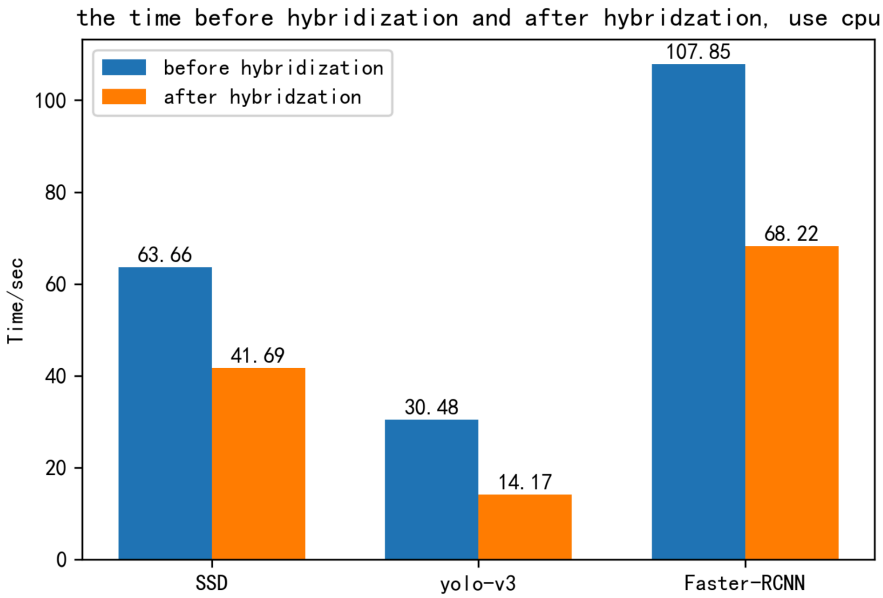

Figure 27 Total inference time before hybridisation and after hybridisation, use cpu.

As the result is shown in Figure 26 and Figure 27, the performance of calculation in all models (SSD, Faster-RCNN and yolo-v3) has improved after hybridisation both on cpu and gpu. Hybridisation method may improve the computing performance of deep learning models.

4.2. CPU and GPU

MXNET allow users choose storage and computing devices, for example, use cpu or gpu. MXNET requires all input data to be calculated in memory and same device. When initialise the deep learning model, users can set the device through the *CTX* parameters.

In this section, it will show the result of cpu and gpu. This experiment will compare the total inference time of using cpu and gpu. The object detection models will read 100 images for inference, the batch size is 1, the size of images is 512*512. During the process of inference, the deep learning models will infer images and return the object classes, bounding boxes and scores.

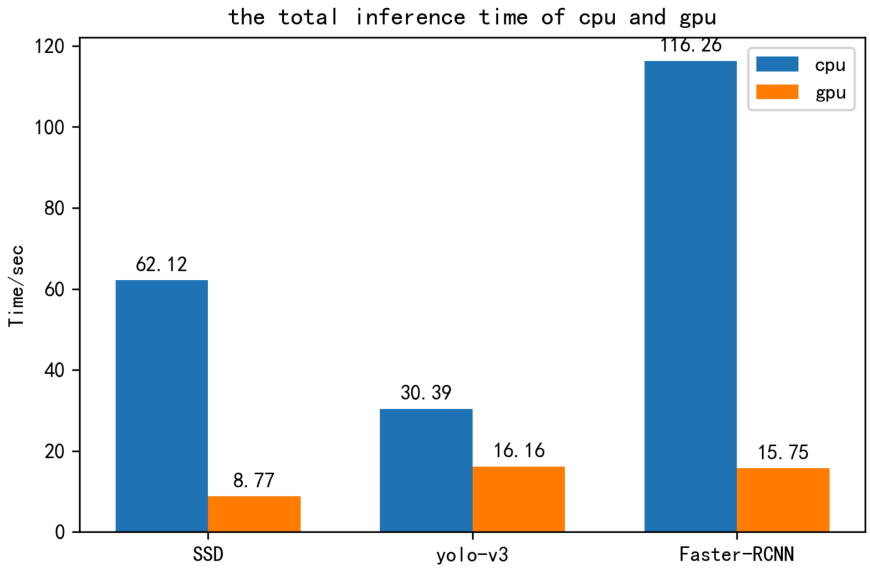

Figure 28 Total inference time of cpu and gpu. batch size 1

The result shows that the object detection use gpu for calculation is much faster than use cpu for calculation. Every object detection models in the Figure 27 improve the performance when they use gpu.

When use cpu, yolo-v3 is faster than SSD and Faster-RCNN, but when use gpu the performance of yolo-v3 is similar to Faster-RCNN. In the previous section, it have discussed the Faster-RCNN is a two-stage model, and Faster-RCNN is SOTA model in accuracy.

4.3. VOC dataset

As mentioned, this section will show the results that three pre-trained object detection models run inference tasks on VOC validation dataset. Total inference time, FPS, mAP, the change GPU temperature and the change swapfile for each model that on Jetson nano will be shown. Batch size is 8.

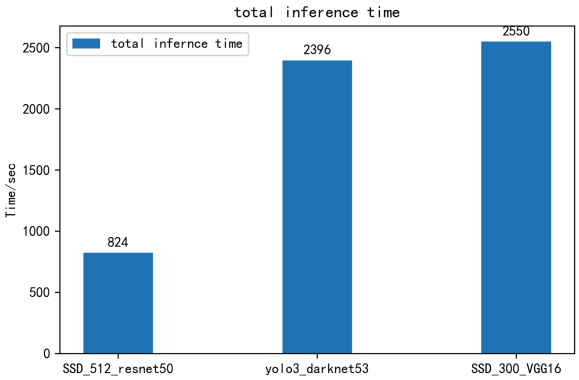

Figure 29 Total inference time on VOC validation dataset batch size 8.

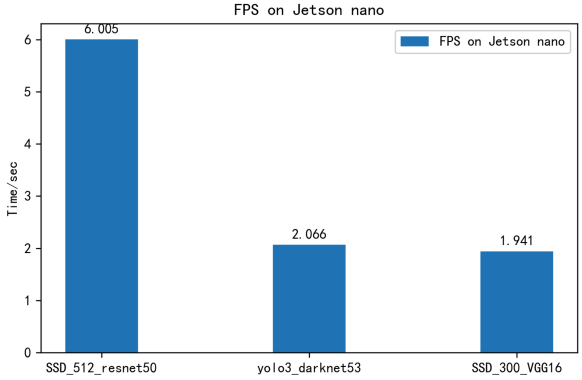

Figure 30 FPS on VOC validation dataset, batch size 8.

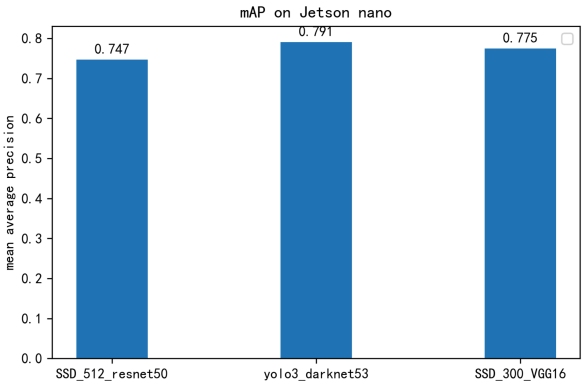

Figure 31 mAP on VOC validation dataset, batch size 8.

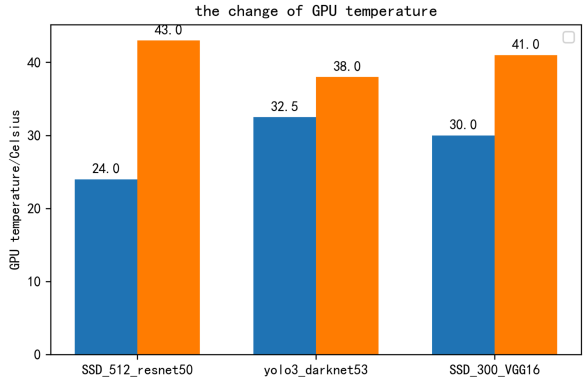

Figure 32 The change of GPU Temperature on VOC validation dataset, batch size 8.

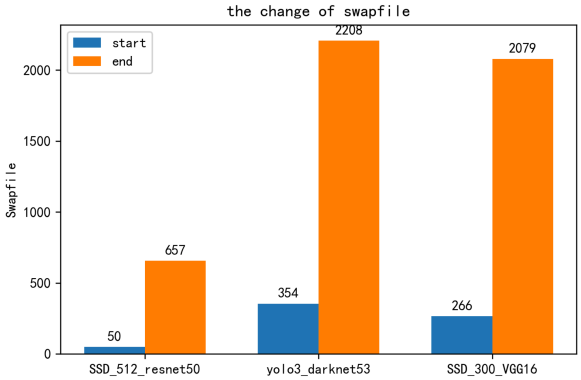

Figure 33 The change of Swapfile on VOC validation dataset, batch size 8.

When run the pertained Faster-RCNN(faster_rcnn_resnet50) on the VOC validation dataset, the object detection task failed. The system shows that CUDA is success, but too many resources requested for launch. The most possible reason is that the graphic memory is too small for the this model. In other words, the Faster-RCNN need more graphic memory than SSD_512_resnet50, yolo3_darknet53 and SSD_300_VGG16.

The results show that SSD_512_resnet50 is faster than SSD_300_VGG16 in total inference and FPS. SSD_512_resnet use the ResNet architecture, this is reason for the result. He(2016) introduce the ResNet that using residual blocks can train the deep neural network more effective[4].

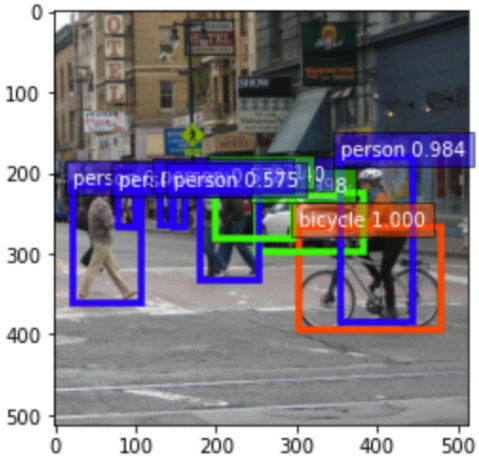

Figure 34 Inference result of SSD_512_resnet50

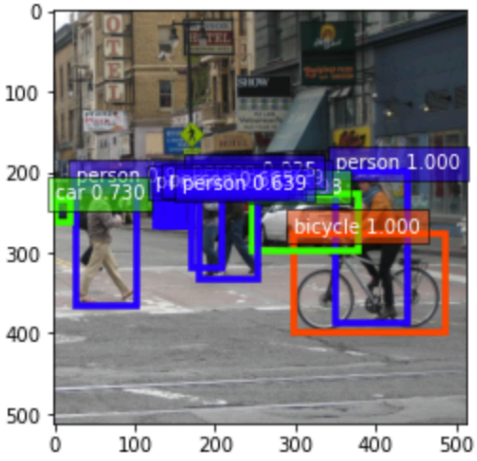

Figure 35 Inference result of SSD_300_VGG16

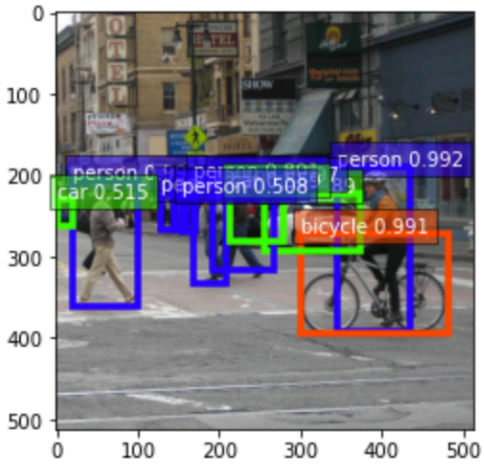

Figure 36 Inference result of yolo3_darknet53

5. Conclusion

Edge computing has signifiant influence on the development of Internet of Things, because it can deal with the real-time data processing. In addition, Edge computing can relieve network pressure and improve the security of both users and networks.One of most headache problem for edge computing is to deal with images. Deep learning techniques make a lot progress in the field of image processing. It is import to balance the edge computing and deep learning techniques.

This report will run object detection tasks on Jetson nano in order to collect the information of the capability of Jetson nano and the SOTA deep learning models. It will help researchers when they deploy deep learning models on Jetson nano.

The report discussed the background information, such as neural work, Jetson nano and object detection in the field computer vision. After introducing the basic concepts of SSD, Faster-RCNN and yolo-v3, the report benchmark these SOTA object detection models by running inference tasks. The hybridisation of MXNET and the use of GPU will improve the inference speed of these models. When run these models on VOC validation dataset(different batch size), the project also collect the information of Jetson nano, such as GPU temperature and Swapfile. As the results show, the SSD_512_resnet50 has the highest FPS, yolo3_darknet53 is the champion of mAP, the biggest change of GPU temperature is SSD_512_resnet50 and the change s of Swapfile for yolo3_darknet53 and SSD_300_VGG16 are bigger than the change for SSD_512_resnet50. The Faster_RNN_resnet50 fail on the inference task on VOC validation task, the most possible reason is that there are no enough graphic memory for it. As mentioned, Faster-RCNN is a 2-stage model.

The introduction is shown in the section 2, the result of experiment is shown the section 4.

5.1. Future Work

Most deep learning frame like TensorFlow, MXNET will use the data accuracy of float 32 to train the neural network. This is the nightmare for the embedded devices, because the computing ability and memory of embedded devices are limited that can not compare with PCs. But it can use low-precision data (INT8) when deploying inference. When the deep learning models use low-precision data , there is less hardware consumption for embedded devices. The author of the article thought that even if low-precision data, such as INT8, is used in inference, it will not cause too much accuracy loss and it will improve the speed of inference[16]. Researchers can get the pre-trained quantified models from GluonCV model zoo. It is valuable to benchmark these deep learning models on Jetson nano.

Acknowledgments. I would like to thank my supervisor Professor Raj Ranjan for supervisor and time during the project. I am quite enjoy the project and benefit a lot. And I would like to thank Dr Bin Qian for spending the time and effort on me.

References

1. Sultana, F. et al.: Advancements in image classification using convolutional neural network. In: 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN). pp. 122–129 IEEE (2018).
2. TURING, I.B.Y.A.M.: Computing machinery and intelligence-AM Turing. Mind. 59, 236, 433 (1950).
3. Shihadeh, J. et al.: Deep learning based image classification for remote medical diagnosis. In: 2018 IEEE Global Humanitarian Technology Conference (GHTC). pp. 1–8 IEEE (2018).
4. He, K. et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016).
5. Shi, W. et al.: Edge computing: Vision and challenges. IEEE internet things J. 3, 5, 637–646 (2016).
6. Ren, S. et al.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in neural information processing systems. pp. 91–99 (2015).
7. Navida: Getting started with Jetson nano Developer Kit, https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#intro, last accessed 27/04/2020.
8. LeCun, Y. et al.: Gradient-based learning applied to document recognition. Proc. IEEE. 86, 11, 2278–2324 (1998).

9. IDC report: The Digitization of the World From Edge to Core, https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf, last accessed 10/05/2020.
10. Mxnet: im2rec tool, https://github.com/apache/incubator-mxnet/blob/master/tools/im2rec.py, last accessed 20/06/2020.
11. Krizhevsky, A. et al.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012).
12. GluonCV: Install GluonCv, https://gluon-cv.mxnet.io/install.html, last accessed 2020/06/12.
13. JetsonHacksNano: Install Swapfile, https://github.com/JetsonHacksNano/installSwapfile.git, last accessed 05/06/2020.
14. Lai, Y.-K. et al.: Intelligent vehicle collision-avoidance system with deep learning. In: 2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS). pp. 123–126 IEEE (2018).
15. Navida: Jetson nano:Deep Learning Inference Benchmarks, https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks, last accessed 20/06/2020.
16. Ungrapalli, V.: Low Precision Inference, https://towardsdatascience.com/low-precision-inference-with-tensorrt-6eb3cda0730b, last accessed 12/07/2020.
17. Pablo, J.: Machine learning edge devices:benchmark report, https://tryolabs.com/blog/machine-learning-on-edge-devices-benchmark-report/#results-analysis, last accessed 12/05/2020.
18. Mxnet: Module code, https://gluon-cv.mxnet.io/_modules/index.html, last accessed 12/06/.
19. Girshick, R. et al.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014).
20. Liu, W. et al.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37 Springer (2016).
21. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv Prepr. arXiv1409.1556. (2014).

Appendices

The experiment results in Jetson nano.

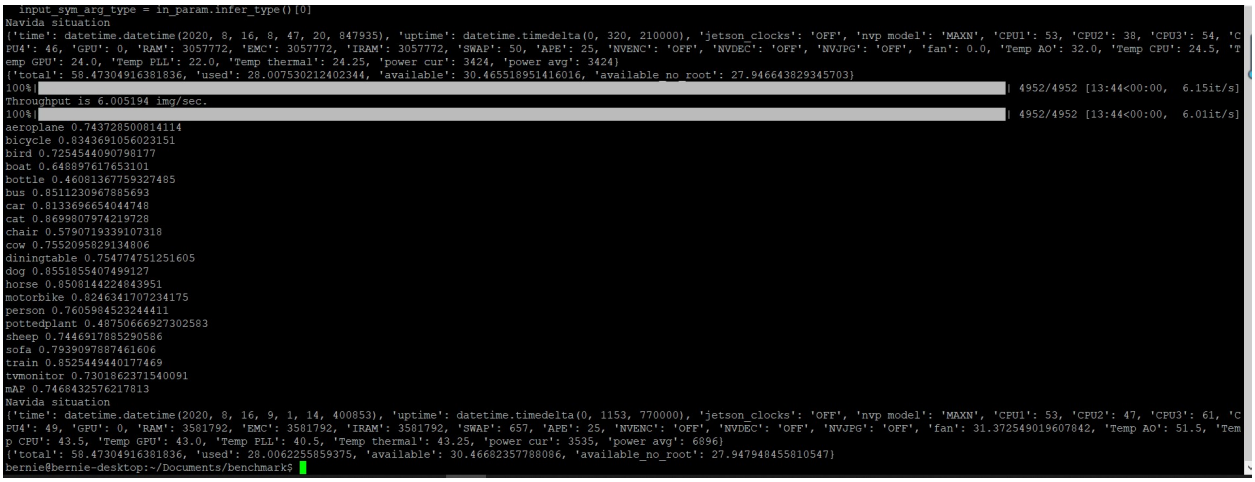

Figure 37 Result of SSD_512_resnet50 run VOC validation in Jetson nano

Figure 38 Result of yolo3_darknet53 run VOC validation in Jetson nano

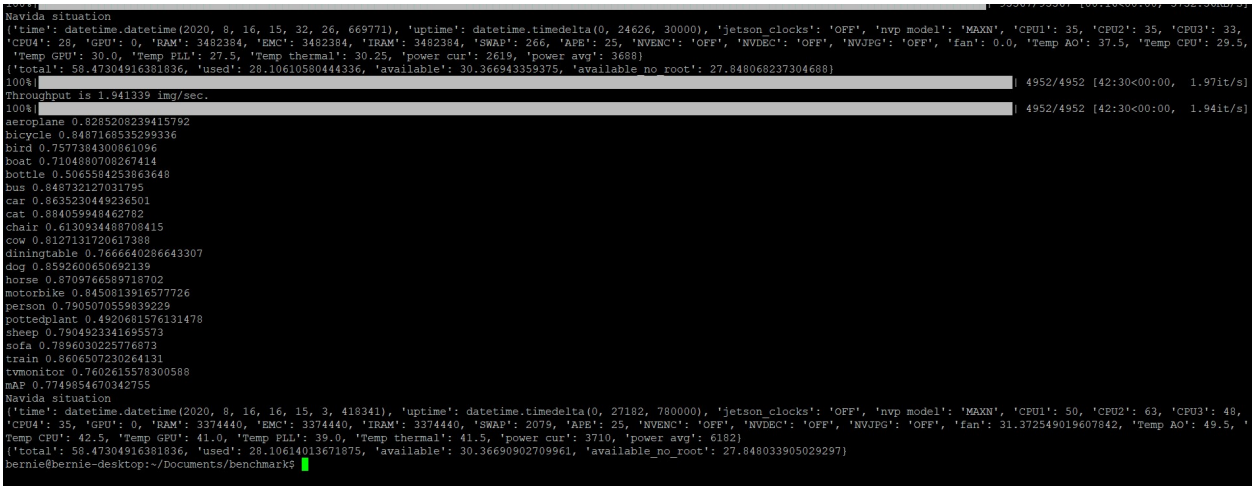

Figure 39 Result of SSD_300_VGG16 run VOC validation in Jetson nano