Active inference for embedded vision systems

Real-time object detection on embedded systems

Haidyn McLeod haidyn.mcleod@gmail.com

Abstract

Real-time robotic inference from vision systems is gaining eminence in today's robotics systems. This paper presents a multifunction neural net model for real-time object classification for use on resource-limited hardware systems, such as the Nvidia Jetson platforms', along with different forms of data acquisition for extending datasets.

I. Introduction

Real-time inference of vision system in robotics has recently acquired significant prominence in self-driving cars, drones, agriculture, food grading, warehousing and surveillance systems. These systems require fast and accurate interpretations of the vision data as delays and inconsistencies can have fatal consequences.

With robotic applications becoming more advanced and no longer being confined to controllable laboratory environments, robots now require self-contained onboard systems that are light, reliable and fast. Recent advancements in both hardware and model architectures have decreased the processing time of a model while simultaneously increasing the accuracies of the predictions.

Current state of the art systems involve large computer hardware to perform time-critical inferencing, but as robots move into more complex and remote environments, network communications can be limited as well as their size and shape limitations can limit the onboard systems. This paper will explore robust methods for training a network-based inference model that can be deployed remotely onto a Nvidia Jetson TX2 embedded platform that provides above 75% classification accuracies at over 100Hz update rate.

The trained model and image processing Python file is available at https://github.com/Heych88/robond-inference.

II. Background

In 1998, a network by LeCun pioneered a seven-layer convolutional neural network (CNN) that takes handwritten numbers and outputs a probability map for each number class called LeNet [1]. In 2012, AlexNet [2] built on this principle and was the first deep neural network (DNN) to win the ImageNet classification challenge, with a top-5 error of 15.4%. Current approaches have improved further on object classification in computer vision by employing various forms of the CNN architecture. The most notable of these architectures include ResNet [3], GoogLeNet [4] and VGG [5].

A. Hardware and Performance

With the improvements of these architectures, has been an increase in accuracy but at the expense of computational resources and time. A computational analysis of state-of-the-art DNNs', trained on ImageNet and deployed on a Jetson TX1, Canziani et al. [6], compared the top1 accuracy, operations, parameters, inference time, power and memory usage. Table I and Figure 1 shows Canziani's results for a selection of the tested models using the ImageNet dataset on the TX1 [6, 7].

TABLE I. Network Hardware Metrics

DNN	AlexNet-BN	GoogLeNet	VGG-16	ResNet-101	Inception-v3
Top 1 (%)	57	68	71	76	78
Opperation(G-Ops)	2	3	31	16	12
Parameters (M)	60	7	135	45	25
Inference (fps)	50	33	6	10	12
Power (w)	10.9	10.7	11.8	12.9	11.6
Memory (MB)	310	200	850	300	200

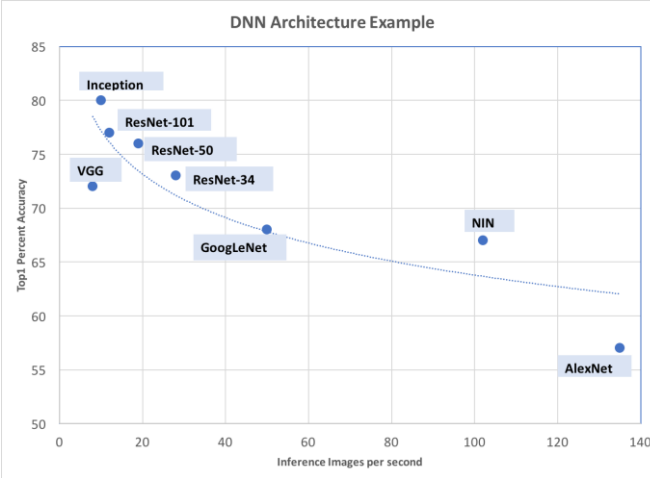

Figure 1: Correlation between inference speed and top-1 accuracy [7].

Since the release of the Canziani paper [6], processes and state-of-the-art features have improved the execution time and the inference speed linearly. Nvidia has since released the Jetson TX2 platform that has increased clock speed across the board

while decreasing power by 30%. Table II compares the Jetson TX1 and TX2 performances on the GoogLeNet architectures.

TABLE II. HARDWARE PLATFORM IMPROVEMENTS

Hardware		TX1	TX2
Max Clock (MHz)		998	1302
GoogLeNet batch = 2	**Inference (fps)**	141	201
	Power (AP+DRAM)	9.14 W	10.1 W
	Efficiency	15.42	19.9
GoogLeNet batch = 168	**Inference (fps)**	204	290
	Power (AP+DRAM)	11.7 W	12.8 W
	Efficiency	17.44	22.7

A second improvement has been on the platform's software. The latest JetPack 3.1 and associated TensorRT 2.1 improves speed by a factor of two. The following Figure compares software versions on the TX2 platform.

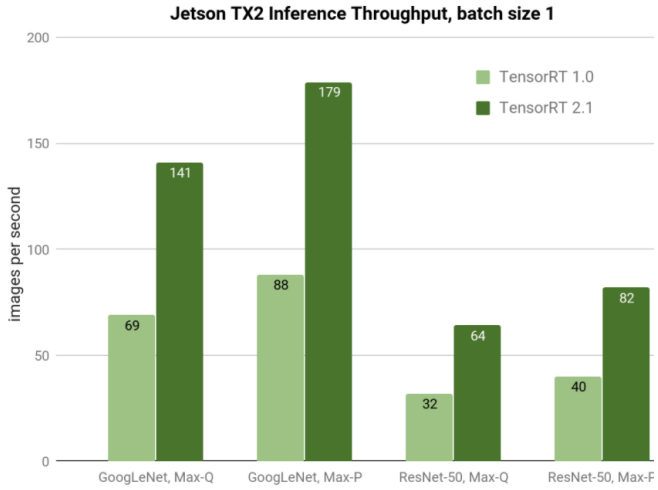

Figure 2: Inference throughput of GoogLeNet using Jetson TX2 with TensorRT version 1.0 vs version 2.1 [7].

The primary goal of this paper is to continue the work of Canziani and test the effectiveness of the GoogLeNet architecture for real-time image analysis on two different datasets of 256x256x3 images at an inference speed of 100Hz with the latest hardware and TensorRT software.

B. *Model*

Correct model selection is dependent on the system conditions that the model is to function. If accuracy is important a more complex architecture, at the cost of speed, can be used, such as life-critical situations or a more straightforward model can be used for faster inference for food sorting machines can require over 300fps.

The GoogLeNet architecture was chosen as the inference model because it offered the smallest resource utilisation with a high inference rate that has comparable accuracy to other resource intense architectures, as shown in Table 1 and Figure 1. With the updated hardware and software, GoogLeNet will be able to produce the target 75% classification accuracy at 100Hz frame rate.

C. *Parameters and Training*

Before deploying the inference onto the TX2, the model is trained using Nvidia DIGITS, a cloud-based deep learning training system. Digits contains predefined network architectures that also speed up and simplify model development.

Both models were trained using the Nesterov's Accelerated Gradient (NAG) [8]. NAG is a first-order momentum optimisation method with better convergence rates than gradient descent in certain situations. In particular, for general smooth convex functions and a deterministic gradient, NAG achieves a global convergence rate of O(1/T2) [9].

An exponential learning rate was employed with a gamma of 0.97 for dataset one and 0.95 for dataset two with a base learning rate of 0.01, over three epochs for the supplied conveyor dataset and 10 for the sports balls dataset.

10% of the data was used as validation data, and all the remaining parameters remained as DIGITS default values.

III. DATA ACQUISITION

Ideal data acquisition would be captured from the hardware device under all possible conditions. However, this may not always be possible, or a dataset may require additional images. Considering this, the model was tested against two datasets. The first is a controlled condition uniform three class dataset and the second is a seven class, non-uniform dataset, acquired using batch downloading software, Fatkun, of Google images which provided significant variations in the data.

The first dataset is the bottle-candy box dataset containing only bottles, candy boxes and nothing images on a conveyor belt to simulate the system under controlled conditions and has been acquired with the Jetson TX2's onboard camera mounted above a black conveyor belt. A sample of the dataset is in Figure 3.

The second sporting data was designed to test the model in non-uniform environments and contained tennis balls, footballs, American footballs, rugby balls, basketballs, volleyballs and random sporting environments called the sports' ball dataset. Each batch of balls produced between 246 (basketball) to 695 (rugby) images which were then modified in a combination of, resized objects and flipped horizontally to produce a six-fold increase in unique images per original image. A sample of the dataset is in Figure 4.

Figure 3: Sample of the bottle-candy box dataset.[7]

Figure 4: Sample of the sports' balls dataset.

IV. Results

A. Bottle-Candy box dataset

The bottle-candy box dataset contains over 10000 labelled images of candy boxes, bottles and nothing. The results of the training obtained an overall accuracy of 99.8047%. Figure 5 shows the system loss and accuracy during training.

The model's inference speed is averaged over five batches of 10 runs of data. The average inference speed was 5.282958ms or 189.29 fps with an accuracy of 75.4098%. Figure 6 shows the test results.

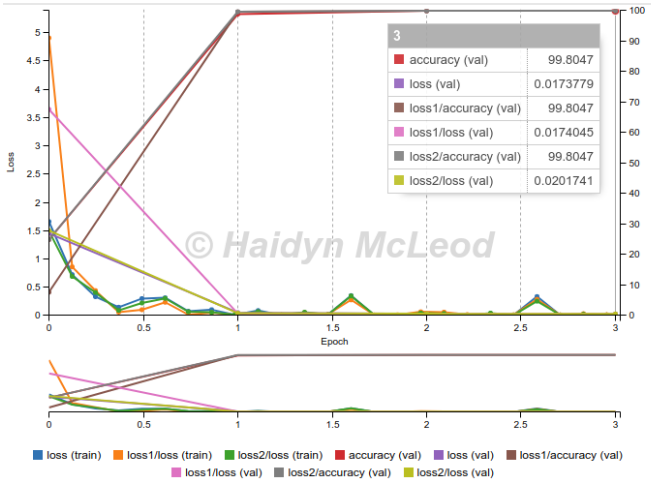

Figure 5: Training loss and accuracy over time for the bottle-candy box dataset.

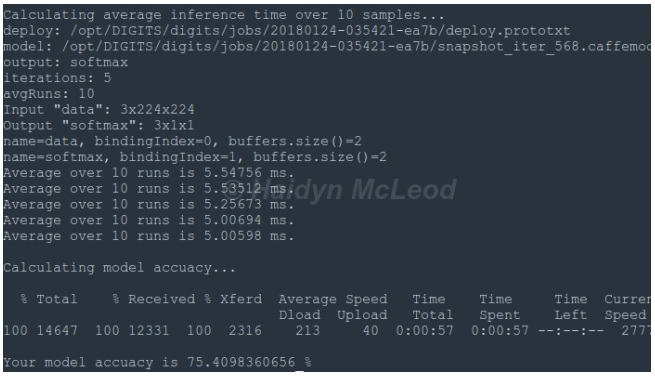

Figure 6: Screen shot of the trained bottle-candy box model under test conditions.

B. Sports' ball dataset

The sports' ball dataset contains over 22000 labelled images in seven classes. Figure 7 shows the breakdown of the class data used for training and validation.

The results of the training obtained a top-5 validation accuracy of 99.9114 and an overall accuracy of 92.9521. Figure 8 shows the system loss and accuracy during training; Table III shows the obtained training losses and accuracies of the system after training.

The model was tested on a small subset of selected training and validation data. The data was selected to contain images of just the class object and environment rich images containing multiple of other objects. Table IV shows the confusion matrix for class inference. The first column or **bold** type is the labelled class being tested, and the first row or *italic* is the predicted state.

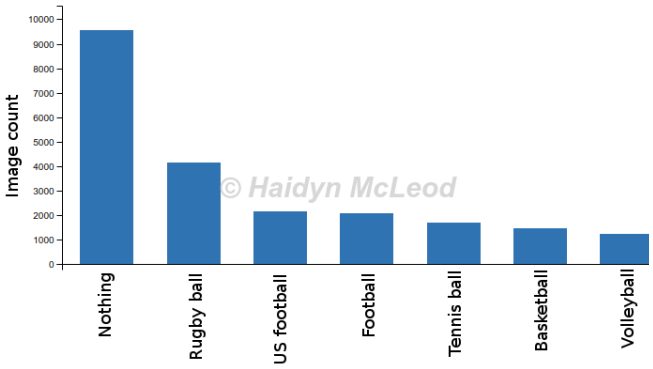

Figure 7: Sports' balls dataset breakdown.

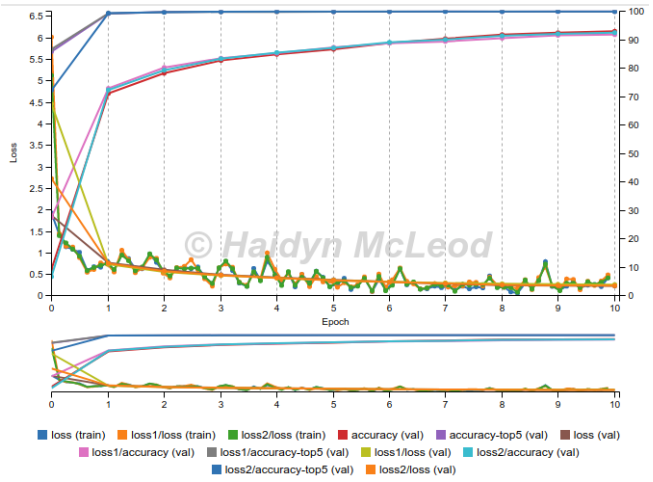

Figure 8: Training loss and accuracy over time for the sports' ball dataset.

TABLE III. NETWORK HARDWARE METRICS

Accuracy (val)	92.9521
Accuracy-top5 (val)	99.9114
Loss (val)	0.221611
Loss 1 / Accuracy (val)	91.75529999
Loss 1 / Accuracy-top5 (val)	99.9114
Loss 1 / loss (val)	0.253411
Loss 2 / Accuracy (val)	92.4645
Loss 2 / Accuracy-top5 (val)	99.9114
Loss 2 / loss (val)	0.226932

The model was tested on a small subset of selected training and validation data. The data was selected to contain images of just the class object and environment rich images containing multiple of other objects. Table IV shows the confusion matrix for class inference. The first column or **bold** type is the labelled class being tested, and the first row or *italic* is the predicted state.

N (nothing), RB (rugby ball), US (US football), F (football), T (tennis ball), B (basketball), V (volleyball).

TABLE IV. NETWORK HARDWARE METRICS

	N	*RB*	*US*	*F*	*T*	*B*	*V*
N	96.02	0.05	0.19	0.02	0.0	0.01	3.71
RB	0.91	57.32	13.35	25.87	2.51	0.01	0.03
US	5.15	31.1	62.8	0.93	0.0	0.02	0.0
F	18.36	14.9	17.92	34.64	1.58	7.4	5.2
T	0.0	0.35	0.08	1.52	97.97	0.0	0.08
B	8.95	0.1	0.49	0.35	0.21	73.94	15.96
V	8.95	0.26	0.05	1.91	0.17	0.56	88.1

A second set of images were tested that contained more complex information or a combination of classes in the same image.

Football

T : 99.91, **F** : 0.08, RB : 0.01

Football and tennis balls

F: 68.04, **T** : 19.27, RB : 10.8, US : 1.43, V : 0.43

V. DISCUSSION

The case study shows that embedded platforms can perform real-time inference on a range of classes in various environments.

The results here show that with the correct training and dataset, accuracy can be achieved in real-time. The results of the tests and experiments suggest that accurate, real-time inference can be achieved with embedded hardware with complex CNN architectures. The model has successfully demonstrated this with the bottle-candy box dataset. The demonstration of the sports' ball dataset has provided valuable feedback on the importance of the data-set and its means of collection. Although the sports' ball dataset achieved mixed results, it has been shown that despite limited data, less than 600 per class, placed in a wide variety of environments, it can still produce remarkable results with only 10 epochs of training. As can be observed in Figure 8, the validation accuracy continues to rise and given more training time, has the potential to increase a further 2-3%.

The second dataset collection approach presented here features some potential problems that can affect the model if not correctly handled. They are;

- Google images contain a variety of unrelated images for the dataset that can incorrectly influence the model if not removed from the set.
- A small dataset of a class can have substantial variation in colours, backgrounds, positioning, texture, shape and size. This can be seen in the football dataset, where the only consistency observed was that of the shape. (This is a contributing reason behind the low football inference accuracies).
- Images that are not captured by the target device can have variations in image size and quality. (It was observed that wider images resized to the 256x256 shape turned oblong rugby balls into circular footballs).
- Classes with consistent background environments that are rarely seen in other classes can cause the model to learn the environment rather than the desired object. (Volleyball class).
- Nothing class can have classed objects present in the form of a small portion of a ball being held in the image and not being correctly removed or markings on the ground or clothes that are the same as the class object such as a logo.
- Images can contain multiple classes.
- Large images with a smaller sized object can lose the detail of the object when resized to 256x256, causing miss-classification.

The sports' ball model suffered from all of the above dataset issues. Still, given the ease and accessibility of Google images, the model has produced an overall correct inference of the classes. Accuracy will improve with more data that is produced minimising the above image collection problems. Further improvements can be obtained by using a more accurate network architecture, as shown in Figure 1, at the cost of inference time.

The predictions obtained from the tested dataset contained a small subset of 20 images that provided ambiguous and potentially confusing situations. This was deliberately chosen to test the models against difficult situations. For the model to accurately predict. A more extensive and broader test set is required to obtain a more accurate model prediction.

VI. Conclusion / Future Work

The paper extends the work done by Canziani [6] and uses recent hardware and software advancements to train and accurately classify small user-defined datasets for embedded systems. While numerous CNN architectures offer accuracy vs speed, the GoogLeNet architecture was used to produce an accuracy of 75% at an inference speed of 189.29 fps.

The network was trained on two different datasets, a three class uniform environment that was obtained using the same TX2 onboard camera and a seven class, non-uniform environment set obtained from Google image collections. Although this method of data collection requires special consideration, training a model with this non-uniform data can decrease the prediction accuracy, it will produce a model that can generalise images under unexpected environments much better. The larger and more uniform the dataset for training, the more precise and generalised the model can predict in unforeseen environments and to aid this in future deployment, datasets should contain a mixture of system obtained images with filtered potential and unexpected images.

Due to the small and various datasets, the approaches taken are intended to only show the capabilities and limitations of a model trained with two different data acquisition methods that can be used for training models for use on embedded platforms.

References

[1] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-based learning applied to document recognition", 1998

[2] A. Krizhevsky, I. Sutskever, G. Hinton, "ImageNet classification with deep convolutional neural networks", 2012

[3] K. He, X. Zhang, S. Ren, J. Sun, "Deep residual learning for image recognition"' arXiv: 1512.03385, 2015

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, "Going deeper with convolutions", arXiv:1409.4842, 2014

[5] K. Simonyan, A. Zisserman, "Very deep convolution networks for large-scale image recognition", arXiv: 1409.1556, 2015

[6] A. Canziani, E. Culurciello, A. Paszke, "An analysis of deep neural network models for practical applications", arXiv:1605.07678, 2017

[7] Udacity, Nvidia – Robotics nanodegree, "Inference lecture modules. 2018

[8] Nesterov, Y. "A method of solving a convex programming problem with convergence rate O(1/sqr(k))". Soviet Mathematics Doklady, 27:372–376, 1983.

[9] I. Sutskever, J. Martens, G. Dahl, G. Hinton, "On importance of initialization and momentum in deep learning" 2013