Deep Learning and Object Classification at the Edge

CS293B – Cloud Computing, Edge Computing, and IoT

David Weinflash

June 3, 2020

Abstract

The following CS293B Course Project pursues an Application Project, integrating an IoT device, Edge device and Cloud computing environment to explore machine learning at the Edge. Specifically, the project introduces a system that uses a robot's camera feed and a deep learning model trained in a Cloud environment to perform object classification on a Raspberry Pi. After setting up the infrastructure, the project focuses on evaluating the most optimal deep learning models to deploy at the Edge, analyzing the trade offs between size, speed and accuracy to best perform object classification.

1 Introduction

Using an architecture similar to *Where's The Bear* [1], the following project sets out to perform on-device machine learning inference at the Edge in order to minimize latency and optimize the computing resources between an IoT device and a deep learning framework. The IoT device, or a toy robot named Cozmo created by Anki, is primarily capable of driving, altering camera angles and sending images to the nearby Edge cloud. The Edge cloud, or a local Raspberry Pi, is set up to act as a light weight server that handles client requests, performs inference, returns results and stores a trained deep learning model. Finally, a Cloud environment, or Google's hosted Jupyter notebook service Google Colaboratory, is utilized for its powerful computing resources (GPU) to train deep learning computer vision models and store the corresponding training, validation and testing data sets.

Upon setting up the infrastructure, the project then pivots to focus on discovering the most optimal computer vision model to deploy at the Edge. Tasked with classifying the hand gestures "rock", "paper" and "scissors", prominent convolutional neural networks (mobilenet, inception_v3, resnet) are deployed, analyzed and contrasted with the more modern family of light weight EfficientNet models [2]. In the end, the most effective model is deployed to the Edge to perform object classification in an integrated IoT, Edge and Cloud environment.

2 Architecture

The project provides an end-to-end integration of a deep learning computer vision model in an IoT, Edge and Cloud infrastructure to perform object classification. The following subsections provide an overview of each individual device in the system.

2.1 IoT

Cozmo, a toy robot by Anki, provides the camera feed and corresponding images for input into the object classification model. In terms of system design, Cozmo's engine is packaged in an iPhone application. The robot itself is connected to the application through a local WiFi network. In order to program and control Cozmo, a PC (running Cozmo's Software Development Kit) is connected to the iPhone via USB.

Controlled via a Flask web page on the PC, Cozmo may drive, adjust its camera angle and send photos using *scp* to the local Edge cloud for classification. Results from the inference model are displayed in Cozmo's HUD, showing the predicted object class, class probability and total inference time (ms).

2.2 Edge

The Edge cloud is designed to be as light weight and efficient as possible. All in all, the files packaged in the Edge cloud (Raspberry Pi) include a TensorFlow Lite model and corresponding labels (13 MB), a TensorFlow Lite Interpreter to perform inference with the TensorFlow Lite model (5 MB) and the most recent image for classification sent by Cozmo (8 kB). Once deployed, the Edge cloud takes upon the role of a local server, handling only two client requests ("classify" or "exit"), and returning classification results in the "prediction, probability, inference time" format as described above.

2.3 Cloud

Google Colaboratory, or a hosted Jupyter notebook service with GPU compute capability, served as the integral cloud environment and was utilized primarily for model training and data storage. By providing GPU compute services, Google Colab allowed the most expensive operation in the machine learning pipeline, or model training, to be offloaded and executed as efficiently as possible. What is more, Google Colab - in conjunction with Google Drive - stored the most memory intensive files, or the training image data sets, on the Cloud and off local hardware.

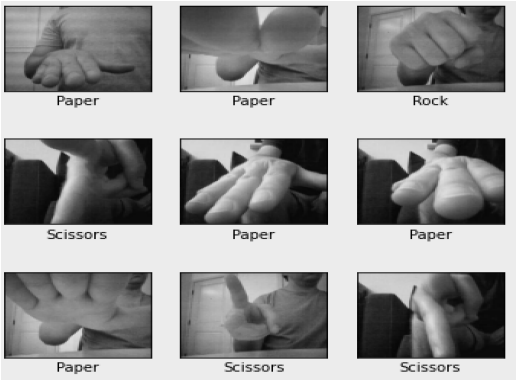
3 Classification

As a first attempt, the system was deployed using a pre-trained model and public data set. Specifically, an off-the-shelf Mobilenet model trained on the public data set ImageNet was deployed at the Edge. After this proof of concept, the project then pivoted in an attempt to find the most efficient convolutional neural network to classify the hand gestures "rock", "paper" and "scissors" in view of Cozmo's video feed.

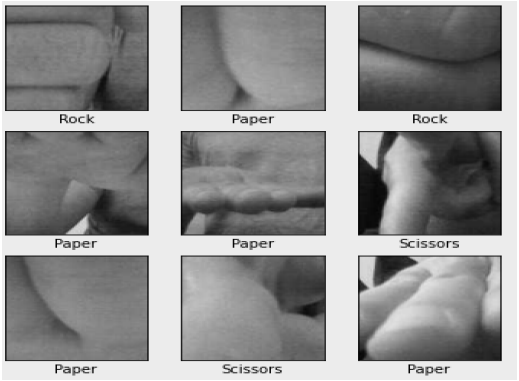
Figure 1: Object classification using an off-the-shelf Mobilenet trained on the ImageNet dataset.

3.1 Data Collection

A "rock, paper, scissors" training data set was compiled using images from Cozmo's camera feed. Specifically, the training procedure involved posing in front of Cozmo with a hand gesture and collecting photographs for 300 seconds, where Cozmo would take and store a photo every 1 second. The images would then be sent to Google Drive, where they would ultimately be imported into the Google Colab cloud environment during model training. After all data collection sessions and image augmentation techniques (random contrast, brightness, resize, crop, etc.) were complete, the final training data set accumulated to 3,600 raw images and 21,600 augmented images, or 25,200 images total.



(a) Raw

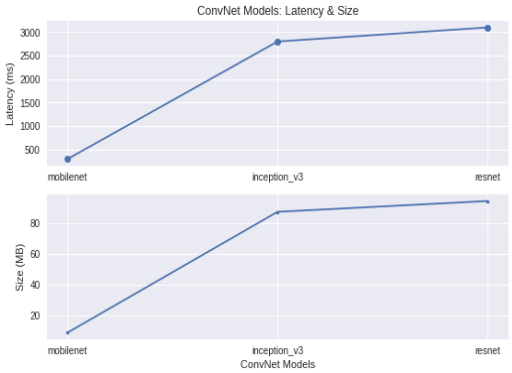


(b) Resize

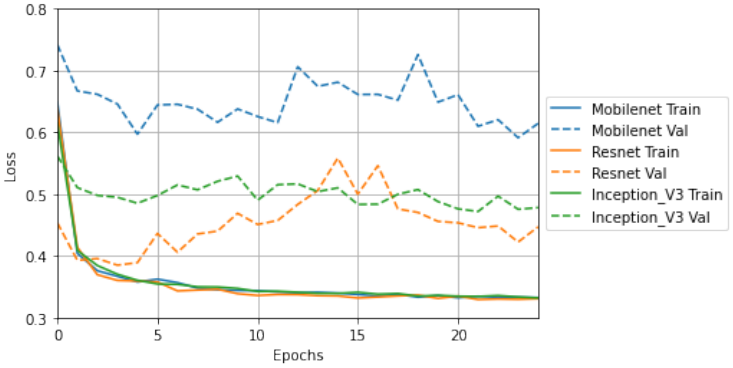
Figure 2: Raw and Augmented Training Images

3.2 Model Selection

In order to find an optimal image classification model, size, accuracy and inference speed were all taken into account. In particular, a group of popular convolutional neural networks (Mobilenet, Resnet and Inception_v3) are compared with a more modern family of scaled neural networks, or EfficientNets [2]. Marketed to be a best of both worlds architecture, EfficientNets use a *compound scaling* method to achieve high accuracy while minimizing floating point operations and limiting size. In the end, the EfficientNet-lite3 architecture offered the most promise with a high test set accuracy, quick inference speed and manageable size.

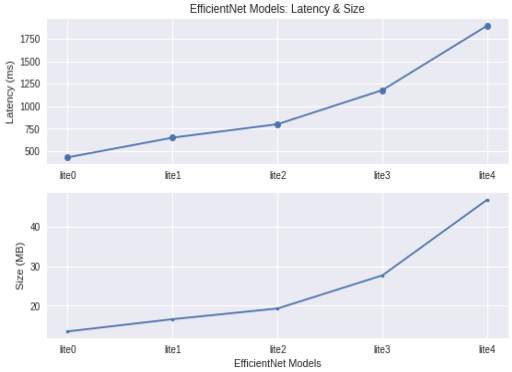


(a) Inference Speed and Model Size

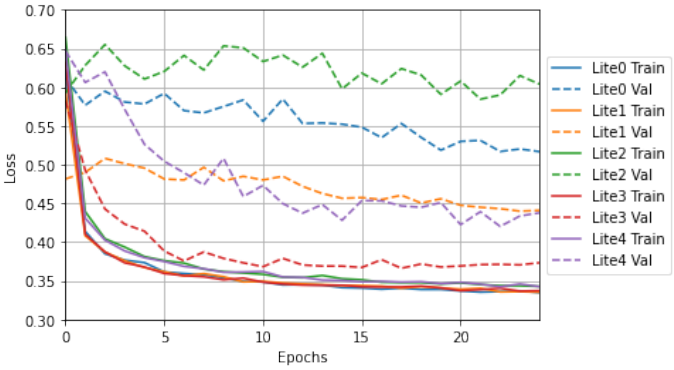


(b) Training and Validation Loss

Figure 3: Popular Convolutional Neural Networks

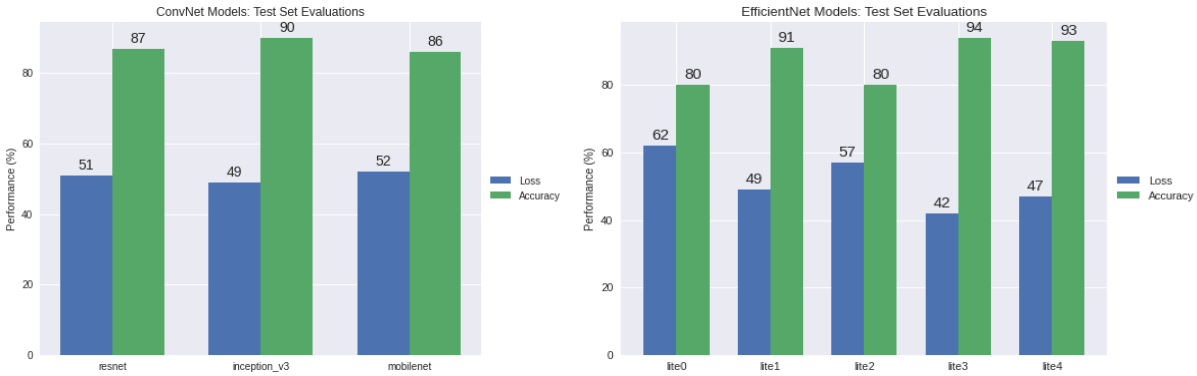


(a) Inference Speed and Model Size



(b) Training and Validation Loss

Figure 4: EfficientNet Models



(a) Convolutional Neural Networks

(b) EfficientNet Models

Figure 5: Test Set and Loss Performance

4 Discussion

Deploying machine learning models to the Edge and performing inference on-device has become ever more achievable with the TensorFlow Lite tool set and EfficientNet family of models. As compared to a group of well known image classification networks, EfficientNet models performed inference quicker, with higher accuracy and a smaller binary size. While more work needs to be done in order to increase the model's generalizability, the EfficientNet model provides a promising solution to delivering machine learning inference at the Edge.

References

[1] Chandra Krintz Andy Rosales Elias, Nevena Golubovic and Rich Wolski. Where's the bear? automating wildlife image processing using iot and edge cloud systems, 2017.

[2] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[1] [2]