Towards more efficient data preprocessing in deep networks

Fernando Peña Bes[*,1], Ana C. Murillo[*,1], Darío Suárez Gracia[*,1]

[*] *DIIS, I3A, Universidad de Zaragoza, Spain*

ABSTRACT

The analysis of highly dimensional data, such as hyperspectral images, is challenging due to the complexity and size of the datasets. We present an analysis of the bottlenecks associated to processing this kind of images for a recognition task in a recycling setup and show that preprocessing plays a critical role in the optimization of hyperspectral pipelines. These observations can help guide the design of future hardware accelerators close to the sensors that eliminate redundancies in the captured data, minimizing data transfers and contributing to efficiently process large datasets.
KEYWORDS: hyperspectral imaging; semantic segmentation; power consumption; hardware acceleration

1 Introduction

As deep learning models become increasingly prevalent in industry it is essential to consider their performance. Training and deploying these models require a significant amount of computational resources. As the complexity and size of these models increase, so does their demand for energy, which contributes to carbon emissions that harm the environment.

Therefore, many specialized accelerators have been proposed to speed-up deep learning, mainly focusing on the hidden layers of the networks [SCYE17, AJH+16]. However, depending on the sensor, the vast amount of data transferred between its acquisition and processing can consume a significant amount of energy and time, even moving data within the chip has become very expensive [Dal22]. Most importantly, reducing the amount of processed data can also help saving energy, specially in complex task such as classification that require large computing resources to often run deep networks.

Within this huge domain, our work focuses on the preliminary analysis of hyperspectral images in the context of a recognition task for a recycling plant. Specifically, we address the challenge of detecting voluminous objects within a recycling line, which have the potential to cause obstructions in subsequent processes. Hyperspectral images, although rich in information, often suffer from redundancy and high computational and storage requirements. With the objective of handling this data efficiently, we analyze an image processing pipeline to find its bottlenecks and compare alternatives for reducing the dimensionality of the data and for distributing the computation between a CPU and a GPU.

[1]E-mail: {ferp,dario,acm}@unizar.es

2 Methodology

Dataset The task at hand consists on segmenting two kinds of voluminous plastic objects in recycling images: large pieces of plastic wrap and plastic baskets. We prepared a small dataset 222 frames acquired using an image system equipped with a scan-line RGB camera and a near-infrared hyperspectral camera (also linear).

The spatial resolution of the cameras is 1184 and 640 pixels respectively. In order to process the data captured by the cameras in further stages, we first group successive scan lines into rectangular RGB and hyperspectral images with resolution 1200×1184 and 600×640. After that, we apply a preprocessing step that consists on cropping and downsampling them to 200×200 pixels with nearest-neighbors interpolation and normalizing their values to 32-bit float values in the range $[0, 1]$. In total, the dataset contains 213 segmentation masks for the films and 164 for the baskets (some samples are shown in Figure 1).

Although this dataset can be considered as a "toy" dataset in the sense of size and number of classes, it is derived from a real-world scenario and reflects the challenges encountering in addressing this specific problem, while it provides enough data to measure and evaluate performance insights.

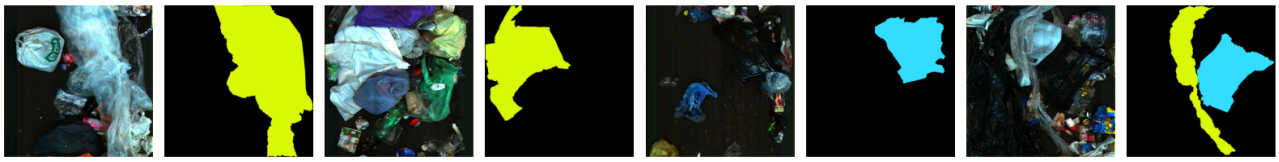

Figure 1: Examples of the RGB images and the annotated objects (large wrappings in yellow and baskets in blue)

Approaches compared in the study In our study we compared different strategies for handling the input data and utilizing the computational resources efficiently. All of them were integrated in a segmentation framework based on MiniNet-v2 [ARM20] implemented on top of PyTorch 2.0.1, which can be found at `https://github.com/ferpb/mininet-v2`. Here we describe the alternatives considered:

1. RGB image approach: This base approach uses only the RGB images as input to the segmentation model. It is used as reference, to check that the accuracy obtained by the different alternatives with hyperspectral data is reasonable.

2. Full hyperspectral approach: In this approach, we fed the entire hyperspectral data into the model, trying to leverage all the spectral information for accurate segmentation.

3. Linear combination of bands: As an alternative to using the full hyperspectral data, we experimented reducing its dimensionality with principal component analysis (PCA). This method projects the data to a lower dimensionality space in the way that better preserves the variability of the data. Because PCA is applied to the data as a preprocessing step before training and inference, we can easily perform it in a different device, so we considered executing this step in the CPU instead of in the GPU with the objective of reducing the size of data transfers. In all cases, the segmentation model runs on the GPU.

Metrics The results include metrics describing the accuracy of the classification and the efficiency of the computation. For the accuracy we use the mean intersection over union (mIoU) of the two classes considered, while for the efficiency we measure the global execution time, the partial execution time of certain steps (data transfers between devices, preprocessing, dimensionality reduction and inference) and the global energy consumption of each device (GPU and CPU).

Setup/Platform All experiments have been run on a system composed by a Intel Core i7-6700 CPU and an NVIDIA GeForce GTX 1070 GPU running Ubuntu 22.04.2 LTS and CUDA 11.7. Two tools have provided CPU/DRAM and GPU energy: `perf` and `nvidia-smi`. PyTorch's profiling and benchmarking modules helped measuring execution times.

3 Preliminary Results

Table 1 shows the global performance metrics for segmenting a single batch of 16 images. Although not stated in the table, the most time-consuming step in our evaluation setup is loading the images from the hard drive after they are captured by the cameras. On average, it takes 0.2 seconds to load a batch of 16 RGB images, whereas loading a hyperspectral batch takes 2.7 seconds approximately.

After the images are loaded in the computer's RAM, the time required for processing the hyperspectral batch is mainly influenced by preprocessing (downsampling and normalization) and data transfers to GPU, rather than the segmentation model itself. When performing PCA on the CPU, we see that this percentage is reduced as fewer data is transferred to the GPU, but because performing PCA in the CPU in a sequential way is slow, the overall time increases. On the other hand, CPUs usually require less power compared to GPUs, so in this base, spending more time on the CPU PCA and transferring less data to the GPU interestingly causes a slight reduction on the total energy consumption.

Regarding the mIoU metric, employing only RGB images already provides a favorable tradeoff between accuracy and time, but further improvements can be achieved by using hyperspectral images at the cost of increased time and energy consumption. In a problem domain like the one we are addressing, where interruptions in the recycling plant block the whole process, trading off additional time for enhanced accuracy can be an interesting consideration. We notice that PCA proves to be highly beneficial in extracting informative features and reducing the inherent reducing of hyperspectral data before using it as input to this neural network. When we apply PCA with 20 and 50 components and this step is performed in the GPU, both the mIoU and the efficiency improve.

4 Conclusions and Future Work

These preliminary results shed light on the bottlenecks associated with working with large data, in particular hyperspectral images. Our results highlight two primary challenges: preprocessing and data transfer between devices.

Efficient preprocessing techniques that can represent the data in a reduced form at the beginning of the pipeline are crucial. An exciting direction is to explore methods that are able to fulfill opposite requirements such as accuracy and energy efficiency by incorporating

Input	Reduction	mIoU ↑	Time (s) ↓	Preparation (%)	Energy (J) ↓
RGB	–	0.437	0.039	7.01	12.921
Hyper	–	0.320	0.293	52.25	121.604
Hyper	PCA 3 GPU	0.241	0.298	86.80	118.521
Hyper	PCA 3 CPU	0.241	0.378	43.81	112.162
Hyper	PCA 20 GPU	0.479	0.290	86.29	117.204
Hyper	PCA 20 CPU	0.479	0.382	41.33	112.845
Hyper	PCA 50 GPU	0.521	0.295	86.01	122.842
Hyper	PCA 50 CPU	0.521	0.457	36.34	122.721

Table 1: Accuracy and efficiency results of the considered alternatives. The "time" and "energy" columns contain the consumption of the pipeline when processing a single input batch. The "preparation" column shows the percentage of time spent preprocessing the input images and transferring data to the GPU before inference, not including PCA.

hardware accelerators close to the sensors, enabling the delivery of preprocessed data that can be efficiently handled in subsequent stages of an image-processing pipeline.

References

[AJH+16] Jorge Albericio, Patrick Judd, Tayler Hetherington, Tor Aamodt, Natalie Enright Jerger, and Andreas Moshovos. Cnvlutin: Ineffectual-neuron-free deep neural network computing. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 1–13, 2016.

[ARM20] Iñigo Alonso, Luis Riazuelo, and Ana C. Murillo. MiniNet: An efficient semantic segmentation ConvNet for real-time robotic applications. *IEEE Transactions on Robotics (T-RO)*, 2020.

[Dal22] William Dally. On the model of computation: Point. *Commun. ACM*, 65(9):30–32, aug 2022.

[SCYE17] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel Emer. Efficient processing of deep neural networks: A tutorial and survey, 2017.

Acknowledgments

This work was partially supported by MCIN/AEI/10.13039/501100011033 (grants PID2019-105660RB-C21 and MIG-20201006), and by Government of Aragon (T5820R research group).