Universitat Autònoma de Barcelona
Degree in Artificial Intelligence
Supervisor: Vanessa Moreno Font

OCR Implementation in a Jetson Nano

Pol Medina Arévalo

March 23, 2025

ABSTRACT

This project investigates the deployment of optimized Optical Character Recognition (OCR) models on the NVIDIA Jetson Nano T1 for real-time text extraction in industrial settings. We aim to address the critical challenge of balancing accuracy and inference speed on resource-constrained edge devices. Specifically, we explore the adaptation and optimization of deep learning-based OCR architectures for efficient execution on the Jetson Nano. Our methodology involves a multi-stage process: (1) establishing baseline performance on a high-performance server, (2) applying model optimization techniques such as quantization, pruning, and TensorRT acceleration to reduce inference time, and (3) deploying and fine-tuning these optimized models on the Jetson Nano. We will evaluate performance using metrics such as character recognition accuracy and inference latency, comparing the results against unoptimized models. The expected contributions include a benchmark of OCR model performance on the Jetson Nano, demonstrating the feasibility of achieving real-time processing within industrial constraints, and a reproducible methodology for optimizing deep learning models for edge deployment. This research contributes to the advancement of embedded AI for industrial automation, providing insights into the trade-offs between accuracy and efficiency in real-time OCR applications.

1 Problem definition

The primary goal of this project is to implement optical character recognition (OCR) models, which involve converting different types of images into machine-readable text, powered by artificial intelligence (AI) on an NVIDIA Jetson Nano T1 microprocessor. The Jetson Nano is designed specifically for AI applications, and due to its powerful GPU and AI-optimized architecture, is well-suited to this task. This leverages its capability for efficient parallel processing, making it an ideal platform for running OCR tasks and offering a compact solution for edge processing where small form factors and portability are essential.

Within this goal there is a mandatory requirement, which involves using deep learning models to perform OCR while optimizing them for speed and efficiency. The challenge is to ensure that these models not only perform well but also have extremely low inference times. This means the project explores AI model acceleration techniques to reduce inference time while maintaining accuracy. The Jetson Nano's GPU, combined with various optimization methods, will make it possible to achieve highly efficient OCR performance that is suitable for the industrial environment. Further explanations in metrics and specific objectives, as well as optimization techniques will be explained further in section 3.

The specific use case for this project, is based in industrial settings, particularly in manufacturing or warehouse environments where product codes and bar codes are scanned. Codes will consist of printed strings of numbers and letters like the one in image 1. The OCR models will be used to interpret these codes at high speed for further use according to the needs of the processes. In these industries, efficiency is crucial. Thousands of products may need to be processed in a very short amount of time, making it necessary to deploy a system that can handle this task swiftly. The Jetson Nano's processor, thanks to its portability and AI-optimized capabilities, is the perfect choice for this type of application. It enables real-time processing, which is essential in these environments.

1.1 Relevant hardware specifications

For the development of the project two different platforms will be used: a server with available GPUs and the Jetson Nano. The code will be first developed in the cluster's GPUs and it will be then transferred to the Jetson Nano environment with the corresponding finetuning. The specifications of two available GPUs and Jetson are specified in table 1.

It is relevant to understand that NVIDIA Jetson Nano works with JetPack, which is a software development kit (SDK) for NVIDIA Jetson devices. It provides everything needed to develop AI and computer vision applications, including Ubuntu based OS, CUDA, cuDNN and TensorRT. It offers an optimized software stack aimed for edge computing on these devices.

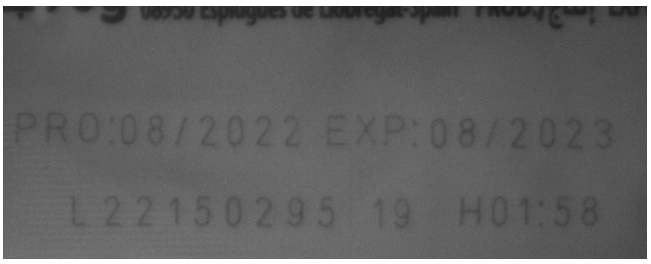

Figure 1: Example of case use images with codes

	Performance	*Cuda Cores*	*Video Memory*	*Power Consumption*
Jetson Nano T1	∼0.5 TFLOPS	128	4GB LPDDR4	5-10W
RTX 2080 Ti	∼13.45 TFLOPS	4,352	11 GB GDDR6	250W
RTX 3090	∼35.6 TFLOPS	10,496	24 GB GDDR6	350W

Table 1: Hardware specifications for available resources

2 Previous Context and State of the Art

In the realm of Optical Character Recognition (OCR), achieving a balance between accuracy and computational efficiency is crucial, particularly when deploying models on resource-constrained devices like the NVIDIA Jetson Nano. While cloud-based OCR solutions leverage large Transformer-based architectures to attain high accuracy, these models pose significant challenges for edge deployment due to their high computational and memory requirements.

2.1 Benchmarks of Existing OCR Models

Recent evaluations highlight the trade-offs between accuracy and inference time among various OCR models, particularly in terms of efficiency for edge computing:

EasyOCR: An open-source model achieving a character accuracy of approximately 85% with an inference time of 0.042 seconds per image. EasyOCR utilizes a CRNN architecture, balancing simplicity and performance. [1]

PaddleOCR: Another open-source solution with a character accuracy of around 90% and an inference time of 0.110 seconds per image. PaddleOCR offers a range of models, including those optimized for mobile devices, demonstrating a focus on efficiency. [2]

Google Cloud Vision API: A cloud-based service offering a character accuracy of about 95% with an inference time of 0.063 seconds per image. While highly accurate, cloud-based APIs introduce latency and are not ideal for real-time edge processing. [3]

Tesseract OCR: A traditional OCR engine, efficient for printed text but struggles with complex layouts and handwritten text. It offers speed but lacks the deep learning-based accuracy of newer models. [4]

These metrics underscore that while cloud-based models like Google Cloud Vision provide high accuracy, they may not be suitable for real-time applications on edge devices due to latency and resource constraints. Additionally, conventional OCR solutions such as Tesseract OCR, while efficient for printed text, struggle with deep learning-based requirements like handwritten recognition and complex text extraction.

2.2 Challenges with Transformer-Based Models on Edge Devices

Modern OCR systems increasingly rely on Transformer-based architectures, which, while highly effective, present significant challenges for deployment on resource-constrained hardware such as the Jetson Nano:

- **High Memory Usage:** Self-attention mechanisms require large memory overhead, making real-time inference difficult on low-power devices. [5]
- **Inference Latency:** Unlike CNN-based models, Transformer-based architectures often introduce longer inference times due to the sequential nature of attention computations. Techniques like distillation are being explored to mitigate this. [6]
- **Computational Demands:** Transformers require substantial parallel processing power, making them inefficient for devices with limited GPU resources. Model quantization and pruning are essential for deploying these models on edge devices. [7]

While cloud-based OCR systems employing Transformer architectures offer high accuracy, their computational demands render them impractical for real-time applications on edge devices. Therefore, developing lightweight OCR models optimized for edge computing—through techniques such as model quantization, pruning, and TensorRT acceleration—presents a viable solution for deploying efficient OCR systems in resource-constrained environments like our industrial use case setting.

Moreover, regarding directly to the Jetson Nano, while there are existing projects handling several computer vision tasks such as object segmentation or human pose estimation, there does not exist a direct approach to accelerated OCR computation in the platform. Works such as *On-device real-time hand gesture recognition with deep learning* [8] show the capability of Jetson Nano for computer vision, but lack the OCR focus. This creates the necessity of creating this implementation in a Jetson device.

3 Objectives and Contributions

The primary objective of this project is to develop and evaluate an optimized Optical Character Recognition (OCR) system for real-time text extraction on the NVIDIA Jetson Nano T1, tailored for industrial use cases. The core contributions will include:

Optimizing AI models for Jetson Nano: To achieve a target inference time of less than 100 milliseconds per image on the Jetson Nano, we will employ techniques such as model quantization (8-bit and mixed-precision), pruning (magnitude-based and structured), and TensorRT acceleration. The baseline model, trained on a server with RTX 3090 would, in a desired way, be optimized to reduce its size by at least 30% while maintaining a character recognition accuracy within 5% of the original model.

Benchmarking the efficiency: We will conduct a comparative analysis of various OCR models (EasyOCR, PaddleOCR, CRNNs, and Transformer-based architectures), evaluating their performance in terms of inference time, character recognition accuracy, and model size. This benchmark will provide insights into the trade-offs between these metrics on both server and Jetson Nano environments.

Adapting the models to the Jetson Nano: The optimized models will be converted to TensorRT format and fine-tuned for efficient execution on the Jetson Nano. This adaptation will involve optimizing layers and model parameters to maximize the GPU utilization of the Jetson Nano.

Evaluation of final performance: A comprehensive evaluation will be conducted, comparing the inference time and character recognition accuracy of the unoptimized models, the optimized models running on the server, and the final models deployed on the Jetson Nano. It is aimed to achieve a minimum speedup of 2x in inference time on the Jetson Nano compared to the unoptimized baseline, while maintaining a character recognition accuracy of at least 85% on the industrial use case dataset.

The main contributions of this work include:

- Developing an optimized OCR system tailored for real-time edge computing on the Jetson Nano, demonstrating the feasibility of achieving high performance within the constraints of resource-limited hardware.
- Providing a detailed benchmark of inference speed, character recognition accuracy, and model size for various OCR models in embedded environments, offering valuable insights for future research and development.
- Demonstrating and evaluating hardware acceleration techniques, specifically TensorRT optimization, for real-time OCR in industrial applications, showcasing the potential for efficient edge deployment.
- Providing a reproducible methodology for optimizing deep learning models for edge deployment, including code and documentation for further similar applications.

4 Methodology and Work Plan

The project will follow a structured methodology to achieve its objectives. The work is divided into several stages:

Environment Setup on the Server: We will establish a high-performance server environment equipped with an NVIDIA RTX 3090 and an NVIDIA RTX 2080 Ti. The software stack will include Ubuntu, CUDA, cuDNN, PyTorch and OpenCV with CUDA support. This environment will facilitate efficient training and initial testing of OCR models.

Image Preprocessing: To enhance OCR performance, we will apply a series of different preprocessing techniques using OpenCV with CUDA acceleration. This can include:

- Noise reduction using Gaussian blur and median filtering.
- Adaptive thresholding (e.g., Otsu's method) to binarize the images and isolate characters.
- Contrast enhancement using histogram equalization and CLAHE (Contrast Limited Adaptive Histogram Equalization).

Model Selection: We will evaluate many OCR models, as long as resource constraints allows us to, including:

- EasyOCR, utilizing a CRNN (Convolutional Recurrent Neural Network) architecture.
- PaddleOCR, exploring their models optimized for mobile and edge devices.
- Custom CRNN architectures, tailored for the specific characteristics of our industrial dataset (alphanumeric codes).
- Transformer-based models (e.g., using the DETR architecture adapted for text recognition), to assess their performance and feasibility on edge devices.

Baseline Model Development: We will use pretrained baseline models on the server using the industrial dataset, which consists of images containing alphanumeric codes similar to the example shown in Figure 1.

Model Optimization: To reduce model size and improve inference speed, we will apply the following techniques:

- Quantization: 8-bit and mixed-precision quantization using TensorFlow Lite and TensorRT.
- Pruning: Magnitude-based and structured pruning to remove redundant weights and connections.
- Knowledge distillation: Transferring knowledge from larger, more accurate models to smaller, faster ones.

Jetson Nano Setup and Model Adaptation: The optimized models will be converted to TensorRT format and deployed on the Jetson Nano. The JetPack SDK will be utilized to configure the Jetson Nano environment.

Performance Evaluation: We will evaluate the performance of the models using the following metrics:

- Character Recognition Accuracy: Measured as the percentage of correctly recognized characters.
- Inference Time: Measured in milliseconds per image, to assess real-time processing capabilities.
- Model Size: Measured in megabytes, to evaluate the efficiency of optimization techniques.

We will compare the performance of the unoptimized models, the optimized models on the server, and the final models on the Jetson Nano.

Throughout the project, iterative refinements will be made based on performance evaluations to further optimize the models. The ultimate goal is to develop an OCR system that meets industrial requirements while operating efficiently on resource-constrained hardware.

4.1 Work plan over weeks

Phase/Task	***Weeks***
Phase 1: Setup & Preparation	**(Weeks 1-4)**
- Server environment setup (CUDA, cuDNN, PyTorch, OpenCV...)	Week 1
- Dataset preparation and augmentation (random rotations, scaling, noise addition)	Week 2
- Implementation of image preprocessing pipeline (Gaussian blur, adaptive thresholding...)	Weeks 3-4
Phase 2: Model Selection & Baseline Development	**(Weeks 5-8)**
- Research and selection of OCR models (EasyOCR, PaddleOCR, CRNNs, transformers...)	Week 5
- Implementation of baseline models on the server	Weeks 6-7
- Evaluation of baseline models (character recognition accuracy, inference time)	Week 8
Phase 3: Model Optimization	**(Weeks 9-12)**
- Implementation of quantization, pruning and knowledge distillation	Weeks 9-10
- Benchmarking optimized models on the server	Week 11
- Selection of best-performing optimized models for Jetson Nano deployment	Week 12
Phase 4: Jetson Nano Deployment & Adaptation	**(Weeks 13-17)**
- Jetson Nano environment setup (JetPack SDK, TensorRT installation)	Week 13
- Model conversion to TensorRT format	Weeks 14
- Model adaptation and optimization for Jetson Nano	Weeks 15-16
- Initial testing and debugging on Jetson Nano	Week 17
Phase 5: Performance Evaluation & Finalization	**(Weeks 18-20)**
- Comprehensive performance evaluation on Jetson Nano (accuracy, inference time, model size)	Weeks 18-19
- Comparison of Jetson Nano performance with server baseline	Week 20
- Writing final report and documentation	Week 20

References

[1] EasyOCR. (n.d.). https://www.jaided.ai/easyocr/

[2] PaddleOCR. (n.d.). https://github.com/PaddlePaddle/PaddleOCR

[3] Google Cloud Vision API. (n.d.). https://cloud.google.com/vision/docs/ocr

[4] Smith, R. (2007). An overview of the Tesseract OCR engine. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 629-633.

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *arXiv preprint arXiv:1706.03762*.

[6] Tang, R., Lu, Y., Liu, L., Mou, L., Vechtomova, O., & Lin, J. (2019). Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136*.

[7] Zafrir, O., Boudoukh, G., Izsak, E., & Wasserblat, E. (2019). Q8bert: Quantized 8bit bert for natural language understanding. *arXiv preprint arXiv:1910.06188*.

[8] Krekhov, A., & Alahi, A. (2020). On-device real-time hand gesture recognition with deep learning. *arXiv preprint arXiv:2111.00038*.

[9] Technical City. (2019). GeForce RTX 2080 Ti vs Jetson Nano GPU.

[10] Seul-Ki Yeom, Tae-Ho Kim (2024). UniForm: A Reuse Attention Mechanism for Efficient Transformers on Resource-Constrained Edge Devices. arXiv preprint arXiv:2412.02344.

[11] AIMultiple. (2024). OCR Accuracy: The Complete Guide to OCR Accuracy in 2024.

[12] NVIDIA. (2024). JetPack SDK 5.1.

[13] Expert Beacon. (2024). OCR Accuracy: Factors Affecting OCR Performance.