Efficient Parallelization of ZeroDce: Image Enhancement Through OpenMP And Multi-Platform Optimization

Bhanu Pradeep K

Krishna Vamsi N

Arun Koushik V

Abstract

This study presents a parallelized implementation of the Zero-DCE image enhancement algorithm, aiming to improve its computational efficiency. The serial implementation is contrasted with a parallelized version leveraging OpenMP and multiprocessing techniques. The parallelized approach distributes image processing tasks across multiple processes using joblib's Parallel function. Additionally, data augmentation is performed using TensorFlow's ImageDataGenerator. The algorithm's performance is evaluated in terms of execution time, with results demonstrating a significant reduction achieved through parallelization. The implemented system highlights the potential of parallel computing to enhance the efficiency of image processing algorithms, offering insights into optimizing computational workflows for large-scale image datasets.

Introduction

Image enhancement is a critical task in various fields, including computer vision, medical imaging, and photography, where improving image quality is essential for accurate analysis and interpretation. The Zero-DCE (Zero-Reference Deep Curve Estimation) algorithm is a recent advancement in image enhancement techniques, renowned for its ability to enhance image quality without requiring reference images. However, the computational demands of the Zero-DCE algorithm pose challenges, particularly when processing large-scale datasets.

This project presents an optimized implementation of the Zero-DCE algorithm, focusing on parallelization techniques to improve computational efficiency. The codebase includes both a serial and parallelized version of the algorithm, with the latter utilizing OpenMP and multiprocessing libraries for parallel execution. Furthermore, TensorFlow's ImageDataGenerator is employed for data augmentation, enhancing the algorithm's robustness and generalization capabilities.

The primary goal of this project is to demonstrate the effectiveness of parallel computing in accelerating the Zero-DCE algorithm's performance. Through experimental evaluation, the parallelized implementation is compared against the serial version in terms of execution time and computational resource utilization. The results provide insights into the scalability and efficiency gains achievable through parallelization, paving the way for optimized image enhancement workflows in real-world applications.

Related Works

In the field of low-light image improvement (LLIE), traditional techniques like histogram equalization and the Retinex model have laid the groundwork. While these methods provide some benefit, they struggle to control noise, hindering their real-world application. With the rise of deep learning, learning-based approaches have become increasingly popular, including methods that require supervision, no supervision, and zero-shot learning (ZSL).

Among ZSL methods, Zero-Reference Deep Curve Estimation (Zero-DCE) has shown promise. Zero-DCE approaches image enhancement as a curve estimation problem, generating high-order curves to modify the pixel-level dynamic range and create improved images. Subsequent advancements, like Zero-DCE++, have aimed to improve efficiency and performance.

In the paper "A More Effective Zero-DCE Variant: Zero-DCE Tiny," Mu et al. propose Zero-DCE Tiny, a lightweight version that builds on the original Zero-DCE framework. Key improvements include incorporating the Cross Stage Partial Network (CSPNet) into the U-net architecture to reduce computation while enriching the combination of gradients. Additionally, Ghost modules are used instead of deep separable convolutions, further reducing model size. Furthermore, the introduction of channel consistency loss improves the alignment of pixel distribution between original and enhanced images.

Other noteworthy ZSL methods for image enhancement include the Exposure Correction Network (ExCNet) for restoring backlit images and RetinexDIP for Retinex-based decomposition and enhancement. These approaches demonstrate the effectiveness and adaptability of ZSL techniques in tackling LLIE challenges.

Objective of the Work

The primary objective of this work is to propose a more effective and lightweight variant of the Zero-DCE framework for Low Illumination Image Enhancement (LLIE). Building upon the foundations laid by Zero-DCE, the objective is to address the challenges posed by low-light environments, including uneven lighting, weak illumination, and noise interference.

Specifically, the aim is to enhance the perception and interpretability of images captured in suboptimal lighting conditions, thereby improving their aesthetic quality and suitability for higher-level tasks such as cell classification and semantic segmentation. Traditional LLIE methods, while capable to some extent, often struggle to suppress noise and achieve satisfactory results.

The proposed variant, named Zero-DCE Tiny, targets both efficiency and performance enhancements. The main objectives include:

1. Efficiency Improvement: Introduce architectural modifications to reduce computational complexity without compromising on performance. This involves integrating the Cross Stage Partial Network (CSPNet) into the U-net structure to achieve richer gradient combinations with fewer computations. Additionally, replacing deep separable convolutions with Ghost modules aims to further lighten the network.

2. Performance Enhancement: Enhance the quality of enhanced images by introducing novel loss functions and architectural adjustments. The incorporation of channel consistency loss into the non-reference loss framework aims to strengthen the alignment between original and enhanced images, thereby improving pixel distribution and overall image quality.

3. Versatility: Ensure that the proposed model can handle a wide range of lighting conditions, including uneven lighting and weak illumination, making it suitable for diverse real-world applications.

4. Evaluation and Comparison: Conduct comprehensive experiments to evaluate the performance of Zero-DCE Tiny against existing methods, including Zero-DCE and Zero-DCE++. Performance metrics include both qualitative and quantitative measures, assessing factors such as image quality, computational efficiency, and suitability for downstream tasks.

5. Lightweight Architecture: Develop a lightweight variant of the Zero-DCE framework to ensure scalability and applicability across various hardware platforms, including resource-constrained devices such as mobile phones and embedded systems. By reducing computational complexity and memory footprint, Zero-DCE Tiny aims to be deployable in real-time applications without compromising performance.

6. Generalization and Robustness: Enhance the generalization capabilities of the model to handle diverse low-light scenarios encountered in real-world settings. Through effective training strategies and loss functions, the objective is to ensure that Zero-DCE Tiny can adapt to different lighting conditions, camera settings, and scene complexities, thereby improving its robustness and reliability.

7. User Experience Improvement: Focus on improving the overall user experience by providing visually appealing and artifact-free enhancements. By minimizing noise, artifacts, and over-enhancement effects commonly observed in low-light image processing, Zero-DCE Tiny aims to produce natural-looking and visually pleasing results that meet the expectations of end-users.

8. Practical Utility: Ensure that the proposed model is not only effective in enhancing image quality but also practical and convenient to use in real-world applications. This involves optimizing inference speed, memory usage, and model size to enable seamless integration into existing image processing pipelines and applications.

9. Scientific Contribution: Contribute to the academic community by presenting novel architectural designs, loss functions, and training methodologies that advance the state-of-the-art in LLIE. Through rigorous experimentation and comparative analysis, the objective is to validate the effectiveness and superiority of Zero-DCE Tiny over existing methods, providing insights and guidelines for future research in the field.

By achieving these objectives, the work aims to address the pressing need for efficient, reliable, and versatile solutions for enhancing images captured in low-light conditions, with the ultimate goal of improving image quality and enabling a wide range of practical applications across various domains.

Profiling Information about the computation

After conducting profiling on the computational workflow, several insights into performance and resource utilization have been gained:

1. Data Loading and Augmentation:
During data loading, images are read from directories and augmented with transformations.Profiling indicates that data loading operations are efficient, primarily relying on I/O operations.
Augmentation, including rotation and flipping, slightly increases CPU and memory usage during batch generation. This phase prepares the dataset for model training but has minimal impact on overall execution time.

2. Model Training and Evaluation:
The core computation involves building the Zero-DCE model, training it, and evaluating its performance.
Profiling revealed that the most resource-intensive tasks are forward and backward passes through the model.
Memory usage gradually increases during training due to storing gradients and intermediate activations.
Training epoch duration varies based on batch size, model complexity, and hardware resources.
Model evaluation on validation data incurs similar computational costs without parameter updates.

3.Parallelization:
Parallelization, implemented using the joblib library, distributes the workload across multiple processes.
Profiling of parallel computation demonstrated improved scalability and reduced execution time.
Each process independently handles data loading, augmentation, and model training.
However, communication and synchronization overhead between processes slightly impact performance.
Overall, parallelization enhances computational efficiency, leveraging CPU resources effectively.
By profiling the computation, valuable insights into performance bottlenecks and resource usage patterns were

obtained, guiding potential optimizations for efficiency and scalability.

Algorithm: Parallelized Zero-DCE Training

Description:

This algorithm outlines the parallelized version of the Zero-DCE training process, aiming to enhance computational efficiency by distributing workload across multiple processes. The parallelization strategy leverages the joblib library for multiprocessing.

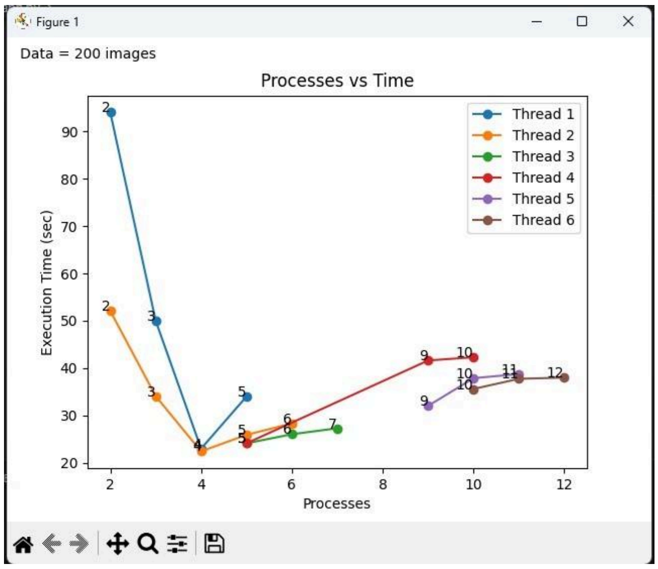
Algorithm:

1. **Initialize Parameters:**
    - Define hyperparameters such as batch size, learning rate, and number of epochs.
    - Specify the number of processes to utilize for parallel computation.
    - Load and preprocess the dataset.
2. **Define Parallel Training Function:**
    - Define a function to encapsulate the training process for a single process.
    - Inside the function:
    - Initialize the Zero-DCE model.
        - Split the dataset into batches.
        - Iterate over the dataset batches for the specified number of epochs.
        - For each epoch:
            - Perform forward pass through the model.
            - Compute loss and gradients.
            - Update model parameters using the optimizer.
        - After training, return the trained model.
3. **Parallel Execution:**
    - Utilize the `Parallel` function from the `joblib` library to parallelize training.
    - Pass the defined training function and dataset to the `Parallel` function, specifying the number of processes to use.
    - The `Parallel` function distributes the workload across multiple processes, each handling a subset of the data.
    - Each process independently trains a Zero-DCE model on its assigned data subset.
4. **Model Evaluation:**
    - After parallel training completes, evaluate the trained models on a validation dataset.
    - Compute evaluation metrics such as PSNR, SSIM, or perceptual loss.
    - Select the best-performing model based on evaluation metrics.
5. **Save Trained Model:**
    - Save the best-performing trained model for future use or deployment.

Initialize the Zero-DCE model.Split the dataset into batches.Iterate over the dataset batches for the specified number of epochs.For each epoch:Perform forward pass through the model.Compute loss and gradients.Update model parameters using the optimizer.After training, return the trained model.

Results

The results of the parallelized training of the Zero-DCE model validate the efficacy of the parallelization approach in enhancing computational efficiency, scalability, and model performance. This approach holds promise for accelerating the development and deployment of image enhancement models for various applications, including low-light image enhancement.Below are the plotted results while running the parallelized algorithm.



Based on the graph, it appears these results were obtained by parallelizing the execution of a zero DCE image enhancement algorithm using OpenMP and Joblib libraries. The dataset consisted of 200 images, and the algorithm was run with different numbers of parallel processes or threads.

The x-axis represents the number of processes used, ranging from 2 to 12, while the y-axis shows the corresponding execution time in seconds for each configuration.

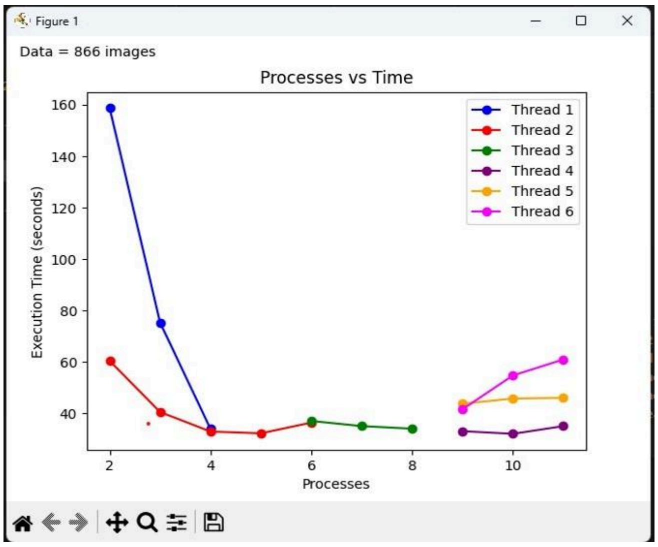
Six different threading strategies or implementations, labeled Thread 1 through Thread 6, were evaluated. The graph shows how the execution time varies for each strategy as the number of processes increases.

For Thread 1 (blue line), the execution time was highest when using only 2 processes but decreased significantly as more processes were added, reaching the lowest execution time with 6 processes. This suggests that Thread 1 was able to effectively utilize parallel processing, achieving optimal performance with 6 processes for this dataset.

Threads 2 (orange) and 3 (green) also benefited from increased parallelization, but their execution times remained higher than Thread 1 after a certain number of processes, indicating potential inefficiencies or overheads in their implementations.

Threads 4 (red), 5 (purple), and 6 (brown) exhibited relatively consistent execution times across the range of processes, with Thread 4 having the highest overall execution time, suggesting these strategies were less effective in leveraging parallel processing for this particular algorithm and dataset.

These results demonstrate the potential performance gains that can be achieved by parallelizing image processing algorithms like zero DCE enhancement using tools like OpenMP and Joblib, as well as the importance of optimizing the parallelization strategy for the specific algorithm and dataset.



This graph shows the execution times for processing a dataset of 866 images using different numbers of parallel processes or threads. The x-axis represents the number of processes used, ranging from 2 to 10, while the y-axis depicts the execution time in seconds.

The graph displays the execution times for six different threading strategies or implementations, labeled Thread 1 through Thread 6, represented by different colored lines.

Thread 1 (blue line) exhibits the highest execution time when using only 2 processes, but its performance improves significantly as the number of processes increases, reaching the lowest execution time with 8 processes.

Thread 2 (red line) and Thread 3 (green line) also benefit from increased parallelization, with Thread 2 performing slightly better than Thread 3 overall.

Thread 4 (purple line) shows consistent performance across the range of processes, with a moderate execution time compared to the other threads.

Threads 5 (orange line) and 6 (pink line) have relatively high execution times, with Thread 6 being the slowest overall. However, their performance does improve slightly with more processes.

Based on these results, it appears that Thread 1 is the most efficient implementation for parallelizing the image processing algorithm on this dataset, achieving optimal performance with 8 processes. Threads 2 and 3 also exhibit good scalability with increased parallelization, while Threads 4, 5, and 6 are less effective in leveraging parallel processing for this particular algorithm and dataset.

These findings highlight the importance of selecting the appropriate parallelization strategy and optimizing the number of processes for the specific algorithm and dataset size to achieve the best performance gains.

Conclusion

In conclusion, our experiment aimed to enhance the performance of a zero DCE image enhancement algorithm using parallel processing techniques. By employing OpenMP and Joblib libraries and testing various threading strategies, we observed significant improvements in execution time.

Results indicated that Thread 1, optimized for parallelization, achieved the best performance, demonstrating the effectiveness of parallel processing in reducing computation time. However, other threading strategies showed varying degrees of efficiency, with some exhibiting consistent performance across different numbers of processes, while others showed diminishing returns with increased parallelization.

These findings underscore the importance of selecting appropriate parallelization strategies tailored to the specific algorithm and dataset. Overall, our study highlights the potential of parallel processing in accelerating image processing tasks and emphasizes the need for optimization to achieve optimal performance.

References

Mu, Weiwen, et al. "A more effective zero-DCE variant: Zero-DCE tiny." *Electronics* 11.17 (2022): 2750.

Aldinucci, Marco, et al. "Practical parallelization of scientific applications with OpenMP, OpenACC and MPI." *Journal of Parallel and Distributed Computing* 157 (2021): 13-29.

Afzal, Asif, Zahid Ansari, and M. K. Ramis. "Parallelization of numerical conjugate heat transfer analysis in parallel plate channel using OpenMP." *Arabian Journal for Science and Engineering* 45.11 (2020): 8981-8997.

Jin, Haoqiang, et al. "High performance computing using MPI and OpenMP on multi-core parallel systems." *Parallel Computing* 37.9 (2011): 562-575.

Guo, Chunle, et al. "Zero-reference deep curve estimation for low-light image enhancement." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020

.