PARALLEL IMAGE FILTERING USING INDEPENDENT AND DEPENDENT FILTERS

Vishwa Kumaresh - AI&DS A (55)
Rhea Pandita - AI&DS B (100)

Github Link

PROJECT SUMMARY

The aim of our project is to parallelize image filtering operations with MPI to optimize processing time for efficient image manipulation, evaluating dependent and independent filter applications. By comparing the serial and parallel runtimes for both filters, we quantify the improvements achievable through parallelization.

BACKGROUND

Traditional image filtering processes often suffer from long processing times, hindering real-time applications and scalability.

This project addresses the growing demand for efficient image processing solutions by leveraging the parallel computing library MPI to significantly improve filtering performance and scalability.

OBJECTIVES

- Implement parallel processing techniques using MPI for image filtering tasks.
- Applying dependent and independent filters to images in parallel.
- Quantify and compare the runtime performance between serial and parallel implementations for dependent and independent filters.

SCOPE AND APPLICATIONS

The project primarily focuses on implementing parallel processing techniques using MPI specifically for image filtering tasks. Both dependent and independent filters will be considered for parallelization to provide a comprehensive comparison of performance gains. Evaluation will primarily concentrate on runtime performance metrics such as speedup and efficiency.

Implementation of parallel image filtering techniques can be extended to other image processing tasks, for example, edge detection, noise reduction and feature extraction. The performance insights gained from this project can be used in further applications of parallel algorithms for image-related tasks.

DATASET DESCRIPTION

A model like this one will typically require a large-scale dataset with atleast 50,000 images. The Fashion Product Images dataset was downloaded from Kaggle, containing 7000 training images. These images were duplicated so that dataset contains approx 57,000 images.

ASPECTS OF PARALLELISM

DATA

Using independent filters to process each pixel independently

TASK

Using dependent filters that rely on info from neighboring pixels

ARCHITECTURE

SIMD: All processing units execute the same set of instruction. Each processing unit operates on either different pixels from the same image or different images altogether.

APPROACH

Independent Filters :
Split the images into sub-parts on which the filter will be applied. These sub-parts are fed into different processors, operated on and the outputs are re-combined to give the filtered image.

Dependent Filters :
Each image is fed into a different processor and operated on.

IMPLEMENTATION

The code for this implementation has 8 important functions :

1. images_from_directory_with_filter ()
2. serial_code()
3. images_from_directory_with_filter_independent_mpi()
4. parallel_independent_code()
5. images_from_directory_with_filter_dependent_mpi()
6. parallel_dependent_code()
7. run_independent_filter()
8. run_dependent_filter()

CHALLENGES

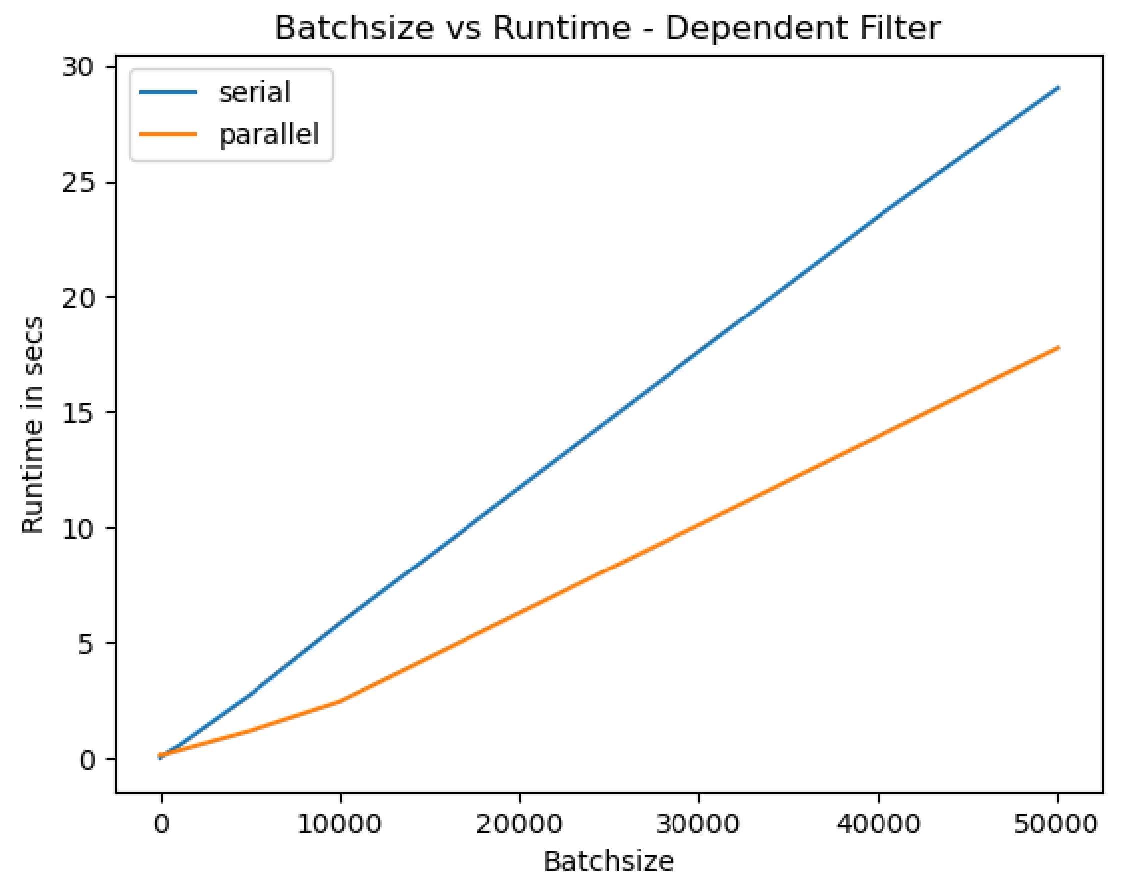
1. **Data Dependency Management:** Ensuring correct data dependencies between parallel tasks for dependent filters.
2. **Load Balancing:** Achieving optimal distribution of computational load across MPI processes.
3. **Communication Overhead:** Minimizing overhead associated with inter-process communication in MPI.

RESULTS

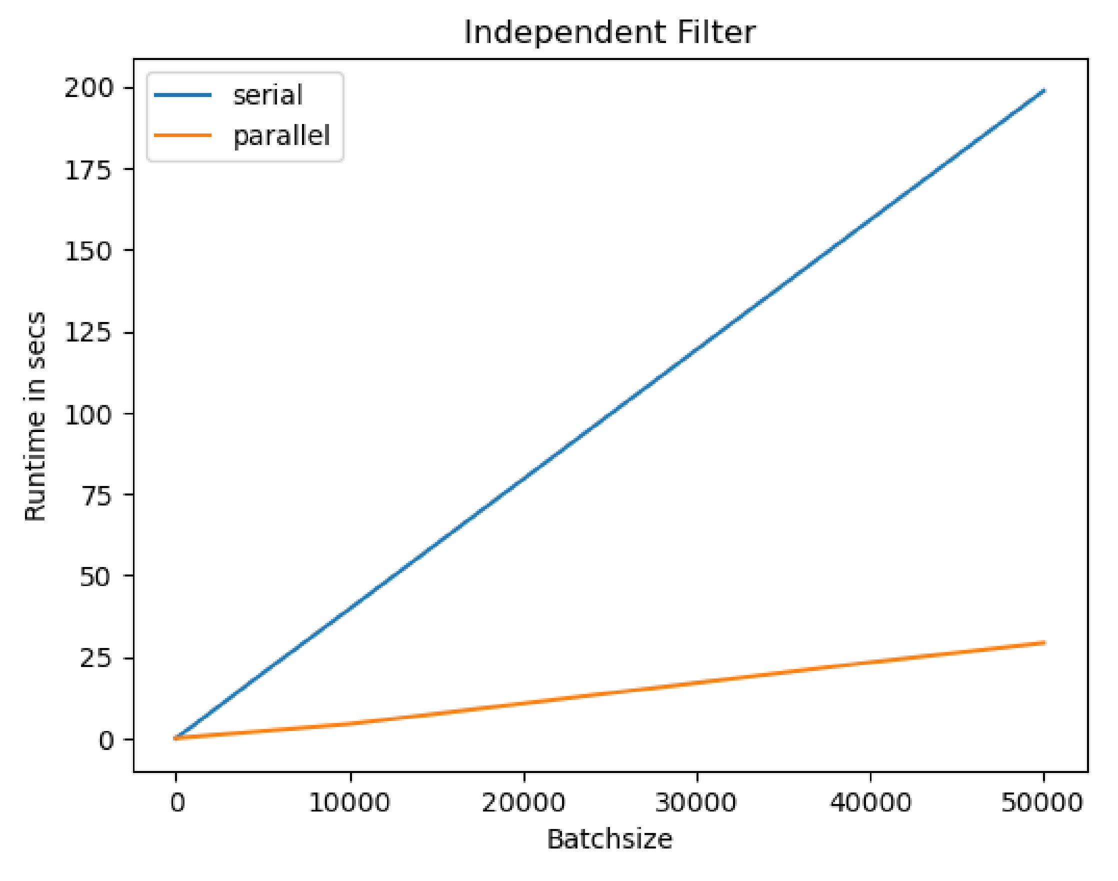
The following readings have been taken after applying both independent and dependent filters on images. Image sizes 32x32, 64x64, 128x128 and 224x224 have been used with batch sizes 1,10,50,100,500,1000,5000,10000,40000 and 50000. (Dataset has approximately 57000 images).

Difference between Independent and Dependent Filter on 224x224 size images with various batch sizes

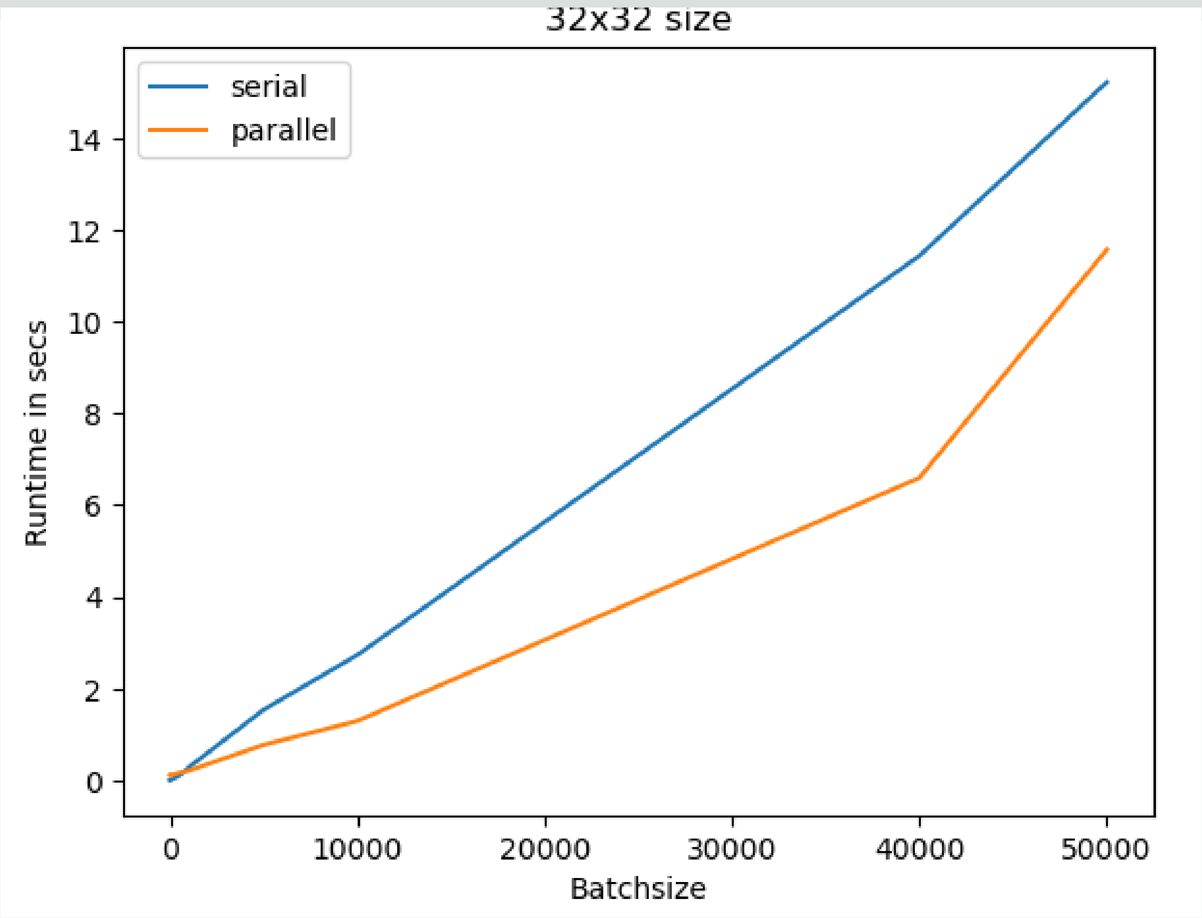
	batchsize	serial	parallel	efficiency	speedup
0	1	0.004539	0.115256	0.39	0.039385
1	10	0.010009	0.106761	0.94	0.093751
2	50	0.035229	0.128810	2.73	0.273496
3	100	0.065904	0.118222	5.57	0.557461
4	500	0.284792	0.205095	13.89	1.388586
5	1000	0.506872	0.310721	16.31	1.631277
6	5000	2.723590	1.164900	23.38	2.338046
7	10000	5.810010	2.429210	23.92	2.391728
8	40000	23.467000	13.930500	16.85	1.684577
9	50000	29.023500	17.752300	16.35	1.634915



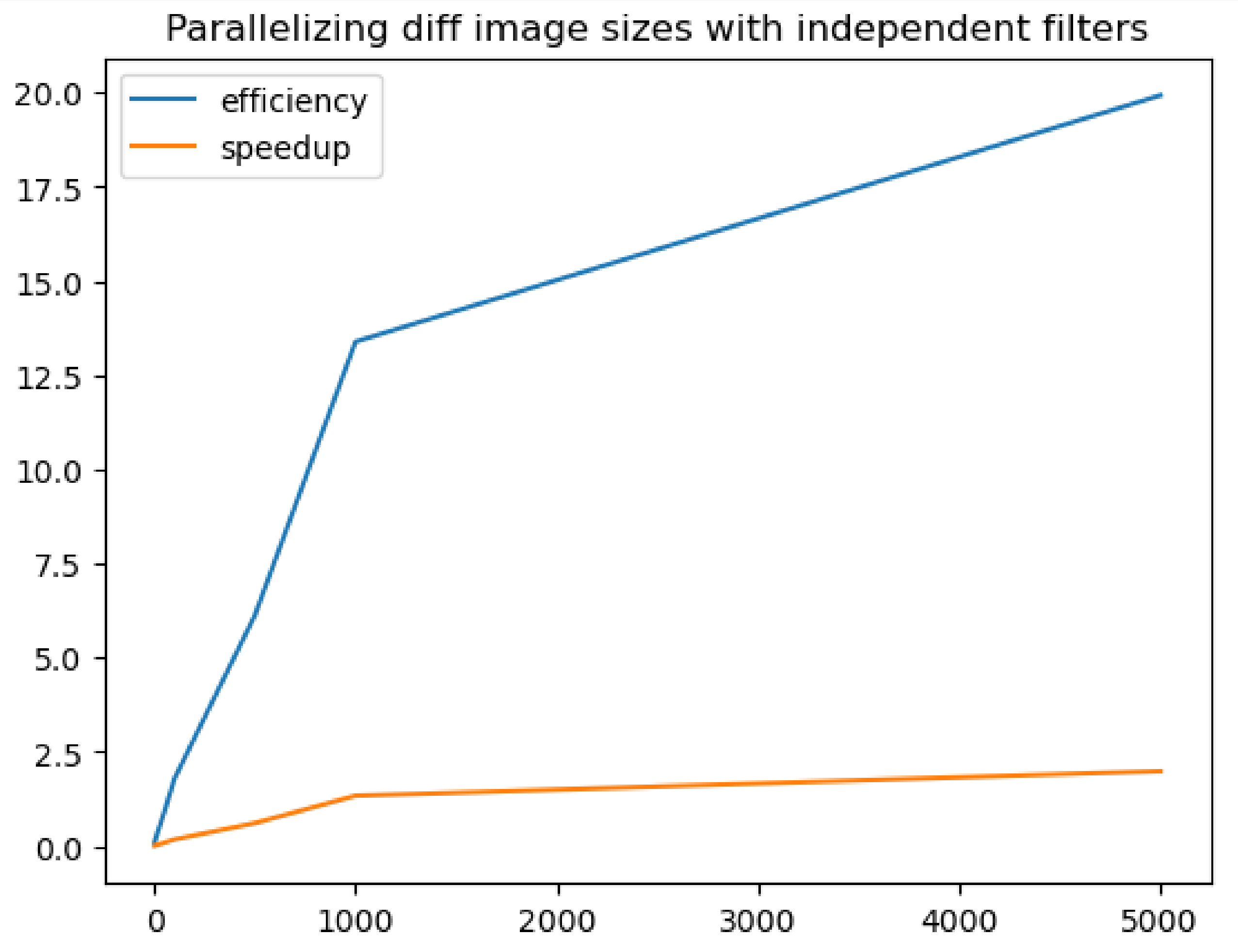
	batchsize	serial	parallel	efficiency	speedup
0	1	0.005271	0.107512	0.49	0.049027
1	10	0.040253	0.099864	4.03	0.403078
2	50	0.198049	0.117462	16.86	1.686069
3	100	0.419870	0.152665	27.50	2.750270
4	500	2.018640	0.312031	64.69	6.469357
5	1000	3.933940	0.516324	76.19	7.619131
6	5000	19.977900	2.241690	89.12	8.911982
7	10000	39.689000	4.405280	90.09	9.009416
8	40000	159.037000	23.322900	68.19	6.818920
9	50000	198.665000	29.254900	67.91	6.790828

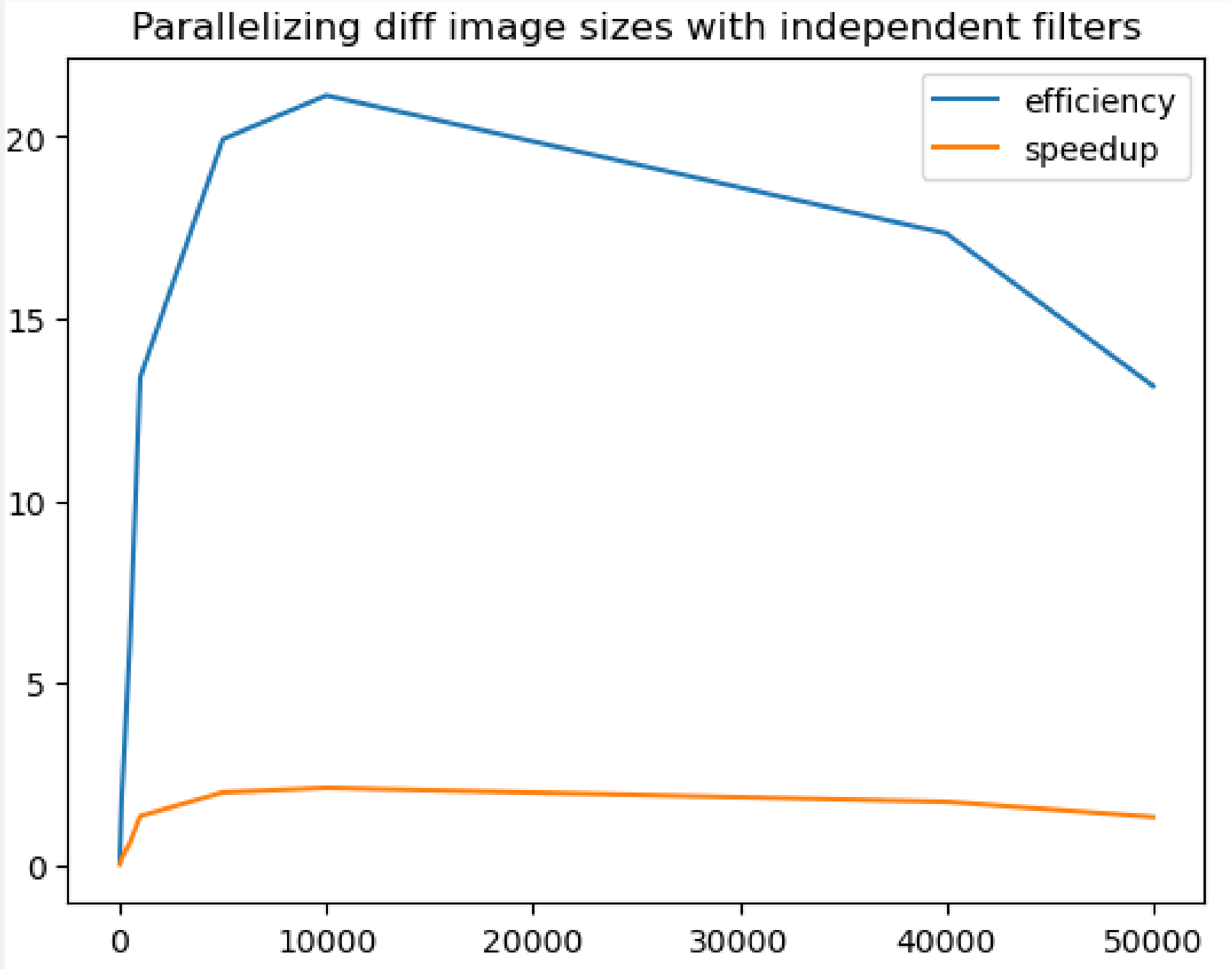


32X32

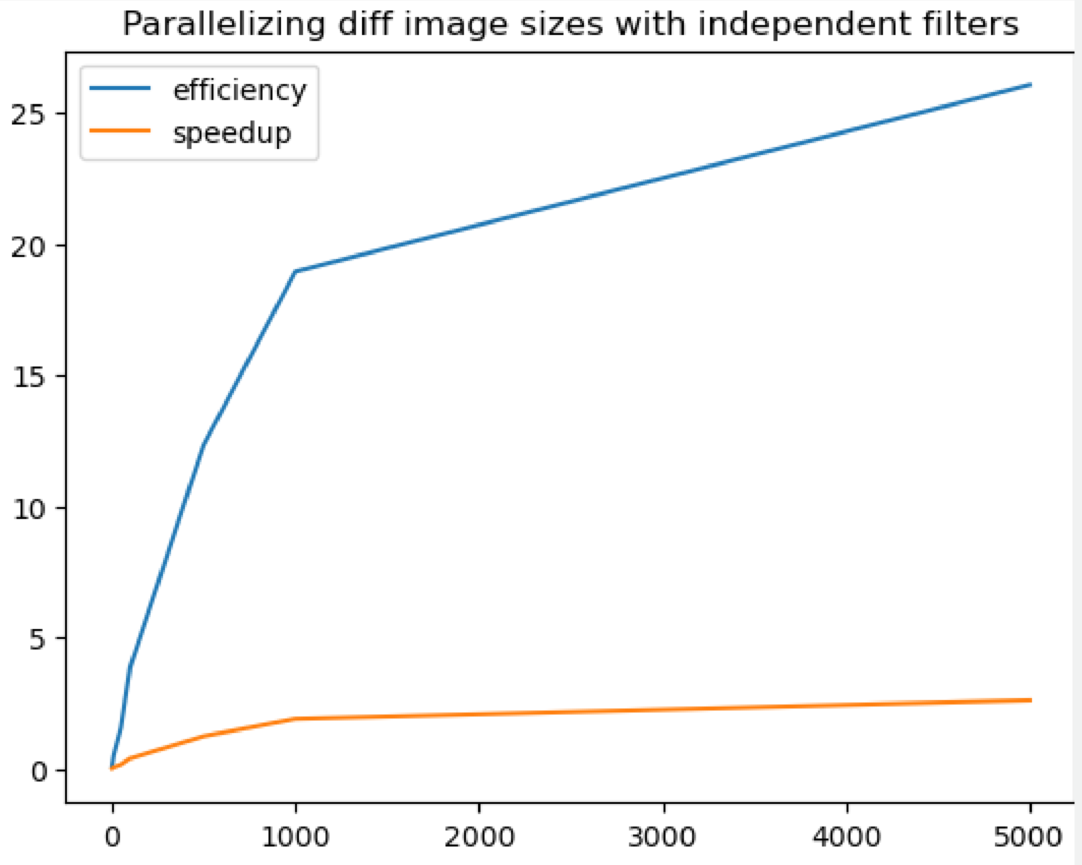


	batchsize	serial	parallel	efficiency	speedup
0	1	0.001007	0.105820	0.10	0.009515
1	10	0.002562	0.097426	0.26	0.026297
2	50	0.010066	0.110478	0.91	0.091113
3	100	0.019955	0.111325	1.79	0.179250
4	500	0.099981	0.163197	6.13	0.612640
5	1000	0.279154	0.208404	13.39	1.339485
6	5000	1.538430	0.771873	19.93	1.993113
7	10000	2.730920	1.292460	21.13	2.112963
8	40000	11.435900	6.594730	17.34	1.734097
9	50000	15.220800	11.571700	13.15	1.315347

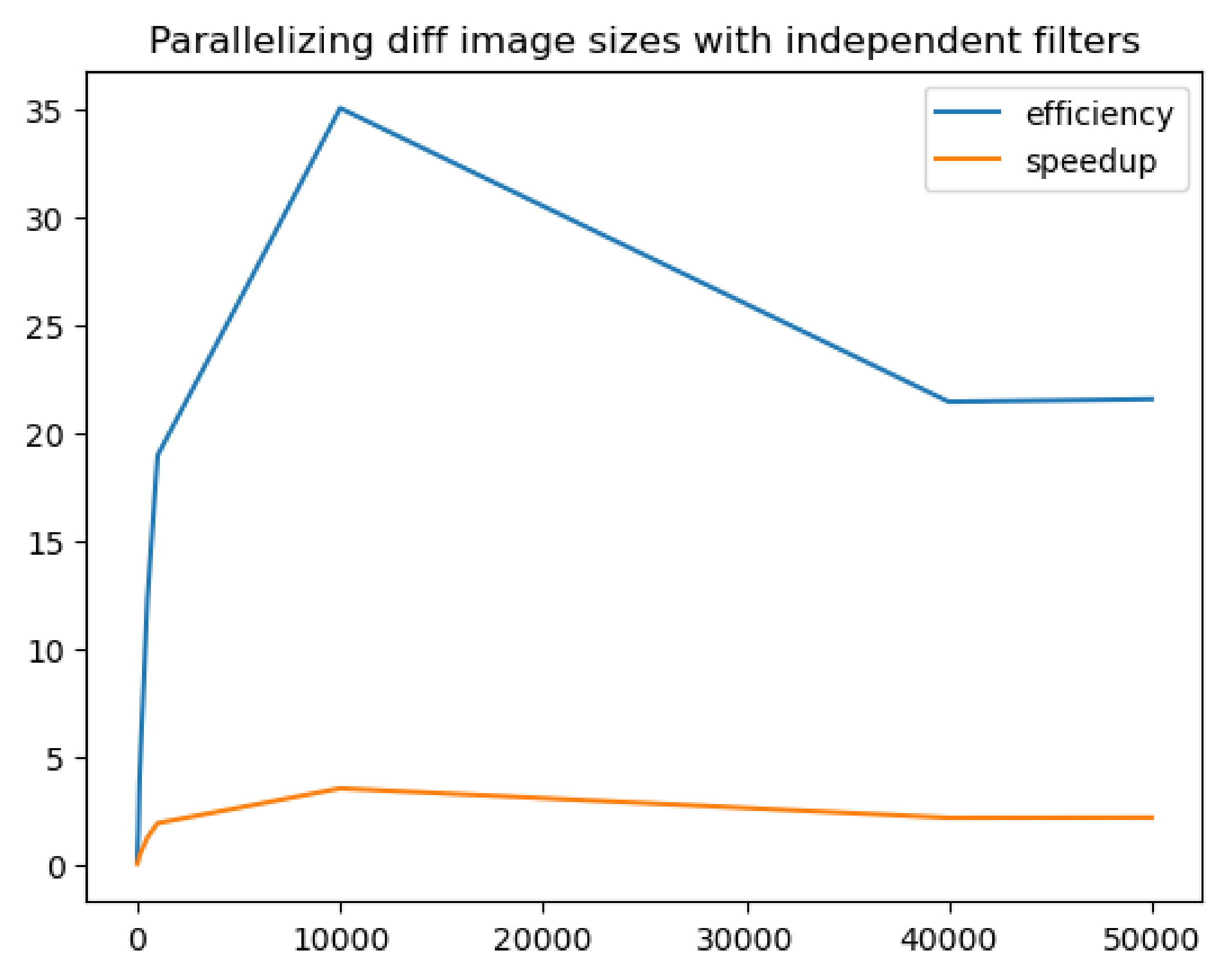


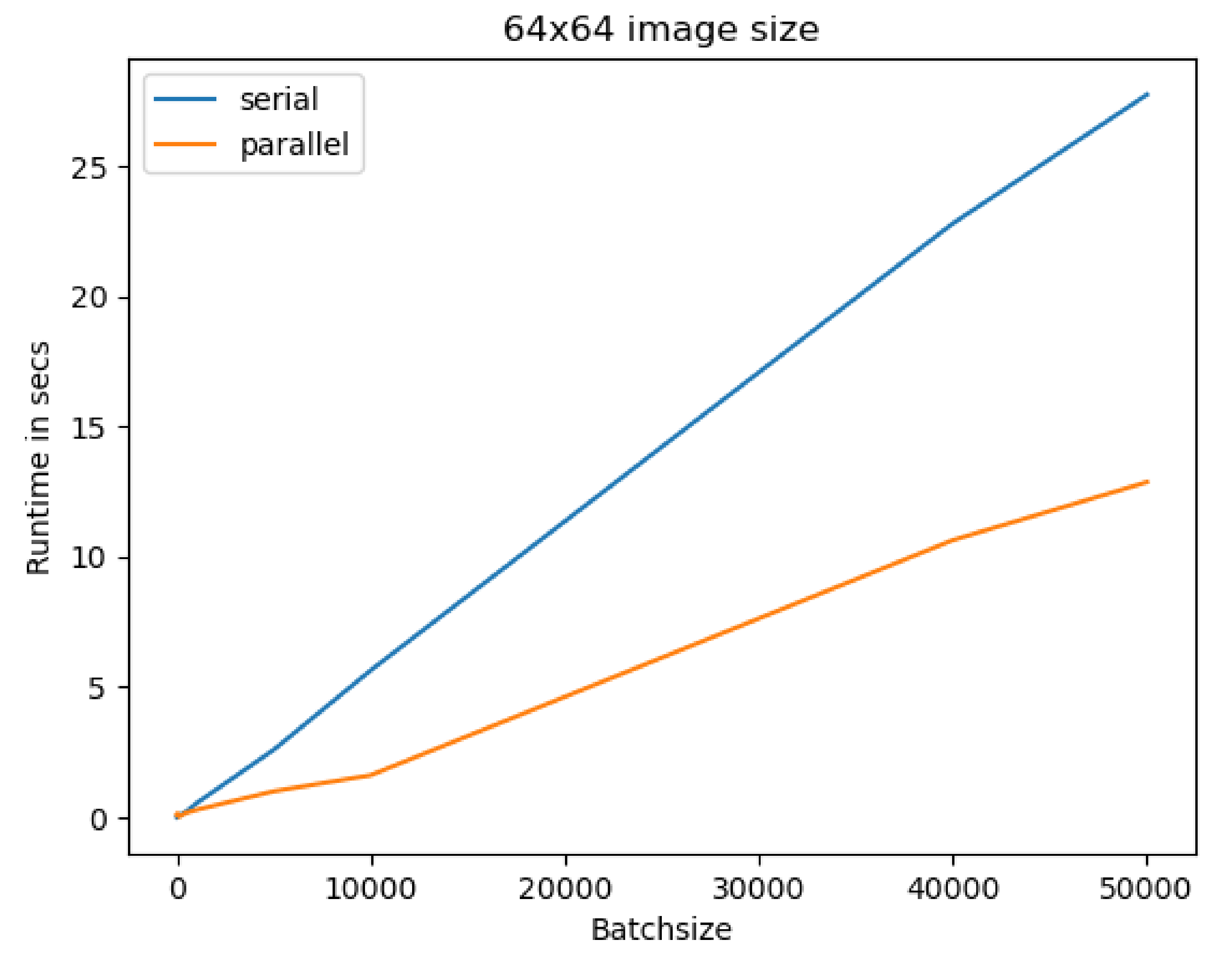


64X64

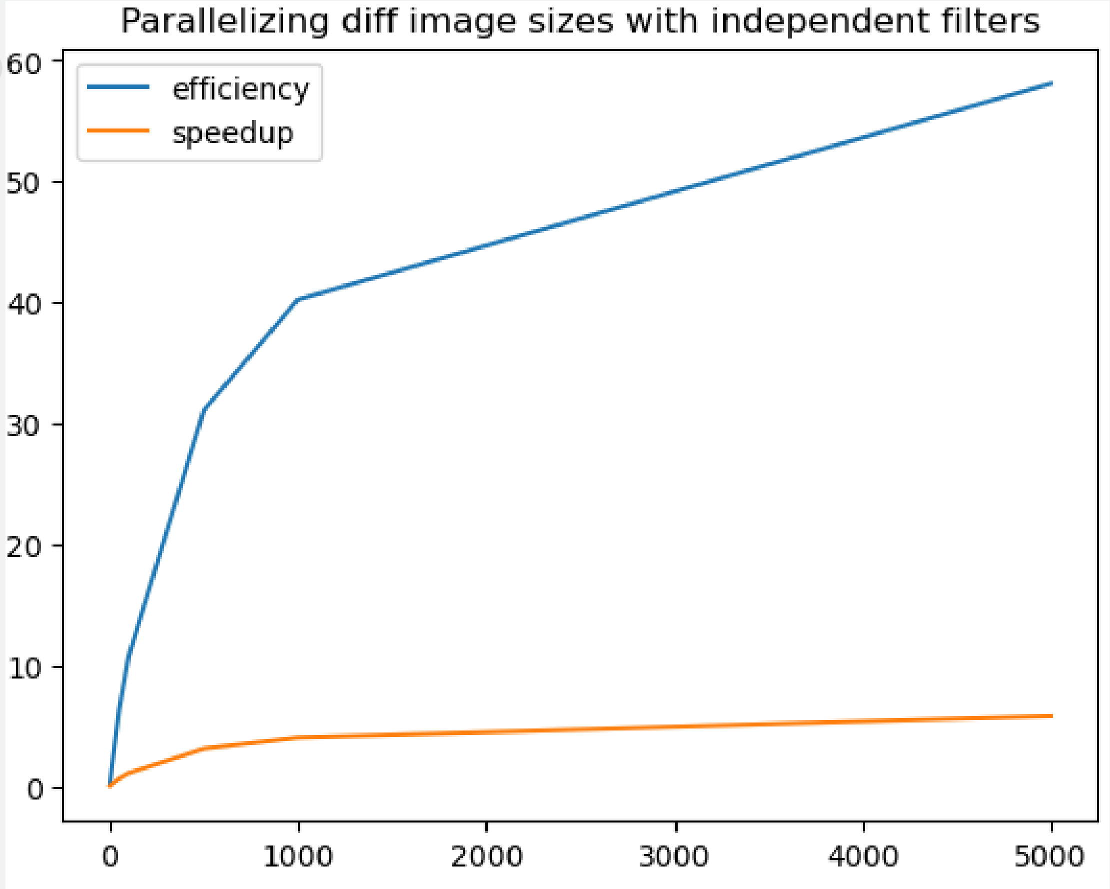


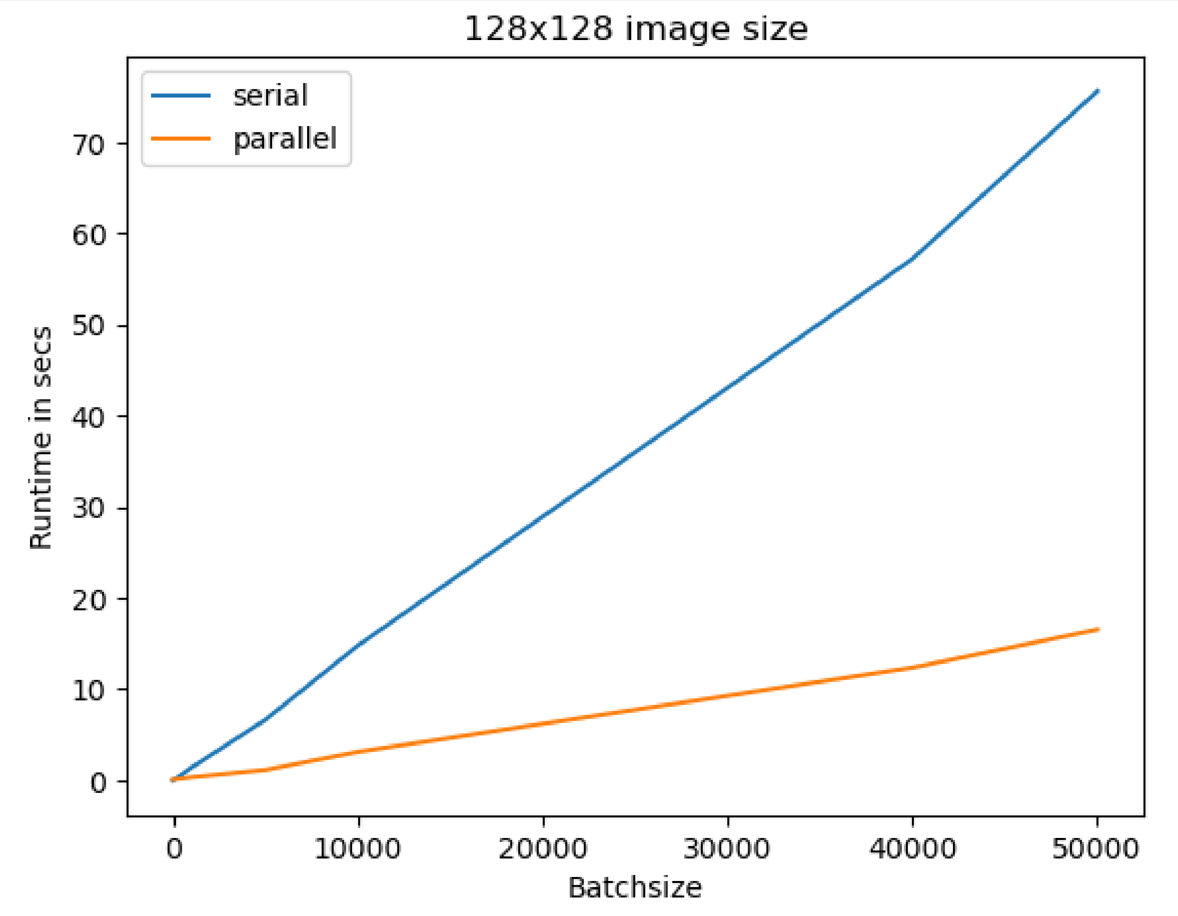
	batchsize	serial	parallel	efficiency	speedup
0	1	0.000994	0.095619	0.10	0.010395
1	10	0.004870	0.097408	0.50	0.050000
2	50	0.022433	0.144760	1.55	0.154967
3	100	0.043708	0.113140	3.86	0.386319
4	500	0.215665	0.174854	12.33	1.233400
5	1000	0.521361	0.275101	18.95	1.895162
6	5000	2.583400	0.991048	26.07	2.606735
7	10000	5.633930	1.607620	35.05	3.504516
8	40000	22.786000	10.632400	21.43	2.143072
9	50000	27.724800	12.858100	21.56	2.156213

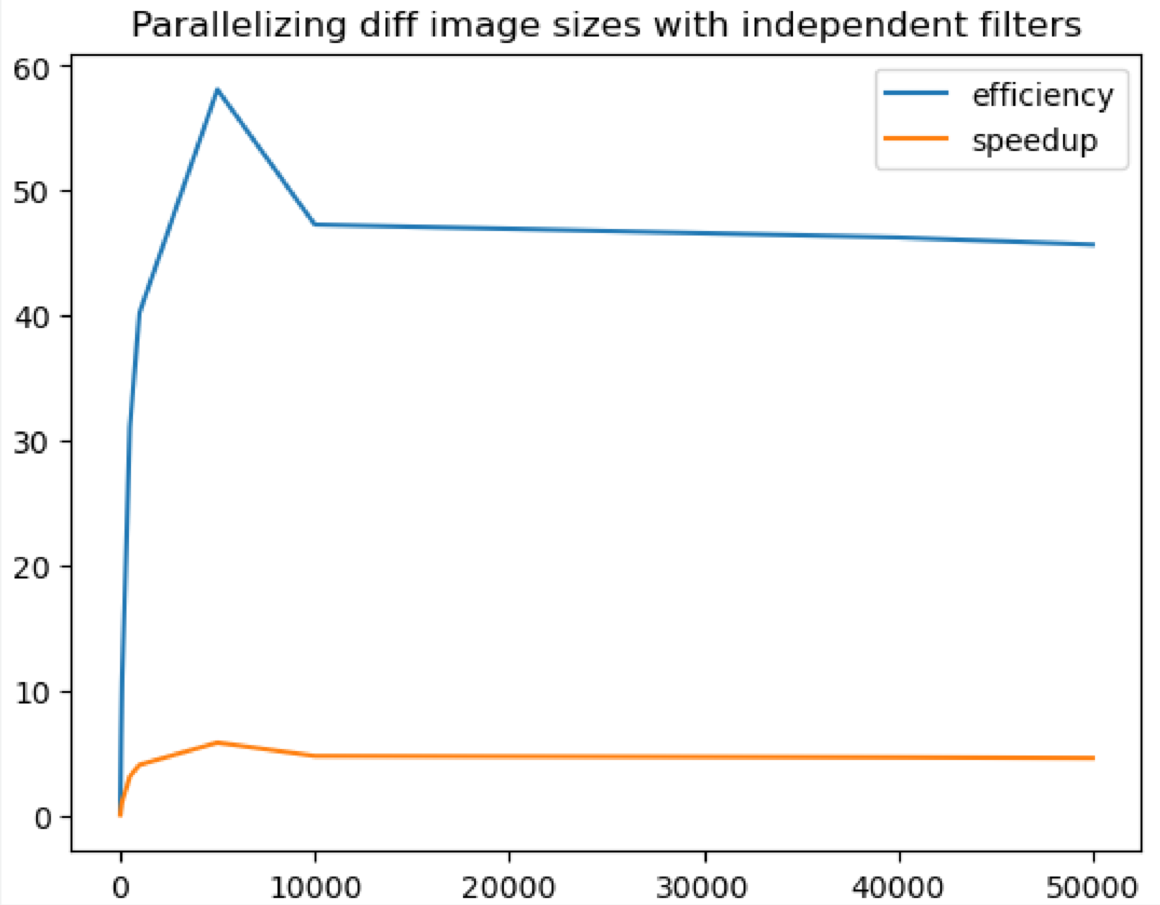




128X128

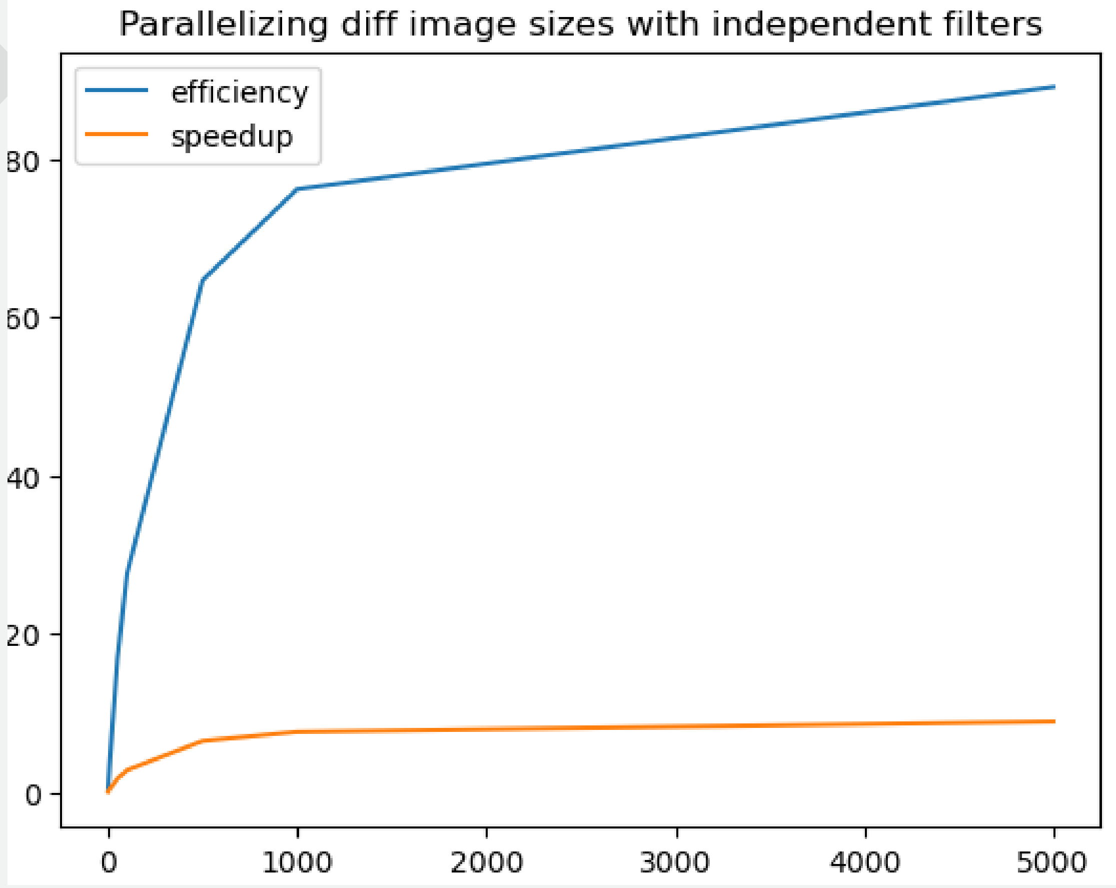


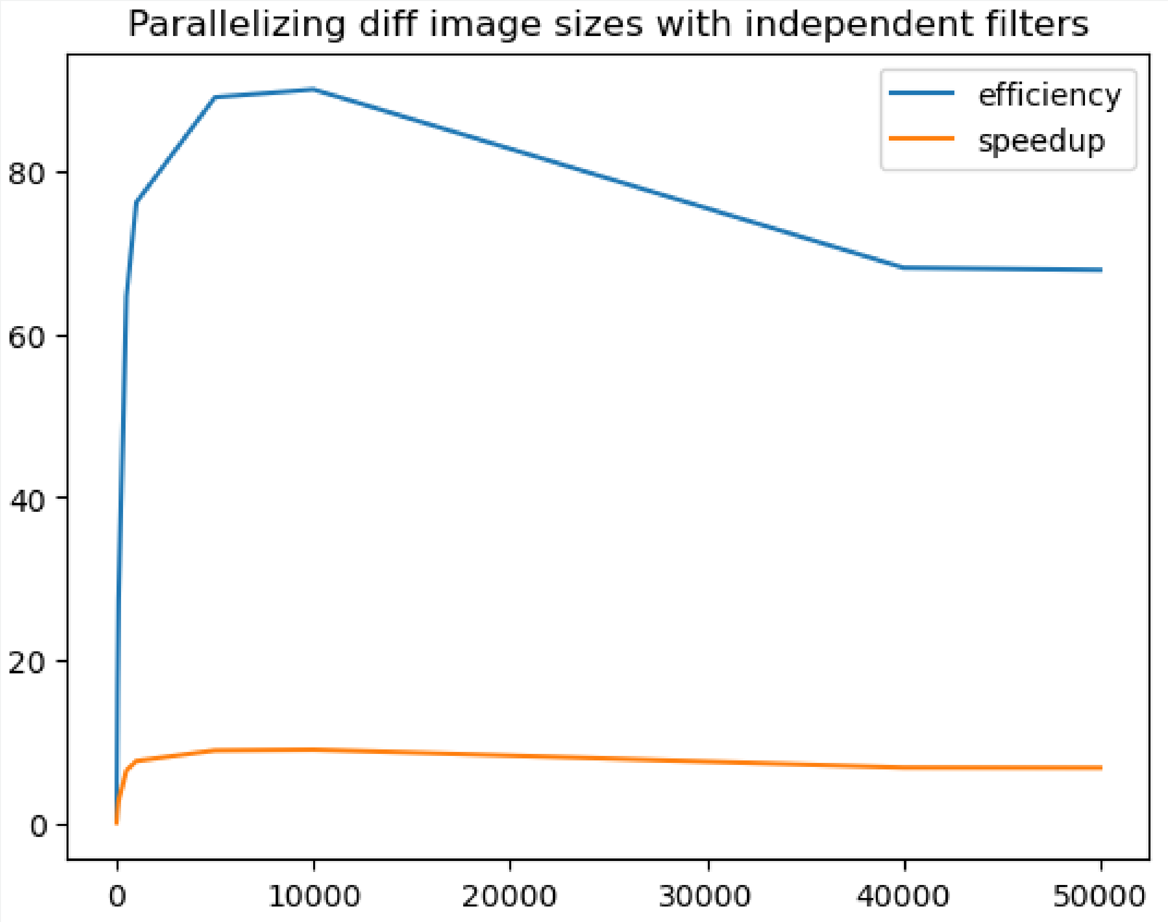


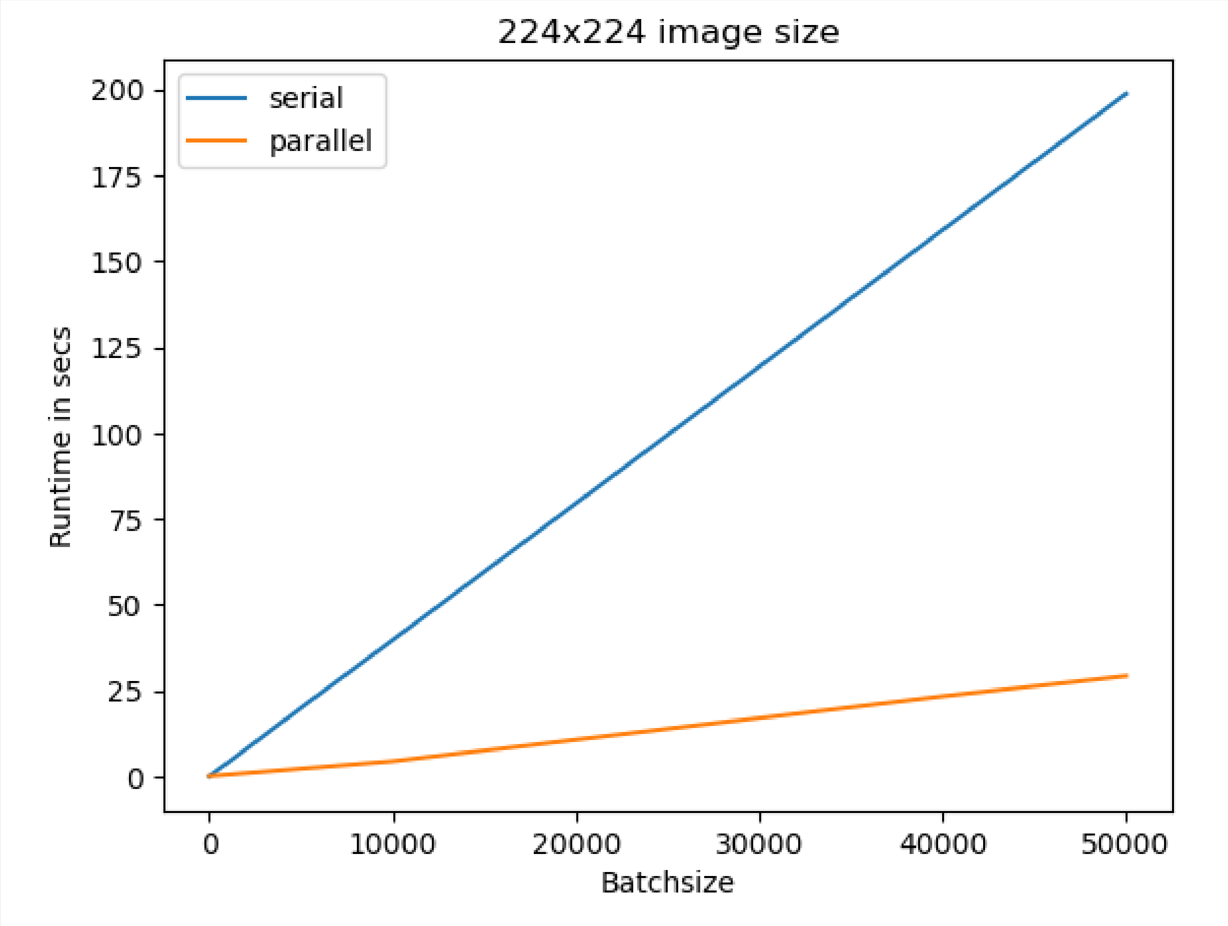


	batchsize	serial	parallel	efficiency	speedup
0	1	0.001752	0.094326	0.19	0.018577
1	10	0.014036	0.097541	1.44	0.143898
2	50	0.067977	0.108618	6.26	0.625837
3	100	0.134422	0.125572	10.70	1.070477
4	500	0.669595	0.215474	31.08	3.107544
5	1000	1.395840	0.346923	40.23	4.023486
6	5000	6.644290	1.143820	58.09	5.808860
7	10000	14.788000	3.127430	47.28	4.728483
8	40000	57.223400	12.372400	46.25	4.625085
9	50000	75.623600	16.554000	45.68	4.568298

224X224



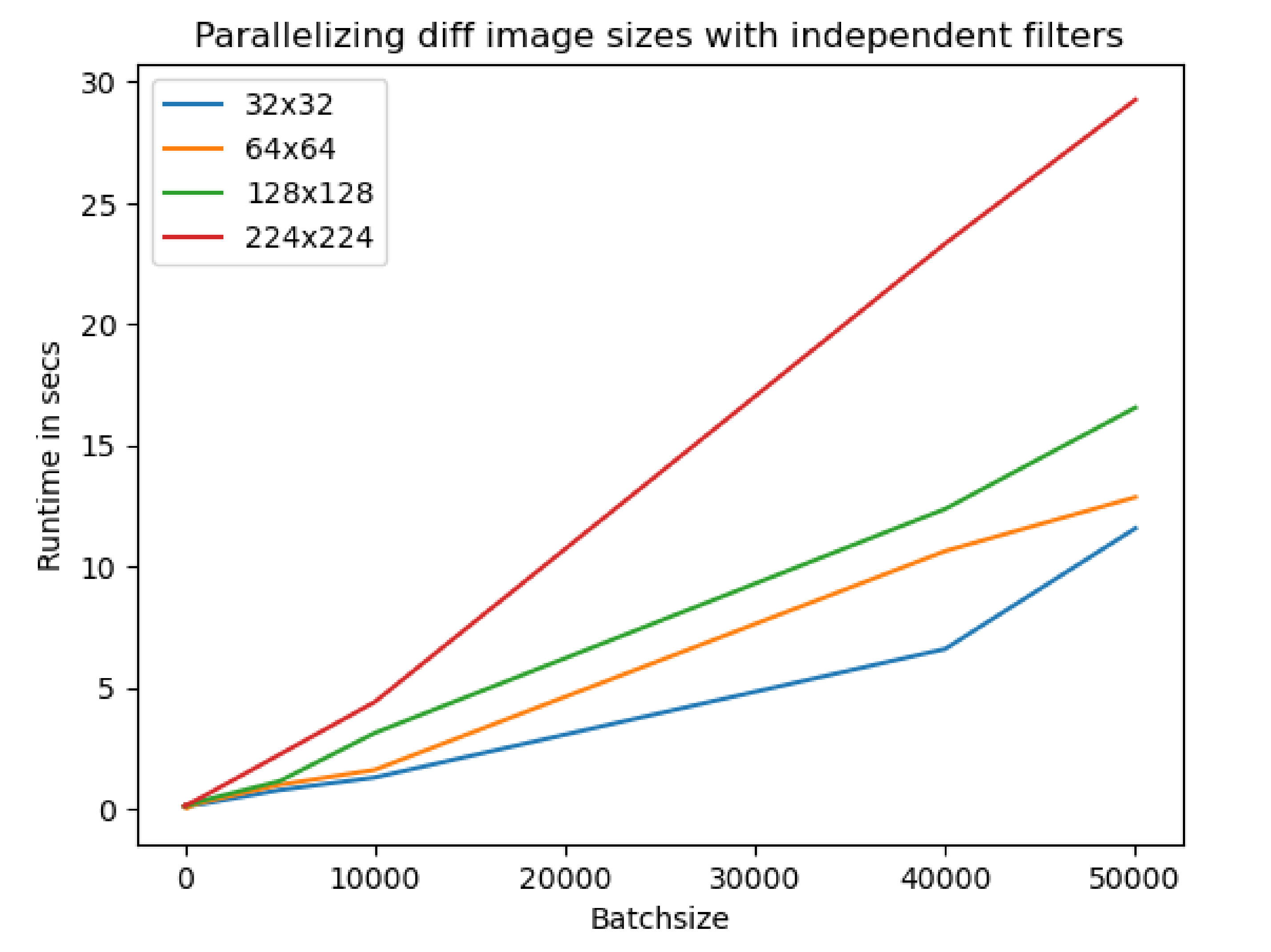




	batchsize	serial	parallel	efficiency	speedup
0	1	0.005271	0.107512	0.49	0.049027
1	10	0.040253	0.099864	4.03	0.403078
2	50	0.198049	0.117462	16.86	1.686069
3	100	0.419870	0.152665	27.50	2.750270
4	500	2.018640	0.312031	64.69	6.469357
5	1000	3.933940	0.516324	76.19	7.619131
6	5000	19.977900	2.241690	89.12	8.911982
7	10000	39.689000	4.405280	90.09	9.009416
8	40000	159.037000	23.322900	68.19	6.818920
9	50000	198.665000	29.254900	67.91	6.790828

COMPARING PARALLEL VALUES

	batchsize	32x32	64x64	128x128	224x24
0	1	0.001007	0.095619	0.094326	0.107512
1	10	0.002562	0.097408	0.097541	0.099864
2	50	0.010066	0.144760	0.108618	0.117462
3	100	0.019955	0.011314	0.125572	0.152665
4	500	0.099981	0.174854	0.215474	0.312031
5	1000	0.279154	0.275101	0.346923	0.516324
6	5000	1.538430	0.991048	1.143820	2.241690
7	10000	2.730920	1.607620	3.127430	4.405280
8	40000	11.435900	10.632400	12.372400	23.322900
9	50000	15.220800	12.858100	16.554000	29.254900



OBSERVATIONS

	batchsize	32x32	64x64	128x128	224x24	e32	e64	e128	e224
0	1	0.001007	0.095619	0.094326	0.107512	0.10	0.10	0.19	0.49
1	10	0.002562	0.097408	0.097541	0.099864	0.26	0.50	1.44	4.03
2	50	0.010066	0.144760	0.108618	0.117462	0.91	1.55	6.26	16.86
3	100	0.019955	0.011314	0.125572	0.152665	1.79	38.63	10.70	27.50
4	500	0.099981	0.174854	0.215474	0.312031	6.13	12.33	31.08	64.69
5	1000	0.279154	0.275101	0.346923	0.516324	13.39	18.95	40.23	76.19
6	5000	1.538430	0.991048	1.143820	2.241690	19.93	26.07	58.09	89.12
7	10000	2.730920	1.607620	3.127430	4.405280	21.13	35.05	47.28	90.09
8	40000	11.435900	10.632400	12.372400	23.322900	17.34	21.43	46.25	68.19
9	50000	15.220800	12.858100	16.554000	29.254900	13.15	21.56	45.68	67.91

1. **As size of the images increases, the efficiency decreases.**
2. **As batch size of the images increases, the efficiency decreases.**
3. **There is not much difference in parallel runtime for images of different sizes when using dependent filters.**

THANK YOU