A Picture Is Worth a Thousand... Features! Using Computer Vision Alongside Machine Learning in Computer Systems

Thaleia Dimitra Doudali
IMDEA Software Institute
Madrid, Spain
thaleia.doudali@imdea.org

1 INTRODUCTION

The use of machine learning methods in computer systems shows great potential to improve upon existing approaches and deal with the increased complexity across application domains, system and hardware configurations and management of current and emerging computing platforms. For example, the use of reinforcement learning is shown to improve upon system [7] and hardware [14] configuration and the use of deep neural networks to result in more sophisticated resource management decisions, such as for hybrid memory management [12] and data prefetching [13, 15]. The common aspect of current use cases of machine learning *for* systems is the type and representation of the input data, which are usually numerical values (raw or encoded) that capture certain behaviors and metrics observed at certain times or though time, e.g., throughout workload execution.

At the same time, there is a proliferation of machine learning methods deployed for image classification, video analytics, object detection and recognition. These machine learning methods are often coupled with computer vision algorithms to build end-to-end image-based pipelines, such as for the case of autonomous driving [6]. In the context of computer systems research, we are yet to deploy such algorithms, because of the lack of using images inside system pipelines.

As researchers and developers of computer systems, we are actually using images in our daily activities. We *create images* to illustrate how our systems operate and compare against other solutions in the research artifacts we produce. In addition, we *observe images* to understand how and why our systems and computing platforms operate in the observed way, for instance by using performance monitoring tools such as Grafana [1] and Intel's VTune [3]. In my own personal experience [11], observing image representations of certain system-level behaviors has helped me immensely in producing insights and ideas for system design.

How much of a wild and crazy idea[1] is it to create image-based computer system pipelines? An image can be a visual representation of an observed system-level behavior, e.g., an image that captures which memory addresses are being accessed throughout workload execution. In addition, an image can also capture the values of an observed metric across time, e.g., CPU usage across the hours of a day and visually illustrate the changes in the value of the usage. However, an image used in a system pipeline doesn't have to be constructed in a way that it is visually comprehensible by the human eye. For instance, in Sinan [16] an image is a 3D tensor consisting of per-tier resource utilization within a pastime window. In summary, an image can be a graphical representation or more simply a multidimensional array that captures an observed system-level information over a window of time.

Why would a system benefit from using images? Well, we are currently not taking advantage of a whole set of useful and optimized algorithms, such as computer vision and image processing algorithms, as well as machine learning methods like classifiers that are effective for image-based object detection and recognition. In addition, using images reduces the dimensionality of our data to fewer dimensions, such as 2D or 3D [8]. Using images is particularly effective for data that exhibit high temporal and spatial correlations, such as data access patterns [10]. Moreover, it can reduce the size of input data and reduce training and learning times. For example, using a 10x10 pixels image to depict a time series of 10,000 points, reduces the length of input data by 100×. In other words, *a picture is worth a 1000... features*. Nevertheless, using image-based representations of system-level data creates an opportunity to leverage new machine learning and computer vision methods and reduce the huge amount of data that systems currently produce.

In the remainder of this document, I will explain why using images inside systems is a tangible idea, via specific example use cases, and describe open questions and challenges that will arise. I hope to provoke your thoughts on how we represent system-level data and open your eyes to a new class of algorithms that we are currently not leveraging. Taking it a step further, I believe this can revolutionize the way we design future systems and hope to establish a new intersection of active research, that is the **SysMLCV**, where we use Machine Learning and /or Computer Vision *for* Systems.

2 LEARNING DATA ACCESS PATTERNS

In this Section, I will showcase how computer vision can be used alongside machine learning in computer systems, focusing on a particular use case, that is to learn data access patterns. More specifically, I will focus at the memory-level, where current state-of-the-art system-level solutions [12] deploy Recurrent Neural Networks to learn such memory access patterns. In order to build an image-based pipeline, we need to create visual representations of how applications access memory. Figure 1 includes an example of such an image, where a single black point corresponds to the virtual memory address of the 4 KB page that was accessed (y-axis) on the specific time step (x-axis). The raw data available for visualization are memory access traces that are publicly available for the SPEC benchmarks [4].

Figure 1 illustrates a system-level solution that creates such images, by visualizing information it collects while monitoring the application's memory accesses (memory access trace), and then uses image-based machine learning methods to recognize and predict future patterns. Depending on the type of access pattern, e.g.,

[1]This document summarizes the author's selected talk at the ASPLOS 2022 Wild and Crazy Ideas (WACI) Session, March 2022, Lausanne, Switzerland.

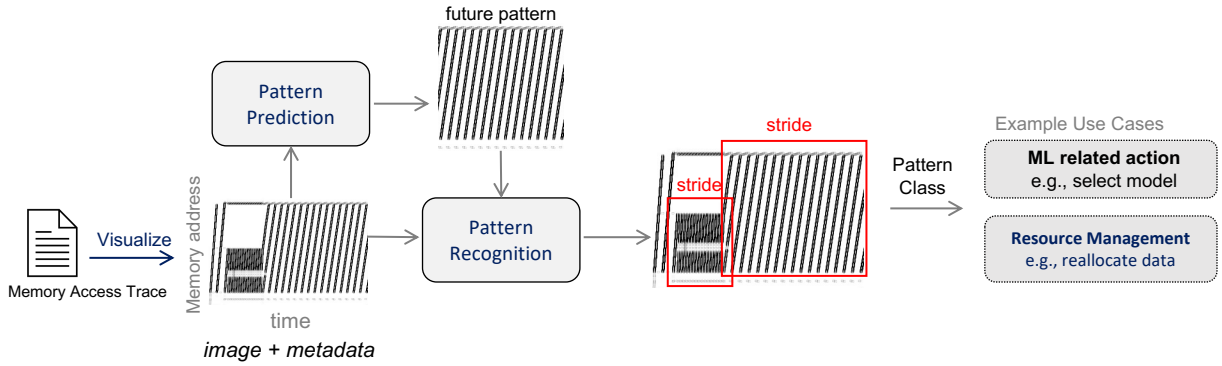

Figure 1: An image-based system for learning, recognizing and predicting memory access patterns.

sequential, strided, random, a system may choose to take a difference decision regarding its machine learning-based pipeline, or make a resource management decision, such as to allocate, move or free up memory. Next, I describe the role of the proposed system's subcomponents and open problems, challenges and opportunities that arise when building an image-based system pipeline.

2.1 Visualization

As mentioned before, creating images reduces the dimensionality of the overall sequence of memory accesses, thus certain patterns may become less visible in a limited 2D space projection. This is particularly the case for long-running applications with large memory footprints, where creating a single image for the whole workload execution results in an all black image, since there can be millions of memory accesses that are now projected in an e.g., 128x128 pixel image. This raises the following challenges and opportunities.

Open Problems:

- How many images to create per workload, how often in time to capture and visualize the access patterns?
- How to determine what size and resolution the images should have? Small images are better for faster image processing, but how much information should they contain to properly visualize it? Do we want clear lines or just trends?
- Are we creating colored, grayscale or black-and-white images?
- What metadata to associate with the image? For example, we could include the workload information, the level of storage hierarchy of the data accesses (cache, memory, disk), the total size of the memory footprint (y-axis) and workload execution time (x-axis). What is the size of th amount of metadata?
- What storage schema is most efficient, depending on the total number of created images and its metadata?
- How to build operating system-, library-, compiler-, runtime-level support for providing the raw data and frequency of creating such images.

2.2 Pattern Recognition

Machine learning algorithms for pattern recognition are trained over labeled data sets, such as ImageNet [2] and CIFAR-10[5] that properly mark and label a pattern or object with its corresponding class. Can we build such a dataset with images of data access patterns? Such a dataset will not only enable the use of pattern recognition methods, but also be a valuable contribution to the academic community. Even just observing the images and getting insight is a useful contribution. Here are the challenges and opportunities raised.

Open Problems:

- What classes to define for labeling data access patterns?
- What labeling guidelines to provide for community contributions to the public dataset?
- Which classifiers to use for pattern recognition?
- What is the impact of a misclassified / unrecognized pattern?
- Will operating system-, library-, compiler-, runtime-level support help in properly marking and labeling the patterns?

2.3 Pattern Prediction

As the image-based pipeline is creating images of data access patterns throughout workload execution, we now have a sequence of such images, that is nothing else but a **video**! Therefore, we can now leverage machine learning algorithms that predict the next frame of a video to predict the next frame, i.e., image of data access patterns, for a particular workload. This raises the following challenges and opportunities.

Open Problems:

- How does such a solution compare to current ML-based methods that predict raw data and a non-ML method? Metrics to consider can be the prediction accuracy, training and inference times and the effect on the system's end decision-making.
- What is the impact of a mispredicted pattern?
- What is the granularity of machine learning model training? Do we train a system-level model that learns all patterns, an application-level model or a pattern-level model?
- How does the system operation interval overlap with the training periods?

2.4 Summary

In summary, I presented how images can be used to build purely image-based system pipelines for the purpose of learning memory

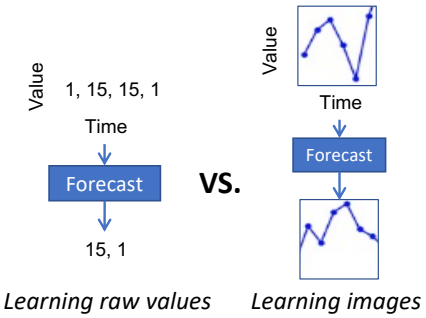

Figure 2: Time series represented with images.

access patterns, which can be extended to any level of data access patterns (application-, cache-, memory-, storage-level). Will such a pipeline be more effective than current solutions? I believe it is worth exploring, especially since it enables the use of algorithms (computer vision, image processing, ML classifiers, video frame predictors) that we have not considered so far because of the way we represent data at the system-level.

3 LEARNING TIME SERIES DATA

In this Section, I present another use case where images could be used in systems. Due to their dynamic operation, systems collect time series data for various performance metrics, such as resource consumption (compute, memory, storage) for native platforms, servers, cloud datacenters etc. Forecasting future values of these metrics helps better provision the hardware, balance the load and improve upon scheduling. Thus, we keep looking for ways to improve upon learning such metrics, enable more accurate predictions, online learning and reduce training and inference times. Recent work in financial time series data shows the potential of having more accurate forecasting when learning images instead of raw data [9]. Will this work for system-level metrics? Is learning images of time series data more effective than learning the raw values of the data, as illustrated in Figure 2? Exploring this raises challenges and opportunities that are similar to the ones described in Section 2.1.

Open Problems:

- How many time steps per image?
- What size, resolution and color should the images have?
- One line or more per image? To potentially capture correlations across metrics, such as compute, memory and disk usage.
- How does the image-based learning compare to forecasting raw values with machine learning or analytical methods? Metrics to consider can be the prediction accuracy, training and inference times and the effect on the system's end decision-making.

4 CONCLUSION

This document makes a case about the potential impact of a new intersection of research areas, that is computer vision, machine learning and computer systems. The goal of this *wild and crazy idea* is to rethink the way we represent system-level information and motivate you to consider visualizing it, that is, creating images. This is essentially a way to reduce the dimensionality of the data, capture spatial and temporal correlations, and potentially reduce machine learning times. Most importantly, it unlocks the opportunity to use computer vision image processing methods, that we currently do not leverage due to the lack of visualization. Will images revolutionize the way we build systems? Will image-based system pipelines that couple computer vision with machine learning be more effective than machine learning-based or analytical approaches? In any case, visualizing system-level behaviors and creating relevant public image datasets, can be an important contribution to the community and lead to greater insight on why systems operate in a certain way and how to design better systems.

ACKNOWLEDGMENTS

The ideas presented in this document are proposed as future work based on the author's PhD dissertation [11] at the Georgia Institute of Technology, that was completed in 2021 under the supervision of Professor Ada Gavrilovska.

REFERENCES

[1] Grafana. https://grafana.com/.
[2] ImageNet. https://image-net.org/.
[3] Intel VTune Profiler. https://software.intel.com/content/www/us/en/develop/tools/oneapi/components/vtune-profiler.html.
[4] ML-based Data Prefetching Competition. https://sites.google.com/view/mlarchsys/isca-2021/ml-prefetching-competition.
[5] The CIFAR-10 dataset. https://www.cs.toronto.edu/~kriz/cifar.html.
[6] Mariusz Bojarski, D. Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, L. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, X. Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *ArXiv*, abs/1604.07316, 2016.
[7] Ruobing Chen, Jinping Wu, Haosen Shi, Yusen Li, Xiaoguang Liu, and Gang Wang. Drlpart: A deep reinforcement learning framework for optimally efficient and robust resource partitioning on commodity servers. In *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '21, page 175–188, New York, NY, USA, 2020. Association for Computing Machinery.
[8] Naftali Cohen, Tucker Balch, and Manuela Veloso. The effect of visual design in image classification. *CoRR*, abs/1907.09567, 2019.
[9] Naftali Cohen, Tucker Balch, and Manuela Veloso. Trading via image classification. *CoRR*, abs/1907.10046, 2019.
[10] Deeksha Dangwal, Weilong Cui, Joseph McMahan, and Timothy Sherwood. Trace wringing for program trace privacy. *IEEE Micro*, 40(3):108–115, 2020.
[11] Thaleia Dimitra Doudali. *Adding Machine Intelligence to Hybrid Memory Management.* PhD dissertation, Georgia Institute of Technology, 2021.
[12] Thaleia Dimitra Doudali, Sergey Blagodurov, Abhinav Vishnu, Sudhanva Gurumurthi, and Ada Gavrilovska. Kleio: A hybrid memory page scheduler with machine intelligence. In *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '19, pages 37–48, New York, NY, USA, 2019. ACM.
[13] Milad Hashemi, Kevin Swersky, Jamie Smith, Grant Ayers, Heiner Litz, Jichuan Chang, Christos Kozyrakis, and Parthasarathy Ranganathan. Learning memory access patterns. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1919–1928. PMLR, 10–15 Jul 2018.
[14] Sheng-Chun Kao, Geonhwa Jeong, and Tushar Krishna. Confuciux: Autonomous hardware resource assignment for dnn accelerators using reinforcement learning. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 622–636, 2020.
[15] Zhan Shi, Akanksha Jain, Kevin Swersky, Milad Hashemi, Parthasarathy Ranganathan, and Calvin Lin. A hierarchical neural model of data prefetching. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2021, page 861–873, New York, NY, USA, 2021. Association for Computing Machinery.
[16] Yanqi Zhang, Weizhe Hua, Zhuangzhuang Zhou, G. Edward Suh, and Christina Delimitrou. Sinan: Ml-based and qos-aware resource management for cloud microservices. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS 2021, page 167–181, New York, NY, USA, 2021. Association for Computing Machinery.