Data Storage and Management for Image AI Pipelines

Utku Sirin
Harvard University
Boston, USA

Stratos Idreos
Harvard University
Boston, USA

Abstract

Image AI is essential for various applications, such as self-driving cars, medical imaging, and smart farming. Data management is key for efficient image AI, from how to store images to how to manage data while processing the images. This tutorial overviews the emerging area of image AI pipelines by combining approaches from various cross-disciplinary areas such as data management, digital signal processing, computer vision, and machine learning. We specifically focus on image storage and data management.

The tutorial first gives an overview of image AI pipelines step by step, how they work and the main the challenges. We then describe the main approaches to making image AI pipelines more efficient. We first cover how image AI pipelines store images based on standard storage formats, learned formats, task-specific learned formats, and self-designed formats. Second, we cover how state-of-the-art approaches manage data within image AI pipelines. We identify and describe three main approaches to making image AI pipelines more efficient by efficiently managing data within the pipeline: (i) compressing intermediate data, (ii) materializing and re-using data objects, and (iii) parallelism for better hardware utilization. Lastly, the tutorial covers open data management and systems problems and future directions in making image AI pipelines more efficient.

CCS Concepts

• **Information systems** → **Record storage alternatives**; • **Computing methodologies** → **Image representations**.

Keywords

AI; Image AI; Image Storage; JPEG; AI Training Time; AI Inference Time; Self-designing Systems; Learned Compression; Systems for ML

ACM Reference Format:

Utku Sirin and Stratos Idreos. 2025. Data Storage and Management for Image AI Pipelines. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3722212.3725640

1 Data Management for Image AI

Image AI Improves Every Aspect of Modern Human Life. Image AI has shown great success in numerous areas, from medical imaging to self-driving cars. Medical doctors now use image AI to get help in their decision-making process in the early/late detection of diseases. Companies deploy image AI tools to enhance worker safety conditions, increasing productivity. Farmers use image AI to efficiently monitor the conditions of their crops and take preventive actions against any potential disease, further improving their yield [31, 42, 122].

Problem: Image AI is Expensive. Every year, billions of digital cameras capture trillions of images. These images consume thousands of petabytes of storage in the cloud. Numerous AI applications process these massive datasets for various purposes, such as personalized content management, image reconstruction, and visual captioning. Applications access data over remote storage servers, and data needs to be moved and processed in compute nodes using high-end processors. Moving and processing data incurs an immense dollar cost for the cloud vendors and application developers, limiting the accessibility of image AI [11].

Image AI Pipeline. A typical image AI pipeline is composed of four main operations. The first one is fetching raw data from persistent storage, which can be remote, local, or a combination of the two. The second is converting raw data into a learnable in-memory format, such as the visually recognizable red-green-blue (RGB) format. Today, CPUs primarily perform this conversion. Third, the learnable format of images from the host memory moves to one or more GPU devices. GPUs primarily run image AI models in a single or multi-GPU setting, depending on the AI model's size and data. This movement typically happens over a device-to-device connection such as PCIe or a fast interconnect network providing fast direct communication across devices within the same server. Lastly, the AI models process the moved images in their learnable formats to perform the particular AI task, such as training the AI model or performing inference over the images using the AI model residing on the GPU [51].

Data Management is Key for Efficient Image AI. The cost of these operations is fundamentally defined by how efficiently image AI systems manage their data. Fetching and storing image data are defined by the storage/file formats used for images. While some algorithms aggressively compress images, some lightly compress, each leaving an image AI pipeline with different trade-offs. Converting raw image data into a learnable format and how and when the conversion happens is defined by the storage format used. While standard storage formats use visually recognizable RGB format [2, 4], recent AI-task-specific formats use semi-compressed visually unrecognizable formats for a more efficient decoding process [16, 101]. Moving images from host memory to the GPU memory depends on how images are represented in main memory, as well as available GPU memory consumed by multiple parties such as the AI model itself as well as the batches of images and gradients [52, 92]. Efficient management of data objects within and across host-to-GPU memory is crucial to fit large models into GPUs and train them efficiently [50, 85, 121]. Similarly, performing an AI task

SIGMOD-Companion '25, June 22–27, 2025, Berlin, Germany

ACM ISBN 979-8-4007-1564-8/2025/06
https://doi.org/10.1145/3722212.3725640

within a GPU requires efficiently handling data objects of the AI model itself together with the gradients and batches of images, including compressing them efficiently [54, 77, 107, 112], swapping them back-and-forth GPU memory [52, 92], and executing multiple AI models or multiple data batches in parallel to improve GPU utilization [53, 117]. From raw data to execution of AI models, managing data plays a key role in making image AI systems efficient.

Tutorial Structure. This tutorial highlights the connection between image AI and data management systems. It covers major image AI studies in the data management field. It combines them with cross-disciplinary studies from digital signal processing, computer vision, and machine learning into a unified view of data management systems. Figure 1 presents the tutorial's structure over an image AI pipeline. It is composed of four main sections. The first section covers basic problem setup, including how typical image AI pipelines work, the main bottlenecks, the main challenges for making image AI pipelines efficient, and the main approaches to address these challenges. The second and third sections detail how data management solutions can drastically improve image AI systems performance. The second section covers how image AI pipelines store images. It contains four sub-sections: (i) standard formats, e.g., JPEG, (ii) learned formats, (iii) task-specific learned formats, and (iv) self-designed formats. It describes major solutions within each sub-category and discusses their challenges and relationships with each other. The third section covers how image AI pipelines manage data within the pipeline. It contains three sub-sections: (i) compression, (ii) materialization and reuse, and (iii) parallelism for better hardware utilization. It covers the major approaches within each subsection and discusses how they are different from each other and how they complement each other. Finally, the tutorial concludes with future research opportunities and open problems in making image AI systems efficient.

2 Image AI Pipeline

An image AI pipeline refers to a computing stack that (i) stores images, (ii) reads images, (iii) coverts them from raw data into a learnable format, (iv) moves them across and/or within machines, and (v) finally executes one or more AI models over the images for training and/or inference, as also shown in Figure 1. An image AI application goes through the entire stack, typically iteratively, multiple times over static and/or dynamic datasets. This part of the tutorial describes the image AI pipeline in detail. It exposes a data-centric view of image AI systems to understand the data management challenges and how we can approach the problems.

1. Storing and Reading Images. Storing images requires processing images into an on-disk format with a combined lossy and lossless compression technique. Reading an image is the inverse of storing it, which requires performing the inverse of every operation that storing does. While both storing and reading images can be in the application's critical path, images are typically stored once and read multiple times due to the iterative nature of the image AI pipelines [4, 51, 101].

2. Decoding. Once read, images are simply series of bytes in their raw format. Image AI pipelines require converting them into a learnable format, such as a visually recognizable red-green-blue (RGB) format by decoding images. A learnable format is a format that AI models can operate on. There are various learnable formats. The most popular format is the visually recognizable RGB format. Other formats include self-designed formats and learned formats. All these formats represent images in a different domain, offering different cost vs. model quality trade-offs [4, 80, 101].

3. Moving Images within/across Machines. AI models today typically reside on a GPU machine, where the host machine is a server-grade CPU with a large main memory in the orders of hundreds of GBs. The host machine is connected to co-processor GPU(s) with its own private memory, typically 20 to 80 GBs. While some machines have a single GPU, some have four, and some have up to 32 GPUs. CPU and GPUs are connected with various links, such as a PCIe link or a faster NVlink that allows direct CPU-to-GPU and GPU-to-GPU communication. While CPU and GPU are both typical sources of bottlenecks in image AI pipelines, the network connection between CPU and GPUs and within GPUs are critical to keeping GPUs busy, which otherwise leads to a GPU underutilization constituting a major source of inefficiency for image AI pipelines [53, 117].

4. AI Model Execution over Images. AI models are executed over images once they reach their final destinations and produce the outcome. Image AI models are today layered, such as convolutional neural networks (CNNs) and visual transformers [48, 74]. Image AI models process images layer by layer in a sequential manner, where every layer consumes the output of its previous layer. The output of a layer is referred to as an activation/feature map. Depending on whether it is training or inference, AI model execution produces different amounts and types of data. Inference requires keeping a single activation map during the inference, which incurs a low memory overhead. Training requires not only the current activation map but also several previous ones and gradient matrices to be able to update model parameters. As GPU memories are significantly smaller than host memory, i.e., CPU memory, training is usually a memory-intensive operation for GPUs and hence often constitutes a bottleneck. Algorithms employ various methods such as swapping data matrices across CPU and GPU memory or compressing them to mitigate this problem and accelerate image AI systems [52, 92].

3 Image Storage for AI

Image storage is one of the fundamental elements of image AI pipelines. Image AI pipelines read images based on how they are stored and convert them into a learnable format primarily based on how images are stored. JPEG, to illustrate, stores images after applying a version of Fourier transformation and uses a lossy compression algorithm coupled with a lossless encoding algorithm. Reading JPEG files requires fetching so many bytes of JPEG data that are decided by the combined lossy and lossless compression algorithms. Decoding JPEG files into a learnable format requires converting them into a visually recognizable red-green-blue (RGB) format with a specific resolution, which is also decided by the specific version of the used JPEG. The resolution further decides how much data needs to be decoded and transferred within/across the machine from the host CPU to the GPU or across GPUs and processed by the GPU. Therefore, storage impacts all time components

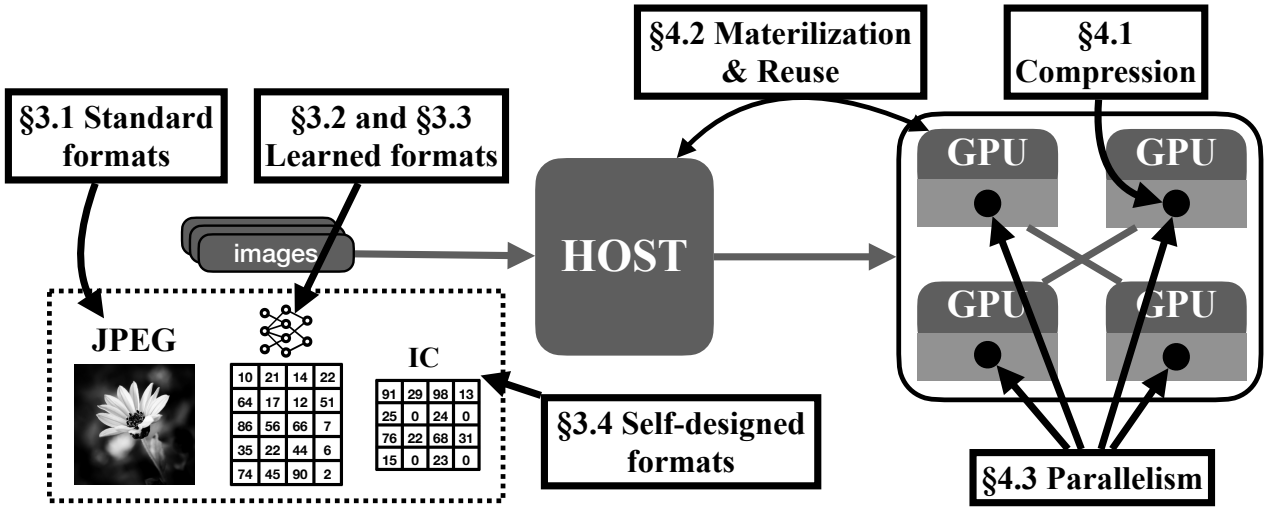

Figure 1: Image AI pipeline is composed of four stages: (i) reading image files, (ii) decoding them into a learnable format, (iii) transferring them to a GPU cluster, and (iv) executing an AI model over images. This tutorial covers all aspects of making image AI pipeline efficient, from storage to managing data within the pipeline.

in an image AI pipeline and is crucial for making image AI systems efficient.

1. Standard Storage Formats. Many image AI systems store images by using a standard storage format, such as JPEG or PNG. There are two types of standard storage formats: (i) lossy formats and (ii) lossless formats. Lossy storage formats employ a lossy compression algorithm, where some data is lost concerning a certain optimization criterion. For JPEG, it is the visual quality. Its lossy compression algorithm is based on losing information by minimally hurting the visual quality of the compressed image. Losing information brings a significant advantage regarding space-saving, as lossy compression compresses data directly, removing a certain amount of information. However, it could sometimes be detrimental as it nevertheless causes a loss in the optimization criteria, such as the visual quality [4, 9, 55, 105, 108]. Lossless compression algorithms are encoding algorithms that exploit data characteristics, such as repetitions or order of values in the data, to encode data with an amount of data that is less than the original data. PNG [2], for example, uses the DEFLATE algorithm [37], which is a combination of LZ77 and Huffman coding. DEFLATE creates a table of symbols in the data, such that the most frequently used symbols are encoded with the least number of bits. This way, it exploits the repetitions in the data and encodes data with fewer bits than the original data. As a lossy overall compression algorithm, JPEG uses a lossless compression algorithm in its final stage to efficiently encode a final set of bytes. It uses an arithmetic encoding algorithm, where data is sorted concerning a specific order and encoded by using a run-length encoding algorithm to exploit repetitions in the data. Lossless compression algorithms dramatically reduce image file sizes, as images are high in repetitions in their data values. The trade-off they offer is space versus decoding time. The deeper an image is compressed and encoded into a file, the longer it takes to decode it. Image AI systems need to balance this trade-off, as decoding is typically performed on a CPU and can easily become a bottleneck of an image AI pipeline.

2. Learned Formats. Efficiently compressing and storing images saves a large amount of data to store and move. Hence, even small improvements in compression ratio can provide millions of dollars of cost-savings and thousands of pounds of carbon emissions thanks to energy savings. Learned storage formats aim to learn a specific representation of images that is cheap to store and contains enough information to restore the image to its original format fully. Cheaply storing an image requires losing lots of information from data, whereas fully restoring images with high visual quality requires keeping all information in images. As these two goals contradict, learned formats use a parameter that balances this trade-off between compression ratio and visual quality, i.e., rate-distortion trade-off. Studies have shown that learned formats can improve standard formats by more than 2x in terms of compression ratio while achieving a similar visual quality. Learned formats, however, suffer from increased encoding and decoding times compared to standard formats. Standard formats use static data transformations to transform images from their native visually recognizable format to another domain to compress them efficiently. JPEG, for example, uses a version of Fourier transformation, discrete-cosine transformation (DCT), to transform images into the frequency domain since the frequency domain allows a more efficient lossy compression. While DCT is expensive, it is significantly cheaper than executing an AI model to obtain a learned representation. Similarly, reconstructing an image requires an inverse DCT operation for JPEG, which is significantly cheaper than executing another AI model to restore an image from its learned and compressed format fully. Therefore, while learned formats can successfully improve compression ratio for a similar visual quality, they suffer from increased read and write times for images. For image AI applications, where read and write are on the critical path, and there is only a limited

amount of GPU resources, learned formats can cost a significantly higher cost than standard formats [5, 6, 80].

3. Task-specific Learned Formats. Recent studies took a step forward and improved learned formats by learning a representation that contains just enough information to perform a specific AI task successfully. Most AI tasks can be performed with much less information than the original images contain. Task-specific learned formats exploit this dimension of the problem. Unlike learned formats that target visual quality, they run AI tasks straight over the learned representation. Hence, they are significantly fast when they decode images, though being still expensive when they encode images due to using an AI model to perform the encoding. Task-specific formats differ in terms of the AI model they use and their target AI task. While some studies learn a single representation for a specific task [15, 17, 28, 72, 106], some learn a shared representation for a collection of tasks [16, 29, 30, 73, 111], and some others learn a modified version of a standard format, such as JPEG, tailored for a specific AI task [13, 14, 36, 38, 44, 75, 97, 118, 119]. All these studies set a trade-off parameter among two objectives of the learning algorithm: reducing data storage versus achieving high accuracy. For different trade-off settings, studies have shown that compression ratio can be improved by more than 2x, and accuracy can be improved by more than 10% compared to standard formats, thanks to learning a task-specific representation.

4. Self-designed Formats. More recently, researchers from the database community proposed a self-designing storage format that designs itself for a given AI task. Like task-specific learned formats, self-designed formats tailor their storage for a specific AI task. Instead of learning a new format each time, however, self-designed formats rely on a one-time created, rich design space of storage formats, each offering a different trade-off regarding compression ratio, end-to-end inference time, and model quality. Each AI task has its intrinsic characteristics regarding how much data should be maintained in images to perform the AI task successfully. Given an AI task and performance budget, self-designed formats search within its design space for the most accurate storage format that fits into a given storage budget. As the user adapts its budget, self-designed formats allow cheap what-if analyses over different storage formats to explore the design space and decide on what storage format would work the best for a given AI task. Compared to learned storage formats, self-designed formats use a richer design space of storage formats, as they have more freedom in designing storage formats with different data representations and compression ratios. For example, learned formats learn a fixed shape of representation of images, i.e., all images are transformed into a tensor of, e.g., 64x16x16. Self-designed formats, however, offer different ranges of representations, which allows much more flexible trade-off space in inference time versus model quality, as representation shape plays a crucial role in GPU time, much like the resolution of an image. Furthermore, self-designed systems learn all trade-offs at once and reuse them for all performance budgets. In contrast, learned formats require re-learning a different representation for each performance budget. As a result, studies have shown that self-designed formats can reduce end-to-end inference time by more than 10x, by losing little or no accuracy, compared to standard and learned storage formats [62, 101, 102].

4 Managing Data Inside Image AI Pipeline

Image AI pipelines produce and consume enormous amounts of data only while processing the images. Today's AI models are composed of a set of sequential layers. Every layer transforms a batch of images from one representation to another, finally producing the outcome of the AI model. The intermediate representations are called activation or feature maps and are heavy three-dimensional objects with complex floating points, high memory consumption, and high processing costs. If the image AI pipeline performs inference, activation maps can be maintained one at a time and flushed out otherwise. If the image AI pipeline is performing training, many activation maps are necessary to keep in the pipeline to perform training operations. Activation maps take a significant amount of GPU memory, often limiting training capacity regarding how large models can be trained and/or how efficiently training can happen. This part of the tutorial will describe the primary data management solutions for making image AI pipelines efficient when managing intermediate data. The section contains three subsections centered around the main methods used to improve image AI pipelines: (i) compression to reduce moved/processed data, (ii) materialization and reuse to reduce computation, and (iii) parallelism for better hardware utilization.

1. Compression to Reduce Moved/Processed Data. Image AI pipelines produce significant data while processing images, including activation/feature maps and gradients. Large amounts of data limit training quality, as GPU memory is usually small, and models start not fitting into single or even multi-GPU memories. In a distributed setting, large amounts of intermediate results cause high communication costs across GPUs, causing GPU under-utilization and high training times. This subsection will cover how image AI pipelines use compression to reduce GPU memory consumption and cross-GPU communication, what trade-offs they offer, and their limitations.

Compressing Activation Maps. The first method is compressing activation maps. Activation maps usually dominate GPU memory consumption during training, as they are heavy large-scale objects with expensive high-precision floating point operations. Numerous compression methods have been proposed, from dropping values below a specific threshold [107] to using sophisticated error-bounded lossy compression algorithms over activation data [54]. Compressed activation maps consume significantly smaller GPU memory, allowing training larger models or using larger batch sizes to use GPU cycles more efficiently. Compression brings a three-dimensional trade-off: reduced memory consumption versus decreased accuracy versus training time. Compressed activation maps often need to be decompressed to their original format with a small bounded error to use in the main training pipeline, particularly during backward propagation. Hence, deeply compressed activation maps will take a long time to decompress and incur increased training times. Similarly, lossily compressed activation maps will cause inaccurate training results, causing decreased model accuracy. These disadvantages will come with the benefit of training large models on small cheap GPUs, which otherwise would be impossible [10, 41].

Compressing Gradients. Training in modern image AI pipelines is done by the gradient-descent (GD) algorithm. GD computes derivative of the optimization function and gradually updates parameters

to minimize the difference between the predicted and true values of the training samples. Derivatives of the optimization function are called gradients. While gradients consume less memory than activation maps, they must be communicated across different GPUs in distributed training settings. With the massive growth in dataset sizes, distributed training has become a standard procedure in image AI training. An important distribution strategy is data-parallel execution, where gradients are synchronized across GPUs and are used to update model parameters globally across all GPUs. This requires communication between GPUs, which becomes a main limiting factor in scaling performance. Gradient compression is used to reduce this communication cost. Methods range from layer-wise adaptive gradient compression [77] to lightweight compression algorithms [1, 112] and to fusing multiple compression operations into a single operation [71]. The trade-off is reduced model quality due to losing information versus reduced communication cost and compression computation overhead. Studies balance these goals with algorithmic improvements and heuristics, achieving 3-5x improvement in reduced communication cost with little or no model quality loss and computation overhead.

2. Materialization and Reuse to Reduce Computation. Most image AI pipelines are iterative. Developers use the same pipeline iteratively to refine their AI models, hyper-parameters, data augmentation methods, and many other parameters that could yield a better performance. The iterative nature of the pipelines entails large opportunities for materialization and reusing intermediate variables. This subsection will cover the main studies that exploit materialization and reuse for image AI pipelines.

Reuse in Activation Maps and Model Parameters. Image AI pipelines often use transfer learning in their training pipelines for faster training. Transfer learning refers to transferring a learned task to another similar task. An example would be training a new but only slightly different AI model with the same hyperparameters and dataset. In this case, the new AI model can transfer all the knowledge that the previous model could use. Materializing intermediate results, such as activation maps, and reusing common sub-operators across similar AI models could drastically improve training performance, eliminating a large amount of shared computation. The trade-off is that materialized results consume extra storage, and shared models consume active GPU memory, which can exhaust available resources. Hence, algorithms optimize across storage and memory constraints and find optimal materialization and reuse strategies across intermediate data such as activation maps and shared model parameters [39, 83].

Sacrificing Reuse for Reducing Peak Memory Consumption. Image AI training is composed of successive iterations over the AI model using randomly sampled image batches. Every iteration comprises two stages: a forward and a backward pass. A forward pass produces activation maps later used during a backward pass when computing successive gradients. These activation maps often exceed total GPU memory and need to be compressed as described earlier, checkpointed by swapping from GPU memory to CPU memory, or discarded and re-computed when necessary. Doing so allows image AI systems to scale to large-scale AI model training with smaller-sized and cheaper GPUs, significantly decreasing the cost of image AI applications. This, however, comes at the expense of increased training time, which image AI systems employ sophisticated optimization programs such as mixed integer linear programs to decide when and how to perform checkpointing/recomputation and achieve fastest training with minimum GPU consumption [24, 43, 52, 76, 91, 92].

3. Parallelism for Better Hardware Utilization. One of the major concerns in making image AI systems more efficient is using parallel hardware resources efficiently. This primarily points to improving GPU utilization. GPUs are often severely underutilized as they are at the end of the pipeline and often stall waiting for data. Also, they are orders of magnitude faster than CPUs or other hardware components in the pipeline, allowing them to finish their computation faster than others and making them remain idle. Parallel training algorithms require communications across GPUs, which further worsens the underutilization. This section will cover the main studies that exploit task/model/data parallelism to improve GPU utilization over image AI pipelines. It will discuss the trade-offs different methods offer, how complementary and different they are, and what limitations they suffer.

Data Parallelism. There are two main methods of parallelism: data parallelism and model parallelism. Data parallelism refers to splitting a training dataset across different GPUs and synchronizing learning across GPUs to achieve the highest accuracy. Data parallelism suffers from GPU underutilization for various reasons, such as low batch sizes, hardware heterogeneity, and workload imbalance. Image AI learning algorithms require batch sizes to be smaller than a certain size to deliver high model quality. However, small batch sizes cause GPUs to finish their computation quickly and idle while waiting for the next batch of images. Hardware heterogeneity makes this works causing some GPUs to finish their part of the learning faster than others. Similarly, workload imbalance results in some GPUs performing more processing than others. Methods improving GPU utilization for data-parallel executions range from efficient synchronization mechanisms to mitigate effects of delays [64, 79], to workload-aware balanced resource allocation [87].

Model Parallelism. Models can be split vertically and/or horizontally across different GPUs to scale learning for large AI models. Modern image AI models are layered, where each layer is executed sequentially, one after the other. Vertical splitting refers to splitting a single-layer computation across multiple GPUs, whereas horizontal splitting refers to splitting different layers into different GPUs. Model-parallel execution suffers from dependencies across intermediate data objects produced during training, such as activation maps and gradients. The challenge is to maximize utilization of GPUs without sacrificing model quality [50, 68, 85, 86, 110, 121, 123].

Data and Model Parallelism. Image AI practitioners often need to perform both model and data parallelism to utilize hardware resources maximally. The challenge is to devise an automated partitioning algorithm that jointly decides on what part of the model will be placed on what device(s) and which device will benefit from how much data parallelism in order to minimize communication and synchronization costs, maximize hardware utilization, as well as achieve a high model quality. Recent methods utilize reinforcement-learning-based search [81], search algorithms with simulated execution times [53, 117], adopt network routing algorithms [126], and use dynamic resource allocation schemes [90, 93, 94].

Task Parallelism. While it is possible to utilize the high degree of reuse across iterations in Image AI pipelines, every iteration is semantically independent. Recent image AI research proposes coupling model parallelism with task parallelism, where each task defines a single iteration, such as evaluating a particular hyper-parameter. Task parallelism adds one more level of parallelism opportunity, where hardware utilization can further be improved, which otherwise would waste valuable GPU cycles due to remaining dependencies within the Image AI pipeline or expensive communication overhead of partially computed gradients [70, 84].

5 Image vs Video Data

A natural question is how do images compare to video with respect to data storage and management. Conceptually, a video is a series of frames, where each frame is a separate image. Storing and managing video data, however, is significantly different than image data. Video data is composed of different types of frames, where each frame contains different types of data. I-frames are key frames offering lowest compression, whereas B- and P-frames are frames that can be composed of their preceding and/or following frames and offer higher compression. For this reason, video storage formats focus on utilizing sophisticated prediction and grouping algorithms to group different types of frames and compose B- and P-frames from their neighboring frames [69, 103, 113].

There is a large body of work on Video Data Management Systems (VDBMSs), where users perform analytical processing over video data using a SQL-like language. The main goal is to efficiently answer queries by query-specific AI-models/storage-formats [3, 12, 23, 32, 33, 56–63, 65, 78, 88, 95, 96, 98, 99, 104, 125], exploiting temporal relationship among video frames [7, 8, 18–22, 25–27, 46, 49, 66, 67, 82, 100, 109, 114–116], and/or exploiting workload locality among different queries [34, 35, 40, 45, 47, 89, 120, 124]. Tuning storage-related knobs, such as resolution of video frames, frame rate, and tile size for tile-based storage, relates to self-designed storage formats in terms of producing a storage for a specific task, where the task is answering a given SQL-like query.

6 Open Problems and Future Directions

This last section describes open problems and future directions for making image AI pipelines efficient. We first discuss how existing lines of research could push their boundaries by addressing overlooked challenges. Examples include constructing a design space of storage formats solely for a specific data type, such as medical images, expanding learned formats for dynamically sized image representations, and revisiting materialization techniques for novel persistent memory technologies.

We then examine how different lines of research, such as storage and data management pipelines, can intersect and shed light on further efficiency. Examples include creating a design space of storage formats where learned representations and self-designed representations are unified, revisiting materialization and reuse for concurrently serving diverse self-designed storage formats, and sharing activation maps and gradients across multi-task learned storage formats.

7 Audience and Outputs

Audience. The target audience for this tutorial is students, academics, researchers, and software engineers with basic knowledge of data systems. We assume a basic understanding of how hardware works, such as disk, CPU, GPU, and communication between CPU and GPU. We assume a basic understanding of data management concepts such as compression, intermediate result materialization and reusing, parallelism, and hardware utilization. We do not assume prior knowledge of image storage or machine learning.
The learning outcomes are as follows.

- Understanding how image AI pipelines work.
- Developing a data-centric view of image AI pipelines.
- Understanding how standard, learned, and self-designed formats store images.
- Understanding how image AI systems manage their data inside their pipelines by using compression, materialization & reuse, and parallelism.
- Exposure to open problems and future directions for making image AI systems efficient.

8 Presenters

Utku Sirin is a postdoctoral researcher at the Data Systems lab at Harvard University, advised by Stratos Idreos. Utku's work on the Image Calculator reimagines Image AI through self-designing AI storage, which always takes the best shape given the AI context and goals, bringing 10x speedup end-to-end. Utku was awarded the Microsoft Research PhD Fellowship in 2017 and the Swiss National Science Foundation Postdoctoral Fellowship in 2021 and 2023. Utku has also been a winner of the ACM SIGMOD Students Research Competition and a recipient of an IEEE ICDE best reviewer award. Before joining Harvard, Utku obtained his PhD from the Data-Intensive Applications and Systems lab at EPFL, advised by Anastasia Ailamaki on hardware-conscious data systems.
Stratos Idreos is a professor of Computer Science at Harvard University, where he leads the Data Systems lab. His research focuses on making it easy and even automatic to design workload and hardware-conscious data objects and data systems with applications on relational, NoSQL, data science, machine learning, and data exploration problems. Stratos was awarded the ACM SIGMOD Jim Gray Doctoral Dissertation award for his thesis on adaptive indexing and the 2011 ERCIM Cor Baayen award from the European Research Council on Informatics and Mathematics. He won the 2011 VLDB Challenges and Visions Best Paper award and the 'Best of Conference' selections at VLDB 2012 and SIGMOD 2017. In 2015, he was awarded the IEEE TCDE Rising Star Award from the IEEE Technical Committee on Data Engineering for his work on adaptive data systems. Stratos also receives the IBM Enterprise System Recognition Award, a Facebook Faculty Award, a NetApp Faculty Award, an NSF Career Award, a Department of Energy Early Career Award, and a Sloan Fellowship.

Acknowledgments

This work is partially funded by the Swiss National Science Foundation Postdoc Mobility scholarship P500PT_217934, and the USA Department of Energy project DE-SC0020200.

References

[1] Ahmed M. Abdelmoniem, Ahmed Elzanaty, Mohamed-Slim Alouini, and Marco Canini. 2021. An Efficient Statistical-based Gradient Compression Technique for Distributed Training Systems. In *MLSys*.

[2] Mark Adler, Thomas Boutell, John Bowler, Christian Brunschen, Adam M. Costello, Lee Daniel Crocker, Andreas Dilger, Oliver Fromme, Jean loup Gailly, Chris Herborth, Alex Jakulin, Neal Kettler, Tom Lane, Alexander Lehmann, Chris Lilley, Dave Martindale, Owen Mortensen, Keith S. Pickens, Robert P. Poole, Glenn Randers-Pehrson, Greg Roelofs, Willem van Schaik, Guy Schalnat, Paul Schmidt, Michael Stokes, Tim Wegner, and Jeremy Wohl. 2003. Portable Network Graphics (PNG) Specification (Second Edition). https://www.w3.org/TR/2003/REC-PNG-20031110 Accessed on December 19, 2024.

[3] Michael R. Anderson, Michael J. Cafarella, Germán Ros, and Thomas F. Wenisch. 2019. Physical Representation-Based Predicate Optimization for a Visual Analytics Database. In *ICDE*. 1466–1477.

[4] William B. Pennebaker and Joan L. Mitchell. 1993. *JPEG: Still Image Data Compression Standard*. Springer.

[5] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. 2017. End-to-end Optimized Image Compression. In *ICLR*.

[6] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. 2018. Variational Image Compression with A Scale Hyperprior. In *ICLR*.

[7] Jaeho Bang, Gaurav Tarlok Kakkar, Pramod Chunduri, Subrata Mitra, and Joy Arulraj. 2023. SEIDEN: Revisiting Query Processing in Video Database Systems. *Proc. VLDB Endow.* 16, 9 (2023), 2289–2301.

[8] Favyen Bastani, Songtao He, Arjun Balasingam, Karthik Gopalakrishnan, Mohammad Alizadeh, Hari Balakrishnan, Michael J. Cafarella, Tim Kraska, and Sam Madden. 2020. MIRIS: Fast Object Track Queries in Video. In *SIGMOD*. 1907–1921.

[9] Vasudev Bhaskaran and Konstantinos Konstantinides. 1995. *Image and Video Compression Standards*. Springer.

[10] Song Bian, Dacheng Li, Hongyi Wang, Eric Xing, and Shivaram Venkataraman. 2024. Does Compressing Activations Help Model Parallel Training?. In *MLSys*. 239–252.

[11] Matic Broz. 2024. How Many Pictures are There (2024): Statistics, Trends, and Forecasts. https://photutorial.com/photos-statistics/ Accessed on July 31, 2024.

[12] Jiashen Cao, Karan Sarkar, Ramyad Hadidi, Joy Arulraj, and Hyesoon Kim. 2022. FiGO: Fine-Grained Query Optimization in Video Analytics. In *SIGMOD*. 559–572.

[13] Lahiru D. Chamain, Sen-ching Samson Cheung, and Zhi Ding. 2019. Quannet: Joint Image Compression and Classification Over Channels with Limited Bandwidth. In *20th IEEE International Conference on Multimedia and Expo (ICME)*. 338–343.

[14] Lahiru D. Chamain and Zhi Ding. 2020. Improving Deep Learning Classification of JPEG2000 Images Over Bandlimited Networks. In *45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 4062–4066.

[15] Lahiru D. Chamain, Siyu Qi, and Zhi Ding. 2021. An End-to-End Learning Architecture for Efficient Image Encoding and Deep Learning. In *29th European Signal Processing Conference (EUSIPCO)*. 691–695.

[16] Lahiru D. Chamain, Siyu Qi, and Zhi Ding. 2022. End-to-End Image Classification and Compression With Variational Autoencoders. *IEEE Internet of Things Journal* 9, 21 (2022), 21916–21931.

[17] Lahiru D. Chamain, Fabien Racapé, Jean Bégaint, Akshay Pushparaja, and Simon Feltman. 2021. End-to-End Optimized Image Compression for Machines, A Study. In *31st Data Compression Conference (DCC)*. 163–172.

[18] Daren Chao, Kaiwen Chen, and Nick Koudas. 2023. SVQ-ACT: Querying for Actions over Videos. In *ICDE*. 3599–3602.

[19] Daren Chao, Yueting Chen, Nick Koudas, and Xiaohui Yu. 2023. Track Merging for Effective Video Query Processing. In *ICDE*. 164–176.

[20] Daren Chao, Yueting Chen, Nick Koudas, and Xiaohui Yu. 2024. Optimizing Video Queries with Declarative Clues. *Proc. VLDB Endow.* 17, 11 (2024), 3256–3268.

[21] Daren Chao and Nick Koudas. 2024. Querying For Actions Over Videos. In *EDBT*. 162–174.

[22] Daren Chao, Nick Koudas, and Xiaohui Yu. 2023. Marshalling Model Inference in Video Streams. In *ICDE*. 1379–1392.

[23] Daren Chao, Nick Koudas, Xiaohui Yu, and Yueting Chen. 2025. Ensembling Object Detectors for Effective Video Query Processing. In *EDBT*. 66–79.

[24] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. 2016. Training Deep Nets with Sublinear Memory Cost. *CoRR* abs/1604.06174 (2016).

[25] Yueting Chen, Nick Koudas, Xiaohui Yu, and Ziqiang Yu. 2022. Spatial and Temporal Constrained Ranked Retrieval over Videos. *Proc. VLDB Endow.* 15, 11 (2022), 3226–3239.

[26] Yueting Chen, Xiaohui Yu, and Nick Koudas. 2022. Ranked Window Query Retrieval over Video Repositories. In *ICDE*. 2776–2791.

[27] Yueting Chen, Xiaohui Yu, Nick Koudas, and Ziqiang Yu. 2021. Evaluating Temporal Queries Over Video Feeds. In *SIGMOD*. 287–299.

[28] Hyomin Choi and Ivan V. Bajić. 2018. Deep Feature Compression for Collaborative Object Detection. In *25th IEEE International Conference on Image Processing (ICIP)*. 3743–3747.

[29] Hyomin Choi and Ivan V. Bajić. 2021. Latent-Space Scalability for Multi-Task Collaborative Intelligence. In *28th IEEE International Conference on Image Processing (ICIP)*. 3562–3566.

[30] Hyomin Choi and Ivan V. Bajić. 2022. Scalable Image Coding for Humans and Machines. *IEEE Transactions on Image Processing* 31, 1 (2022), 2739–2754.

[31] Chooch. 2023. How to use Computer Vision AI for Detecting Workplace Hazards? https://www.chooch.com/blog/computer-vision-ai-safety-technology-to-detect-workplace-hazards Accessed on Nov 14, 2024.

[32] Pramod Chunduri, Jaeho Bang, Yao Lu, and Joy Arulraj. 2022. Zeus: Efficiently Localizing Actions in Videos using Reinforcement Learning. In *SIGMOD*. 545–558.

[33] Maureen Daum, Brandon Haynes, Dong He, Amrita Mazumdar, and Magdalena Balazinska. 2021. TASM: A Tile-Based Storage Manager for Video Analytics. In *ICDE*. 1775–1786.

[34] Maureen Daum, Enhao Zhang, Dong He, Magdalena Balazinska, Brandon Haynes, Ranjay Krishna, Apryle Craig, and Aaron Wirsing. 2022. VOCAL: Video Organization and Interactive Compositional AnaLytics. In *CIDR*.

[35] Maureen Daum, Enhao Zhang, Dong He, Stephen Mussmann, Brandon Haynes, Ranjay Krishna, and Magdalena Balazinska. 2023. VOCALExplore: Pay-as-You-Go Video Data Exploration and Model Building. *Proc. VLDB Endow.* 16, 13 (2023), 4188–4201.

[36] Benjamin Deguerre, Clément Chatelain, and Gilles Gasso. 2019. Fast Object Detection in Compressed JPEG Images. In *22nd IEEE Intelligent Transportation Systems Conference (ITSC)*. 333–338.

[37] Peter Deutsch. 1996. DEFLATE Compressed Data Format Specification version 1.3. https://datatracker.ietf.org/doc/html/rfc1951 Accessed on December 19, 2024.

[38] Max Ehrlich and Larry S. Davis. 2019. Deep Residual Learning in the JPEG Transform Domain. In *ICCV*. 3484–3493.

[39] Jingzhi Fang, Yanyan Shen, Yue Wang, and Lei Chen. 2021. ETO: Accelerating Optimization of DNN Operators by High-Performance Tensor Program Reuse. *Proc. VLDB Endow.* 15, 2 (2021), 183–195.

[40] Apurva Gandhi, Yuki Asada, Victor Fu, Advitya Gemawat, Lihao Zhang, Rathijit Sen, Carlo Curino, Jesús Camacho-Rodríguez, and Matteo Interlandi. 2023. The Tensor Data Platform: Towards an AI-centric Database System. In *CIDR*.

[41] Georgios Georgiadis. 2018. Accelerating Convolutional Neural Networks via Activation Map Compression. (2018), 7078–7088.

[42] Alice Gomstyn and Alexandra Jonker. 2023. What is Smart Farming? https://www.ibm.com/topics/smart-farming Accessed on Nov 14, 2024.

[43] Andreas Griewank and Andrea Walther. 2000. Algorithm 799: Revolve: An Implementation of Checkpointing for the Reverse or Adjoint Mode of Computational Differentiation. *ACM Trans. Math. Softw.* 26, 1 (2000), 19–45.

[44] Lionel Gueguen, Alex Sergeev, Ben Kadlec, Rosanne Liu, and Jason Yosinski. 2018. Faster Neural Networks Straight from JPEG. In *NeurIPS*. 3937–3948.

[45] Brandon Haynes, Maureen Daum, Dong He, Amrita Mazumdar, Magdalena Balazinska, Alvin Cheung, and Luis Ceze. 2021. VSS: A Storage System for Video Analytics. In *SIGMOD*. 685–696.

[46] Brandon Haynes, Amrita Mazumdar, Armin Alaghi, Magdalena Balazinska, Luis Ceze, and Alvin Cheung. 2018. LightDB: A DBMS for Virtual Reality Video. *Proc. VLDB Endow.* 11, 10 (2018), 1192–1205.

[47] Brandon Haynes, Amrita Mazumdar, Magdalena Balazinska, Luis Ceze, and Alvin Cheung. 2019. Visual Road: A Video Data Management Benchmark. In *SIGMOD*. 972–987.

[48] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. 770–778.

[49] Bo Hu, Peizhen Guo, and Wenjun Hu. 2022. Video-zilla: An Indexing Layer for Large-Scale Video Analytics. In *SIGMOD*. 1905–1919.

[50] Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Mia Xu Chen, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, Yonghui Wu, and Zhifeng Chen. 2019. *GPipe: Efficient Training of Giant Neural Networks Using Pipeline Parallelism*.

[51] Alexander Isenko, Ruben Mayer, Jeffrey Jedele, and Hans-Arno Jacobsen. 2022. Where Is My Training Bottleneck? Hidden Trade-Offs in Deep Learning Preprocessing Pipelines. In *SIGMOD*. 1825–1839.

[52] Paras Jain, Ajay Jain, Aniruddha Nrusimha, Amir Gholami, Pieter Abbeel, Joseph Gonzalez, Kurt Keutzer, and Ion Stoica. 2020. Checkmate: Breaking the Memory Wall with Optimal Tensor Rematerialization. In *MLSys*. 497–511.

[53] Zhihao Jia, Matei Zaharia, and Alex Aiken. 2019. Beyond Data and Model Parallelism for Deep Neural Networks. In *MLSys*.

[54] Sian Jin, Chengming Zhang, Xintong Jiang, Yunhe Feng, Hui Guan, Guanpeng Li, Shuaiwen Leon Song, and Dingwen Tao. 2021. COMET: A Novel Memory-Efficient Deep Learning Training Framework by Using Error-Bounded Lossy Compression. *Proc. VLDB Endow.* 15, 4 (2021), 886–899.

[55] JPEG. 2023. JPEG 2000. https://jpeg.org/jpeg2000/ Accessed on Jan. 02, 2023.

[56] Gaurav Tarlok Kakkar, Jiashen Cao, Pramod Chunduri, Zhuangdi Xu, Suryatej Reddy Vyalla, Prashanth Dintyala, Anirudh Prabakaran, Jaeho Bang, Aubhro

Sengupta, Kaushik Ravichandran, Ishwarya Sivakumar, Aryan Rajoria, Ashmita Raju, Tushar Aggarwal, Abdullah Shah, Sanjana Garg, Shashank Suman, Myna Prasanna Kalluraya, Subrata Mitra, Ali Payani, Yao Lu, Umakishore Ramachandran, and Joy Arulraj. 2023. EVA: An End-to-End Exploratory Video Analytics System. In *DEEM Workshop*. Article 8.

[57] Daniel Kang, Peter Bailis, and Matei Zaharia. 2019. BlazeIt: Optimizing Declarative Aggregation and Limit Queries for Neural Network-based Video Analytics. *Proc. VLDB Endow.* 13, 4 (2019), 533–546.

[58] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing Neural Network Queries over Video at Scale. *Proc. VLDB Endow.* 10, 11 (2017), 1586–1597.

[59] Daniel Kang, Edward Gan, Peter Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2020. Approximate Selection with Guarantees Using Proxies. *Proc. VLDB Endow.* 13, 12 (2020), 1990–2003.

[60] Daniel Kang, John Guibas, Peter Bailis, Tatsunori Hashimoto, Yi Sun, and Matei Zaharia. 2021. Accelerating Approximate Aggregation Queries with Expensive Predicates. *Proc. VLDB Endow.* 14, 11 (2021), 2341–2354.

[61] Daniel Kang, John Guibas, Peter D. Bailis, Tatsunori Hashimoto, and Matei Zaharia. 2022. TASTI: Semantic Indexes for Machine Learning-based Queries over Unstructured Data. In *SIGMOD*. 1934–1947.

[62] Daniel Kang, Ankit Mathur, Teja Veeramacheneni, Peter Bailis, and Matei Zaharia. 2020. Jointly Optimizing Preprocessing and Inference for DNN-Based Visual Analytics. *Proc. VLDB Endow.* 14, 2 (2020), 87–100.

[63] Daniel Kang, Francisco Romero, Peter D. Bailis, Christos Kozyrakis, and Matei Zaharia. 2022. VIVA: An End-to-End System for Interactive Video Analytics. In *CIDR*.

[64] Alexandros Koliousis, Pijika Watcharapichat, Matthias Weidlich, Luo Mai, Paolo Costa, and Peter Pietzuch. 2019. Crossbow: Scaling Deep Learning with Small Batch Sizes on Multi-GPU Servers. *Proc. VLDB Endow.* 12, 11 (2019), 1399–1412.

[65] Ferdinand Kossmann, Ziniu Wu, Eugenie Lai, Nesime Tatbul, Lei Cao, Tim Kraska, and Sam Madden. 2023. Extract-Transform-Load for Video Streams. *Proc. VLDB Endow.* 16, 9 (2023), 2302–2315.

[66] Nick Koudas, Raymond Li, and Ioannis Xarchakos. 2020. Video Monitoring Queries. In *ICDE*. 1285–1296.

[67] Nick Koudas, Raymond Li, and Ioannis Xarchakos. 2022. Video Monitoring Queries. *IEEE Transactions on Knowledge and Data Engineering* 34, 10 (2022), 5023–5036.

[68] Joel Lamy-Poirier. 2023. Breadth-First Pipeline Parallelism. In *MLSys*. 48–67.

[69] Didier Le Gall. 1991. MPEG: A Video Compression Standard for Multimedia Applications. *Commun. ACM* 34, 4 (1991), 46–58.

[70] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. 2020. A System for Massively Parallel Hyperparameter Tuning. In *MLSys*. 230–246.

[71] Hyeontaek Lim, David G Andersen, and Michael Kaminsky. 2019. 3LC: Lightweight and Effective Traffic Compression for Distributed Machine Learning. In *MLSys*. 53–64.

[72] Jinming Liu, Heming Sun, and Jiro Katto. 2021. Learning in Compressed Domain for Faster Machine Vision Tasks. In *36th International Conference on Visual Communications and Image Processing (VCIP)*. 1–5.

[73] Jinming Liu, Heming Sun, and Jiro Katto. 2022. Improving Multiple Machine Vision Tasks in the Compressed Domain. In *26th International Conference on Pattern Recognition (ICPR)*. 331–337.

[74] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *ICCV*. IEEE, 9992–10002.

[75] Zihao Liu, Tao Liu, Wujie Wen, Lei Jiang, Jie Xu, Yanzhi Wang, and Gang Quan. 2018. DeepN-JPEG: A Deep Neural Network Favorable JPEG-Based Image Compression Framework. In *55th Annual Design Automation Conference (DAC)*. 1–6.

[76] Sangkug Lym, Armand Behroozi, Wei Wen, Ge Li, Yongkee Kwon, and Mattan Erez. 2019. Mini-batch Serialization: CNN Training with Inter-layer Data Reuse. In *MLSys*.

[77] Ilia Markov, Kaveh Alim, Elias Frantar, and Dan Alistarh. 2024. L-GreCo: Layerwise-adaptive Gradient Compression For Efficient Data-parallel Deep Learning. In *MLSys*.

[78] Amrita Mazumdar, Brandon Haynes, Magda Balazinska, Luis Ceze, Alvin Cheung, and Mark Oskin. 2019. Perceptual Compression for Video Storage and Processing Systems. In *SoCC*. 179–192.

[79] Xupeng Miao, Xiaonan Nie, Yingxia Shao, Zhi Yang, Jiawei Jiang, Lingxiao Ma, and Bin Cui. 2021. Heterogeneity-Aware Distributed Machine Learning Training via Partial Reduce. In *SIGMOD*. 2262–2270.

[80] David Minnen, Johannes Ballé, and George D Toderici. 2018. Joint Autoregressive and Hierarchical Priors for Learned Image Compression. In *NeurIPS*.

[81] Azalia Mirhoseini, Hieu Pham, Quoc V. Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. 2017. Device Placement Optimization with Reinforcement Learning. In *ICML*. 2430–2439.

[82] Oscar Moll, Favyen Bastani, Sam Madden, Mike Stonebraker, Vijay Gadepally, and Tim Kraska. 2022. ExSample: Efficient Searches on Video Repositories through Adaptive Sampling. In *ICDE*. 3065–3077.

[83] Supun Nakandala and Arun Kumar. 2022. Nautilus: An Optimized System for Deep Transfer Learning over Evolving Training Datasets. In *SIGMOD*. 506–520.

[84] Supun Nakandala, Yuhao Zhang, and Arun Kumar. 2020. Cerebro: A Data System for Optimized Deep Learning Model Selection. *Proc. VLDB Endow.* 13, 12 (2020), 2159–2173.

[85] Deepak Narayanan, Aaron Harlap, Amar Phanishayee, Vivek Seshadri, Nikhil R. Devanur, Gregory R. Ganger, Phillip B. Gibbons, and Matei Zaharia. 2019. PipeDream: Generalized Pipeline Parallelism for DNN Training. In *SOSP*. 1–15.

[86] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. 2021. Efficient Large-scale Language Model Training on GPU Clusters Using Megatron-LM. In *SC*.

[87] Xiaonan Nie, Xupeng Miao, Zilong Wang, Zichao Yang, Jilong Xue, Lingxiao Ma, Gang Cao, and Bin Cui. 2023. FlexMoE: Scaling Large-Scale Sparse Pre-Trained Model Training via Dynamic Device Placement. *Proc. ACM Manag. Data* 1, 1 (2023), 19 pages.

[88] John Paparrizos, Chunwei Liu, Bruno Barbarioli, Johnny Hwang, Ikraduya Edian, Aaron J. Elmore, Michael J. Franklin, and Sanjay Krishnan. 2021. VergeDB: A Database for IoT Analytics on Edge Devices. In *CIDR*.

[89] Kwanghyun Park, Karla Saur, Dalitso Banda, Rathijit Sen, Matteo Interlandi, and Konstantinos Karanasos. 2022. End-to-End Optimization of Machine Learning Prediction Queries. In *SIGMOD*. 587–601.

[90] Seo Jin Park, Joshua Fried, Sunghyun Kim, Mohammad Alizadeh, and Adam Belay. 2022. Efficient Strong Scaling Through Burst Parallel Training. In *MLSys*.

[91] Xuan Peng, Xuanhua Shi, Hulin Dai, Hai Jin, Weiliang Ma, Qian Xiong, Fan Yang, and Xuehai Qian. 2020. Capuchin: Tensor-based GPU Memory Management for Deep Learning. In *ASPLOS*. 891–905.

[92] Sanket Purandare, Abdul Wasay, Animesh Jain, and Stratos Idreos. 2023. μ-TWO: 3x Faster Multi-Model Training with Orchestration and Memory Optimization. In *MLSys*. 541–562.

[93] Alexander Renz-Wieland, Rainer Gemulla, Zoi Kaoudi, and Volker Markl. 2022. NuPS: A Parameter Server for Machine Learning with Non-Uniform Parameter Access. In *SIGMOD*. 481–495.

[94] Alexander Renz-Wieland, Rainer Gemulla, Steffen Zeuch, and Volker Markl. 2020. Dynamic Parameter Allocation in Parameter Servers. *Proc. VLDB Endow.* 13, 12 (2020), 1877–1890.

[95] Francisco Romero, Johann Hauswald, Aditi Partap, Daniel Kang, Matei Zaharia, and Christos Kozyrakis. 2022. Optimizing Video Analytics with Declarative Model Relationships. *Proc. VLDB Endow.* 16, 3 (2022), 447–460.

[96] Matthew Russo, Tatsunori Hashimoto, Daniel Kang, Yi Sun, and Matei Zaharia. 2023. Accelerating Aggregation Queries on Unstructured Streams of Data. *Proc. VLDB Endow.* 16, 11 (2023), 2897–2910.

[97] Samuel Felipe dos Santos, Nicu Sebe, and Jurandy Almeida. 2020. The Good, The Bad, and The Ugly: Neural Networks Straight From JPEG. In *27th IEEE International Conference on Image Processing (ICIP)*. 1896–1900.

[98] Ted Shaowang, Nilesh Jain, Dennis Matthews, and Sanjay Krishnan. 2021. Declarative Data Serving: The Future of Machine Learning Inference on the Edge. *Proc. VLDB Endow.* 14, 11 (2021), 2555–2562.

[99] Ted Shaowang, Xi Liang, and Sanjay Krishnan. 2022. Sensor Fusion on the Edge: Initial Experiments in the EdgeServe System. In *BiDEDE Workshop*.

[100] Ted Shaowang, Jinjin Zhao, Stavros Sintos, and Sanjay Krishnan. 2022. Towards Causal Physical Error Discovery in Video Analytics Systems. In *HILDA Workshop*.

[101] Utku Sirin and Stratos Idreos. 2024. The Image Calculator: 10x Faster Image-AI Inference by Replacing JPEG with Self-designing Storage Format. *Proc. ACM Manag. Data* 2, 1, Article 52 (2024).

[102] Utku Sirin, Victoria Kauffman, Aadit Saluja, Florian Klein, Jeremy Hsu, and Stratos Idreos. 2025. Frequency-Store: Scaling Image AI by A Column-Store for Image. In *CIDR*.

[103] Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han, and Thomas Wiegand. 2012. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 22, 12 (2012), 1649–1668.

[104] Abhijit Suprem, Joy Arulraj, Calton Pu, and Joao Ferreira. 2020. ODIN: Automated Drift Detection and Recovery in Video Analytics. *Proc. VLDB Endow.* 13, 12 (2020), 2453–2465.

[105] Peter Symes. 1998. *Video Compression: Fundamental Compression Techniques and an Overview of the JPEG and MPEG Compression Systems.* McGraw-Hill.

[106] Róbert Torfason, Fabian Mentzer, Eiríkur Ágústsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. 2018. Towards Image Understanding from Deep Compression Without Decoding. In *ICLR*. 1–17.

[107] Minh Hoang Vu, Anders Garpebring, Tufve Nyholm, and Tommy Löfstedt. 2024. Compressing the Activation Maps in Deep Convolutional Neural Networks and Its Regularizing Effect. *Transactions on Machine Learning Research* (2024).

[108] G.K. Wallace. 1992. The JPEG Still Picture Compression Standard. *IEEE Transactions on Consumer Electronics* 38, 1 (1992), xviii–xxxiv.

[109] Jingjing Wang and Magdalena Balazinska. 2020. Deluceva: Delta-Based Neural Network Inference for Fast Video Analytics. In *SSDBM*. Article 14.
[110] Shang Wang, Yifan Bai, and Gennady Pekhimenko. 2020. BPPSA: Scaling Back-propagation by Parallel Scan Algorithm. In *MLSys*. 451–469.
[111] Zhenzhen Wang, Minghai Qin, and Yen-Kuang Chen. 2022. Learning From the CNN-based Compressed Domain. In *22nd IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 4000–4008.
[112] Zhuang Wang, Xinyu Wu, Zhaozhuo Xu, and T. S. Eugene Ng. 2023. Cupcake: A Compression Scheduler for Scalable Communication-Efficient Distributed Training. In *MLSys*. 373–386.
[113] T. Wiegand, G.J. Sullivan, G. Bjontegaard, and A. Luthra. 2003. Overview of the H.264/AVC Video Coding Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 13, 7 (2003), 560–576.
[114] Renzhi Wu, Pramod Chunduri, Ali Payani, Xu Chu, Joy Arulraj, and Kexin Rong. 2024. SketchQL: Video Moment Querying with a Visual Query Interface. *Proc. ACM Manag. Data* 2, 4 (2024).
[115] Ioannis Xarchakos and Nick Koudas. 2023. Querying for Interactions. *IEEE Transactions on Knowledge and Data Engineering* 35, 2 (2023), 1977–1990.
[116] Ioannis Xarchakos and Nick Koudas. 2025. Coping With Data Drift in Online Video Analytics. In *EDBT*. 39–52.
[117] Ningning Xie, Tamara Norman, Dominik Grewe, and Dimitrios Vytiniotis. 2022. Synthesizing Optimal Parallelism Placement and Reduction Strategies on Hierarchical Systems for Deep Learning. In *MLSys*. 548–566.
[118] Kai Xu. 2021. *Learning in Compressed Domains*. Ph. D. Dissertation. Arizona State University.
[119] Kai Xu, Minghai Qin, Fei Sun, Yuhao Wang, Yen-Kuang Chen, and Fengbo Ren. 2020. Learning in the Frequency Domain. In *CVPR*. 1740–1749.
[120] Zhuangdi Xu, Gaurav Tarlok Kakkar, Joy Arulraj, and Umakishore Ramachandran. 2022. EVA: A Symbolic Approach to Accelerating Exploratory Video Analytics with Materialized Views. In *SIGMOD*. 602–616.
[121] Bowen Yang, Jian Zhang, Jonathan Li, Christopher Re, Christopher Aberger, and Christopher De Sa. 2021. PipeMare: Asynchronous Pipeline Parallel DNN Training. In *MLSys*. 269–296.
[122] Gang Yu, Kai Sun, Chao Xu, Xing-Hua Shi, Chong Wu, Ting Xie, Run-Qi Meng, Xiang-He Meng, Kuan-Song Wang, Hong-Mei Xiao, and Hong-Wen Deng. 2021. Accurate Recognition of Colorectal Cancer with Semi-supervised Deep Learning on Pathological Images. *Nature Communications* 12, 6311 (2021).
[123] Binhang Yuan, Cameron R. Wolfe, Chen Dun, Yuxin Tang, Anastasios Kyrillidis, and Chris Jermaine. 2022. Distributed Learning of Fully Connected Neural Networks Using Independent Subnet Training. *Proc. VLDB Endow.* 15, 8 (2022), 1581–1590.
[124] Xin Zhang, Yanyan Shen, Yingxia Shao, and Lei Chen. 2023. DUCATI: A Dual-Cache Training System for Graph Neural Networks on Giant Graphs with the GPU. *Proc. ACM Manag. Data* 1, 2 (2023).
[125] Yuhao Zhang and Arun Kumar. 2019. Panorama: A Data System for Unbounded Vocabulary Querying over Video. *Proc. VLDB Endow.* 13, 4 (2019), 477–491.
[126] Yonghao Zhuang, Lianmin Zheng, Zhuohan Li, Eric P. Xing, Qirong Ho, Joseph Gonzalez, Ion Stoica, Hao Zhang, and Hexu Zhao. 2023. On Optimizing the Communication of Model Parallelism. In *MLSys*.