Predicate Optimization for a Visual Analytics Database

Michael R. Anderson, Michael Cafarella, Thomas F. Wenisch
University of Michigan
{mrander,michjc,twenisch}@umich.edu

German Ros
Toyota Research Institute
german.ros@tri.global

ABSTRACT

Querying the content of images, video, and other non-textual data sources requires expensive content extraction methods. Modern extraction techniques are based on deep convolutional neural networks (CNNs) and can classify objects within images with astounding accuracy. Unfortunately, these methods are slow, needing several milliseconds per image using modern GPUs. The cost of content-based queries over a huge video corpus is prohibitive.

A promising approach to reduce the runtime cost of queries of visual content is to use a hierarchical model, such as a cascade, where simple cases are handled by an inexpensive classifier. Prior work has sought to design cascades that optimize the computational cost of inference by, for example, using smaller CNNs. However, we observe that there are critical factors besides the inference time that dramatically impact the overall query time. Notably, by treating the input image format and the requisite data handling costs as part of our query optimization, we can enable much more efficient cascades. We find that by jointly optimizing the CNN architecture and input representation, we can provide up to a 40x speedup over the cascades used in the NoScope video query system. We find up to a 156x speedup over ResNet with no accuracy loss and a nearly 300x speedup for users willing to sacrificing some accuracy.

1 INTRODUCTION

Recent developments in computer vision have made feasible a long-term dream for the database community: a *visual analytics database*, which stores image data and answers user questions about its contents. For example, video from a city's traffic cameras could be used to count cars per minute or construction trucks per hour. Video from a fleet of self-driving cars could be used to compute statistics about other drivers on the road, or even wildlife alongside the highway. The sheer volume and diversity of data that can be captured by cameras opens up a huge range of analytical query possibilities. A simple but very useful visual analytics data system would offer a standard SQL query interface over stored image data with selection predicates that can accurately test image contents.

Deep convolutional neural networks (CNNs)—the family of methods employed in modern computer vision systems—have enabled huge strides in image understanding in the last few years. Unfortunately, deep networks pose a considerable computational challenge when deployed in an analytical database system: classifying a single image can require a lengthy series of large tensor multiplications. Since GPU hardware is generally far more expensive than image sensors, any data system that requires most of a dedicated GPU to process a single stream of video will never process more than a fraction of the video produced in real-world, multi-camera applications. To process massive amounts of video at reasonable cost, we need drastically lower costs for query processing.

There has been some recent work in optimizing the execution time of these image classifiers (e.g., [1, 5, 6]). However, we note that all of the visual data system optimizations to date suffer from a critical defect: they concentrate only on computation and ignore the inevitable data-handling costs, such as loading, marshalling, and transformation. Any query optimization method that focuses solely on reducing computational load cannot exploit complex *data-centric tradeoffs* that weigh the amount of image data available, the classifier accuracy, and data loading and transformation costs.

As an example, consider an optimizer choosing between two CNN models (M_1 and M_2). M_1 uses a 3-channel, full-color 224x224 pixel image as input while M_2 uses a 1-channel grayscale 224x224 pixel image. M_1 has fewer convolutional layers and—despite the larger input—requires fewer tensor operations than M_2, so its inference is faster. M_2 uses less rich data than M_1, but has comparable accuracy due to its additional layers. An optimizer considering only inference speed would choose M_1. However, a data system based on M_2 might be faster, as its smaller inputs are loaded more quickly.

Such data-handling tradeoffs are particularly important because visual analytics systems are likely to be diverse architecturally. Some systems might store multiple versions of the same image data (high-res vs. low-res or color vs. black-and-white). Others might employ different storage systems (local server vs. cloud vs. in-camera storage). Ignoring data-centric tradeoffs in such deployment scenarios sacrifices substantial potential performance.

In this work, we propose a framework for handling data-centric tradeoffs when optimizing visual analytical queries, focusing exclusively on the CNN-based operator that implements an image-sensitive relational predicate.

Our approach — One method of accelerating an expensive-but-accurate CNN is to replace it with cascades of fast, high precision, low recall classifiers [2, 6, 8]. This method is effective but focuses on computational efficiency. Prior to any query processing, we start with a similar approach, training a large number of simple candidate binary-classification CNN models. We expand the candidate space by varying not only CNN hyperparameters (as in prior work [6]), but also the representation of the inputs; for example, we build an n-layer CNN for high-resolution full-color inputs, one for low-resolution full-color inputs, another for grayscale inputs, etc.

From these core candidate models, we then construct a massive number of classifier cascades. All of these cascades have different (and initially unknown) runtime and accuracy characteristics. Our optimization method efficiently evaluates the cascades' accuracy using held-out data, and evaluates their runtime characteristics for the system's current deployment scenario. Finally, it attempts to identify the cascades that are Pareto optimal and should thereby be presented to the user as reasonable options for satisfying the application-specific speed and accuracy constraints.

2 METHODOLOGY

A classifier cascade is a series of classification models run one after another until a trusted classification result is found. Our methods depend upon building a large number of models and combining them to create a huge number of cascades. We parameterize the models in two ways: by varying the architecture of our CNN classifiers and by performing transformations on the input images.

Model architecture variations — Tahoma uses standard convolutional neural networks for its models. Each convolutional layer is followed by a max pooling layer, connected by rectified linear activations (ReLu). The final convolutional layer feeds into a fully connected ReLu layer. A single sigmoid output node provides the inferred label for our binary query task. When creating our set of models, we vary the size and number of the network layers.

Input transformations — We also vary the representation of the input to each model. The set of input transformation functions comprises functions that perform one or more image processing operations, such as image resolution scaling and color channel modifications. These types of transformations are especially useful towards our goal of building fast, small models: reducing image size and color depth reduces number of model input values, directly reducing the number of the tensor operations within the CNN.

The design space defined by these model architecture variations and input transformation functions result in hundreds of different model configurations. Once each model is trained on a labeled subset of images, we can compose the models into cascades. Training takes less than a minute for the smallest networks with the smallest inputs and nearly an hour for the largest. While training all models can take upwards of 12 hours, it is highly parallelizable.

3 EXPERIMENTS

We have implemented Tahoma as a prototype system, for which full experimental results are presented in a paper currently under submission. Here we present several key results that demonstrate the effectiveness of our system in accelerating image classification in a binary predicate setting.

Experimental Setup — To evaluate the methods and components used in Tahoma, we designed a series of experiments using a set of 10 queries with a single *contains-object* binary predicate, chosen from the 1,000 categories in the ImageNet dataset [7]. A portion of the ImageNet images comprised a holdout set for cascade configuration. We evaluated performance using non-ImageNet images.

We used the Keras [3] deep learning library to train and execute our CNNs. We varied several network architecture hyperparameters to create a range of models for each binary predicate: number of convolutional layers and the number of nodes in each layer. We also varied the size of the input images, and for each of the four image sizes, we used five different image representations: 3-channel color, each individual color channel, and single-channel grayscale. Overall, we created 1,301,405 possible cascades per predicate.

Selected Results — We compared Tahoma against a pre-trained ResNet implementation [4], as well as CNN cascade designs proposed by NoScope [6], which comprise a subset of Tahoma's design space. In addition to its cascade designs, NoScope proposes several optimization methods to skip CNN classification of video frames altogether; these methods are orthogonal to ours, but could be applied to Tahoma in a video setting, further increasing speedup. Here, we compare the CNN cascade designs of NoScope and Tahoma.

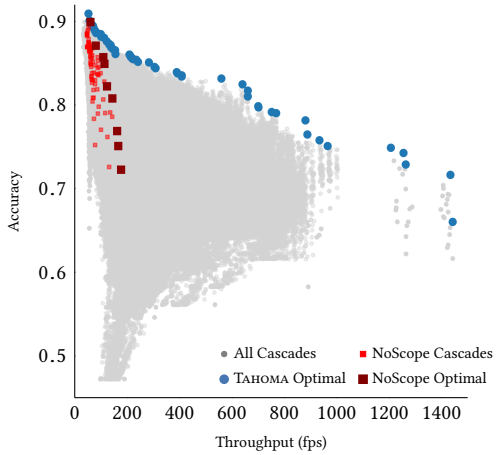

Figure 1: An comparison of the cascade spaces of Tahoma and NoScope. Gray points show all cascades generated by Tahoma, with Pareto optimal points in blue. Cascades that correspond to NoScope configurations are shown in red.

NoScope's cascades share the same objectives as ours, but encompass a much smaller design space: a single simple classifier is followed by a maximally accurate full-cost classifier, with no transforms performed on the input representation. Within our design space, two-level cascades that terminate in a full-cost classifier (i.e., ResNet) and use full-color 224x224 images as input correspond to NoScope's design space. An illustration of the difference in the design spaces is shown in Figure 1.

We compared the performance of Tahoma compared with our baselines under four different cost models that capture different data handling cost models, each representing realistic deployment scenarios with different combinations of data loading, transformation, and inference costs. We found that Tahoma showed speedups ranging from 2x to 40x over NoScope under these models. Compared to ResNet, Tahoma showed up to a 156x speed up with no loss of accuracy. In situations where a loss in accuracy is tolerable, Tahoma achieved an average throughput of 22,343 frames per second—nearly 300 times the throughput of ResNet.

4 CONCLUSION AND FUTURE WORK

In this paper, we have presented a method of accelerating content extraction from large corpora of visual data, with the aim of support visual analytics query. We showed how constructing a huge number of classifier cascades from a wide variety of CNN-based classification models can yield large speedups in content extraction.

While this paper primarily focused on image classification tasks, it is just the beginning of the development of an analytics system for visual data that will take full advantage of spatio-temporal locality present in adjacent video frames to further accelerate content extraction from video. We hope to include new state-of-the-art computer vision methods to extract more complex data, which will then allow the processing of complex analytical queries over video.

REFERENCES

[1] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *KDD*.

[2] Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. 2015. Learning complexity-aware cascades for deep pedestrian detection. In *ICCV*.

[3] François Chollet et al. 2015. Keras. https://github.com/fchollet/keras. (2015).

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

[5] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2014. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*.

[6] Daniel Kang, John Emmons, Firas Abuzaid, Peter Bailis, and Matei Zaharia. 2017. NoScope: Optimizing neural network queries over video at scale. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1586–1597.

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252.

[8] Yi Sun, Xiaogang Wang, and Xiaoou Tang. 2013. Deep convolutional network cascade for facial point detection. In *CVPR*.