Panorama: A Data System for Unbounded Vocabulary Querying over Video

Yuhao Zhang
University of California, San Diego
yuz870@eng.ucsd.edu

Arun Kumar
University of California, San Diego
arunkk@eng.ucsd.edu

ABSTRACT

Deep convolutional neural networks (CNNs) achieve state-of-the-art accuracy for many computer vision tasks. But using them for video monitoring applications incurs high computational cost and inference latency. Thus, recent works have studied how to improve system efficiency. But they largely focus on small "closed world" prediction vocabularies even though many applications in surveillance security, traffic analytics, etc. have an ever-growing set of target entities. We call this the "unbounded vocabulary" issue, and it is a key bottleneck for emerging video monitoring applications. We present the first data system for tacking this issue for video querying, Panorama. Our design philosophy is to build a unified and domain-agnostic system that lets application users generalize to unbounded vocabularies in an out-of-the-box manner without tedious manual re-training. To this end, we synthesize and innovate upon an array of techniques from the ML, vision, databases, and multimedia systems literature to devise a new system architecture. We also present techniques to ensure Panorama has high inference efficiency. Experiments with multiple real-world datasets show that Panorama can achieve between 2x to 20x higher efficiency than baseline approaches on in-vocabulary queries, while still yielding comparable accuracy and also generalizing well to unbounded vocabularies.

1. INTRODUCTION

Videos are a ubiquitous and growing fraction of real-world data. For instance, YouTube alone gets 100s of Petabytes of videos each year [9]. Thus, real-time *video monitoring* applications involving automatic recognition of objects in videos are gaining importance in many domains, including surveillance security [1], crowd control [2], traffic analytics, species monitoring, and more. The state-of-the-art approach for visual recognition is to use deep convolutional neural networks (CNNs) [24, 40]. However, deep CNNs are compute-intensive and have high inference latency, e.g., the popular Mask-RCNN [26] takes 1s for just five frames. Thus, enabling efficient visual recognition queries over video streams is a pressing data systems challenge.

Several recent lines of work in the multimedia, database, and systems communities aim to build more efficient systems for real-time video querying [66, 34, 62, 27, 33]. A common theme is to reduce CNN inference latency with cheaper models with smaller prediction *vocabularies* and using "cascades" of classifiers. But from conversations with video monitoring application users across domains such as surveillance and species monitoring, we find a pressing gap in the existing landscape of systems: *unbounded vocabulary*.

Problem: Unbounded Vocabulary. Almost all popular object recognition CNNs today handle only a finite "closed world" vocabulary of known targets. This is a natural consequence of their training dataset, typically a benchmark dataset like ImageNet [53], Pascal VOC [18], or MS COCO [43]. For instance, Pascal VOC has a tiny vocabulary of only 20 classes, e.g., "person," "bird," and "car." So, models trained on it only tell apart these 20 classes. This may suffice for some applications that only need to tell apart these classes (e.g., for self-driving cars), but for many emerging video monitoring applications, the query requirements are *more granular*: "Who is this person?," "What car model is this?," "What bird species is this?," and so on. In these applications, the target class set is not universally fixed and finite but rather growing over time, sometimes rapidly. For instance, the set of all people or all car models evolves. We call such a prediction target with a fast-evolving set of objects an *unbounded vocabulary*. Note the vocabulary needs to be the sub-classes of a common class, sometimes also known as "subclassing".

Example. Consider a CNN trained to tell apart dog breeds. Suppose its training dataset had a vocabulary of only three popular breeds: `Corgi`, `Labrador`, and `Poodle`. What will it output on an image of a rare dog breed, say, `Azawakh`? It will output junk probabilities for `Corgi`, `Labrador`, and `Poodle`. Of course, this not an issue with the model but rather its prediction vocabulary–a limited multi-class vocabulary is too restrictive. One might ask: Why not get labeled data for all possible classes? Apart from being impractical, such an approach also assumes new dog breeds will not arise. This is a fundamental issue for such applications: the prediction vocabulary is effectively unbounded. This issue is even starker for identifying faces in videos, e.g., for surveillance security, because it is impossible to get labels for all possible faces beforehand; furthermore, the set of faces is not bounded because new people will keep appearing.

Limitations of Existing Landscape. We see two main limitations. First, existing querying systems do not support unbounded vocabularies. Thus, their architectural assumptions and modeling choices need to be revisited. While the ML community has studied learning schemes to support new class labels, e.g., one-shot and zero-shot learning [39, 37], using them requires tedious manual intervention to retrain the model and provide metadata and/or more labels. This needs ML expertise, but video monitoring applications

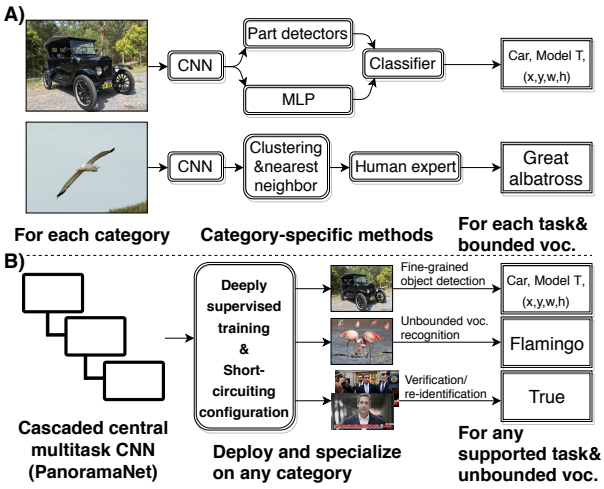

Figure 1: High-level qualitative comparison of existing vision stacks to Panorama's system design philosophy. (A) Each domain has a bespoke pipeline and a finite vocabulary. (B) Panorama enables unbounded vocabulary querying with a unified domain-agnostic data system that is automatically specialized for a given domain.

typically have only non-technical domain users in the loop of a deployed system (e.g., mall security). Second, making existing CNN-based stacks support unbounded vocabularies is not practical because they are often too application-specific and may involve bespoke pipelines, as illustrated by Figure 1(A). Such an ad hoc per-domain approach will duplicate the efforts of building, testing, and maintaining this capability across domains. Overall, the lack of support for unbounded vocabularies in a unified domain-agnostic data system is a bottleneck for emerging video applications.

System Desiderata. We have three related desiderata for a practical data system to support unbounded vocabulary queries over video. (1) Generalizing to new classes in the domain's vocabulary in an *automatic* manner without manual ML re-training. (2) Being *unified and domain-agnostic* to enable existing applications to adopt it without expensive manual customization. (3) Being *resource-efficient* and ideally real-time.

Our Proposed System. *We present Panorama, the first information system architecture for unbounded vocabulary queries over video.* It supports two kinds of queries popular in video monitoring. First is *recognition*: identify which *known object* (or set of objects) appear in a video feed (or an image database), e.g., a mall security officer checks a video feed against an image roster of wanted criminals to spot them in the crowd. Second is *verification*: tell if two frames (or images) have the same object in them regardless of whether the object is known, e.g., the officer compares an old frame with the current video feed to see if anyone reappeared. Our system design philosophy, illustrated by Figure 1(B), is to devise a unified and domain-agnostic system that can be automatically specialized for a given application.

Summary of Our Techniques. Panorama has three main components as Figure 1(B) shows: a new unified CNN architecture we call PanoramaNet, an automated offline training pipeline, and an online inference subsystem. PanoramaNet

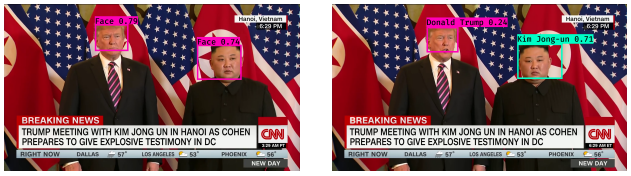

Figure 2: Example Panorama use-case (1): unbounded vocabulary recognition. Left: the frame shows two out-of-vocabulary faces (identities unknown) and the model only labels them as faces. Right: After the user freezes the video, clicks on the bounding boxes and labels them with names, Panorama can recognize these two objects in future frames. The scores shown on the left frame are the probabilities of being faces; on the right frame are the distances from the faces to their nearest neighbor in the *Known Objects* set.

is a careful synthesis of three techniques (Section 4.1): multi-task learning from ML, embedding extraction from vision, and short-circuiting of inference from data systems. It helps meet desiderata (1) and (2). Our automated offline training pipeline is a synthesis of deep supervision (Section 4.2) and weak supervision (Section 4.3) ideas from ML. It helps meet desideratum (2). Finally, our online inference subsystem features a novel short-circuiting configuration technique (Section 4.4) that enables a tunable accuracy-efficiency tradeoff and a synthesis of nearest neighbor search from multimedia systems and query caching from databases to improve efficiency (Section 4.5). It helps meet desideratum (3).

Example Use-Cases. We present two example use-cases to highlight Panorama's functionalities. Section 3 presents the full query API of Panorama and a usage example. Note the user interface is application-specific and orthogonal to our work, here we only show some possibilities of custom-built interfaces.

(1) *Unbounded vocabulary recognition.* Figure 2 shows, say, a journalist spotting people in a news video feed. The vocabulary did *not* have Donald Trump or Kim Jong-Un to start with. Our system constantly detects faces and extracts embeddings from them. In this case, the journalist can select the bounding boxes and type in names for these two faces and their corresponding embeddings, respectively. Once it is done, these named embeddings are added into the known objects which serves as the vocabulary. From then on Panorama can recognize Donald Trump or Kim Jong-Un. This entire process happens on-the-fly and without any re-training of the CNN.

(2) *Unbounded vocabulary embeddings.* Panorama can also output object embeddings (vectors) for further analyses. Figure 3 shows, say, a data scientist analyzing the representation of racial and gender groups in an Oscars video feed. The user runs an off-the-shelf clustering algorithm on the embeddings to get somewhat coherent clusters. Note that Panorama did not have any of these faces in its vocabulary.

Our focus is on a new crucial system functionality for video monitoring applications: the deployed model need *not* be retrained when new classes (objects) arise. That is, users can just name the new objects from the video feed and add them to the known objects set–Panorama will *automatically* start recognizing them in the future. So, the users do not

Figure 3: Example Panorama use-case (2): unbounded vocabulary embeddings extraction for faces. The embeddings are then clustered, yielding somewhat coherent clusters.

need any ML-related expertise or worry about retraining too often. The main technical novelty of this work is a new data system architecture that solves our real-world problem in a domain-agnostic, automated, and efficient manner. To achieve our goal, we draw techniques from diverse fields–vision, ML, databases, and multimedia systems–and synthesize and adapt them for our setting. We developed a new general CNN architecture, weak supervision scheme and auto-training scheme to enable such applications. We also studied trade-off spaces between accuracy and throughput. Overall, this paper makes the following contributions:

- To the best of our knowledge, this is the first paper to propose a unified information system architecture for unbounded vocabulary queries over video using CNNs.
- We create a new multi-task CNN architecture, PanoramaNet, that supports unbounded vocabularies in a unified and unsupervised manner based on embedding extraction and content-based image retrieval (CBIR). We present an automated and domain-agnostic training pipeline combining deep and weak supervision.
- We devise a novel self short-circuiting configuration scheme for PanoramaNet to enable practical accuracy-efficiency tradeoffs. We also create a query cache to improve efficiency further at scale.
- We present an extensive empirical evaluation of the accuracy, efficiency (throughput), and scalability of Panorama on multiple real-world videos. Overall, it offers between 2x and 20x higher throughput with competitive accuracy for in-vocabulary queries, while also generalizing well to out-of-vocabulary queries.

2. SETUP AND BACKGROUND

We start by explaining our problem setup and defining some standard terminology from computer vision relevant for video monitoring applications. We then provide some technical background on multi-task deep learning and embedding extraction needed to understand our system.

2.1 Visual Querying Tasks

A video X is logically a sequence of image frames F_i, i.e., $X \equiv F_1F_2F_3\ldots$. This sequence can be unending for streaming video. The application specifies a *vocabulary* V of *objects* of interest it wants to identify in the images/video. The objects can be at any granularity, ranging from high-level generic categories (e.g., "person," "car," or "bird") to more fine-grained entity-level categories (e.g., "the person Donald Trump," "the car model Ford Mustang," or "the bird species California Quail"). Most prediction tasks in computer vision, as well as a suite of recent video querying systems, assume V is *finite*, perhaps even small. For example, NoScope [34] uses $|V| = 2$, viz., yes or no for a given object type like buses. In our setting, $|V|$ can potentially be infinite–we call this an *unbounded vocabulary*.

We are now ready to define the types of visual querying tasks of interest to us. We will then explain the implications of an unbounded vocabulary.

DEFINITION 2.1. Recognition: *Given an image frame F, identify an object $v \in V$ present in F (if at all). The recognition is called* coarse-grained *if V only has generic object categories. It is called* fine-grained *if V contains entity-level categories too. A frame can have any number of objects. This task is also called* multi-object classification *in image-based applications.*

DEFINITION 2.2. Localization*: Given an image frame F, identify the "regions" of F (e.g., bounding boxes) where all instances of objects from V are present (if at all).*

DEFINITION 2.3. Verification*: Given two image frames F_1 and F_2, identify if the same "object" arises in both images; the object is assumed to be from V.*

The above tasks are not entirely orthogonal to each other. Real-world video frames often do not have only one object. Thus, localization is needed before or during recognition. The distinction between coarse- and fine-grained recognition is also not universal but rather application-defined; a fine-grained V typically distinguishes entities of the same type, e.g., identify the person instead of just is it a person. Fine-grained recognition often leads to unbounded V in real-world video monitoring applications, the focus of this work. For example, a recognition system may be trained on a finite set of people, but it should be able to recognize other people too during usage.

Recall that our goal is to enable unbounded vocabulary querying, both verification and recognition, for video monitoring applications. We now make a key observation that we exploit later in Section 3. If V is finite, verification can be mapped to two recognition queries and comparing the labels. However, this is impossible for unbounded V. Instead, we reverse the mapping, since verification does not need to identify the label: cast recognition as multiple verification queries against known V.

2.2 Background: Multi-task Deep CNNs

Deep convolutional neural networks (CNNs) offer state-of-the-art accuracy for many computer vision tasks [40, 35] and have won many benchmark competitions [17, 53, 43, 18]. CNNs offer two critical ML engineering benefits [35]. First, they automatically learn salient features from the image during training instead of requiring extensive manual

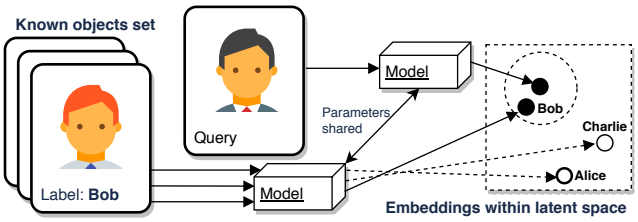

Figure 4: The embedding extracted from the query image is nearest to the known embedding for Bob in the metric space and farther from Charlie's or Alice's. This capability allows the model to distinguish between these entities.

feature engineering [24]. This is done in the form of multiple layers of feature transformations involving operations such as convolutions, pooling, and non-linearity. Second, deep learning is highly flexible in terms of the structures of the inputs and outputs. In particular, *multi-task* deep learning can predict multiple related targets simultaneously while sharing most of the internal features for each task [50]. This capability is especially attractive for our problem since most of the processing for verification and recognition queries can be shared inside a single deep CNN. Later in Section 4.1, we explain how we leverage this capability in Panorama for unified processing.

2.3 Background: Embeddings

In both vision and language understanding, an embedding is an abstract representation of an entity in a metric space. Essentially, given a set of entities S, one learns a mapping $f : S \rightarrow \mathbb{R}^d$ that maps each entity to a d-dimensional vector. Embeddings are especially popular in deep learning since they enable almost all predictive processing computations to use only linear algebra operations. Embeddings have interesting semantic properties that allow us to tell apart entities. For example, FaceNet [56] can classify faces in a known set by extracting embeddings for each, while DeepFace [47] can extract such embeddings even without specific labels. In particular, one can often use distance measures in the high-dimensional space to distinguish between entities, as illustrated by Figure 4. This remarkable capability of embeddings has recently enabled more accurate CBIR applications [63, 68, 31]. Later in Section 4.1, we explain how we leverage this capability in Panorama to tackle the unbounded vocabulary problem.

3. SYSTEM ARCHITECTURE AND API

Overview. Recall that we have three main desiderata: support for unbounded vocabulary, automated domain-agnostic pipeline, and efficiency. To achieve all these, we design Panorama in a top-down manner with three main components, as shown in Figure 5. (1) A centralized *multi-task deep CNN* we call PanoramaNet whose parameters are automatically specialized for a given application, video feed, and a *reference* model; (2) An online phase to answer verification and recognition queries efficiently by short-circuiting PanoramaNet, also called *self-cascading*, possibly combined with *nearest neighbor search*; (3) A one-time offline process of automatic training data creation, training, and configuration of short-circuiting.

Queries and API. Panorama supports verification and recognition queries (Section 2.1). It also supports variable numbers of objects per frame, since it also performs localization implicitly. Table 1 lists the functions in our API, and Listing 1 gives an end-to-end usage example. The main querying routines are `verify` and `recognize`. The `album` is a set of known object images to recognize the video stream, e.g., known people or car models. Panorama allows this set to grow *without* retraining–this supports an unbounded vocabulary, as was shown by the application in Figure 2. The `detect` routine is a fall back for bounded vocabulary recognition. The `embedding` routine extracts object embeddings from a frame; this was used for the application in Figure 3. The other routines are used for the offline phase, which we introduce next.

Table 1: Functions in Panorama API.

Methods	Action
`verify(frame_a, frame_b, target_acc)`	Verification
`recognize(frame, album, target_acc, cache)`	Unbounded-voc recognition
`detect(frame)`	Bounded-voc object detection
`embedding(frame)`	Embedding extraction
`data_gen(video)`	Data creation on the video feed
`fit(data)`	DSN training on the data
`qualify(data, delta, task)`	Configure short-circuiting

```
"""
   Parameters
   ----------
   ref_model: the reference model required for model specialization and
         cascaded query processing
   min_cluster_size: <optional> the minimum size of the cluster, as
         required by HBSCAN algorithm, only needed if the reference model
         is an embedding extractor
   data_path: the directory to the dataset for model specialization
   delta_i: <optional> the slack variable for the cascade intervals
   task: the name of the task to qualify and
   a_g: the target accuracy for query processing on verification tasks
   album_path: the directory to the known objects set for recognition
   cache: use the query cache or not for recognition

   Examples
   --------
   >>> model=Panorama(ref_model, min_cluster_size)

   # invoke data creation, model training and short-circuiting
         configuration
   >>> model.data_gen(video_feed)
   >>> model.fit(data_path)
   >>> model.qualify(data_path, delta, task=["verification", "recognition
         "])
   # run a verification query
   >>> ver_result=model.verify(file://.../frame_1324, file://.../
         frame_3325, a_g=0.9)
   # run a recognition query
   >>> rec_result=model.recognize(file://.../frame_1324, album_path, a_g
         =0.9, cache=False)

"""
```

Listing 1: Panorama API and example for image recognition and verification.

Offline Phase. The user provides a video/image feed, a relevant reference model, and optional configuration parameters for Panorama. The reference model solves the *bounded* vocabulary recognition task, e.g., identify a known set of faces. Panorama's goal is to mimic this model's accuracy on the known object set while generalizing beyond that set with higher efficiency. Using the reference model, Panorama automatically generates training data on the video feed (Section 4.3) and trains PanoramaNet (Section 4.2). If the reference model yields embeddings instead of labels, then a configuration parameter can ask Panorama to generate labels instead. The training of PanoramaNet implicitly configures its short-circuiting using a novel mechanism (Section 4.4).

Online Phase. The user specifies the `verify` and/or `recognize` query as explained above for monitoring the video.

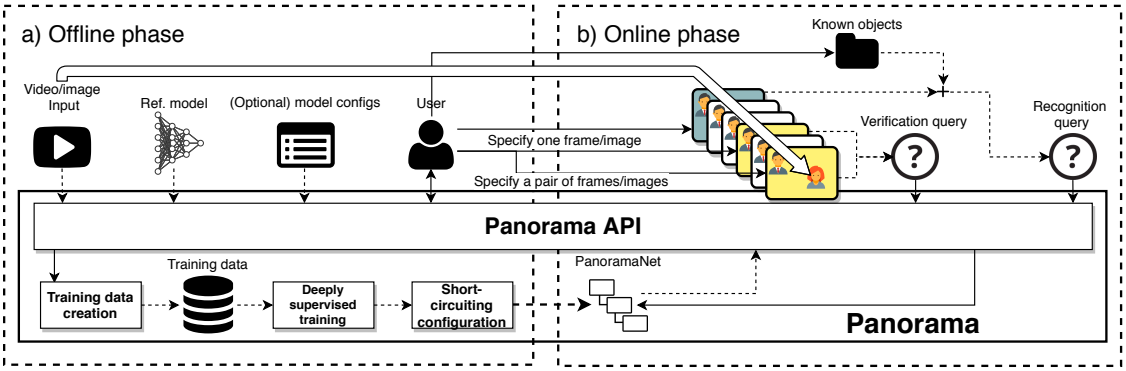

Figure 5: Overall system architecture of Panorama. Solid arrows represent invocations/interactions, while dashed arrows represent the flow of data/results. PanoramaNet is built once offline and then deployed for online video monitoring.

The user interface is application-specific and orthogonal to this work. The interface shown in Figure 2 is only an example. They also specify an *accuracy goal* (relative to the reference model) to customize Panorama's accuracy-efficiency tradeoff. Panorama extracts embeddings from the given frames and compares them for verification or recognition as appropriate. For recognition, a nearest neighbor search is performed during inference.

4. COMPONENTS AND TECHNIQUES

Most existing video querying models perform localization and recognition separately, and they do not support unbounded vocabulary. Adapting them to recognize new entities could require tedious manual retraining. In contrast, Panorama builds a single multi-task deep CNN that is automatically customized to each application. It "short-circuits" itself at query time to improve efficiency. Figure 6 illustrates Panorama's working in more detail. Next, we dive into each component: model architecture, training process, short-circuiting configuration, and inference process.

4.1 Deeply Cascaded Multi-task Model

Goals. At the heart of Panorama is a centralized multi-task deep CNN we call PanoramaNet. We have three goals for the design of this model. (1) Supporting both verification and recognition, as well as localization to identify multiple objects in a frame. (2) Being Domain-agnostic and not too tied to one application, e.g., faces or car models. (3) Being able to gracefully tradeoff inference cost for accuracy.

Basic Idea and Observations. To meet all three goals, we design PanoramaNet as a *multi-task* deep CNN with a *cascaded* modular structure. It has multiple trainable "blocks" of CNN layers, each with its own output block. The lowest layers of the CNNs act as a shared feature extractor for all blocks. All output layers have the same semantics, but they offer different accuracy-runtime tradeoffs. By short-circuiting at an earlier block, the inference cost goes down. Each output block has multiple intermediate targets for supervision during training, including bounding box regression, embedding extraction, and in-vocabulary recognition loss. This multi-task setup is what allows Panorama to simultaneously recognize in-vocabulary objects and generalize to out-of-vocabulary objects after deployment. During the training process, short-circuiting is configured based on a user-given *accuracy goal* and the model's accuracy on a validation set.

PanoramaNet Neural Architecture. Our model performs localization and embedding extraction jointly in one pass. Its base architecture is adapted from Yolov2-tiny [52] with two major modifications for our problem. First, while Yolov2 is a one-pass model for localization and recognition, it does not support unbounded vocabulary. Thus, we augment it by "wiring in" an embedding extraction module. Second, inside each grid cell that segregates the feature maps of the CNN layers in Yolo, the bounding box regressors (for localization) and recognizers work independently. So, even if the bounding box is poor, the recognizer may still yield correct labels. However, in our setting, this property is antithetical for embedding extraction, since the box-segmented image must align well with the object for embedding extractors to work properly. To tackle this issue, we confine each recognizer to its corresponding bounding box regressor via 3D convolutional layers.

Figure 7 shows the high-level architecture of PanoramaNet. Due to space constrains, Stem_1 architecture is presented in Appendix. Figure 8 expands Output Block_i; layer are annotated with their size, name, and the number of filters, e.g., 3x3 Conv2D(1024) means a 3x3 2D convolutional layer with 1024 filters. PanoramaNet stacks many such Stem and Output Blocks. We collectively denote each Stem_i along with its Output Block_i as Block_i. We use an embedding dimension of 128. We use Euclidean distance (L2 norm) to compare embeddings.

Output Blocks. As Figure 8 shows, Output Blocks have modules that are used only during the offline phase. In particular, the embedding extractor is also trained during the offline phase using in-vocabulary labels. During the online phase, the recognizer module is applicable only for in-vocabulary recognition, but the embedding extractor applies to both in- and out-of-vocabulary recognition. Due to our multi-task setup, all outputs have both bounding boxes and embeddings (or labels). Outputs are then thresholded based on the "objectness" of the bounding boxes (explained more in Section 4.2) and then thresholded with non-maximal suppression. Throughout the paper, we set the former to be 0.1 for verification and 0.03 for recognition, and the latter

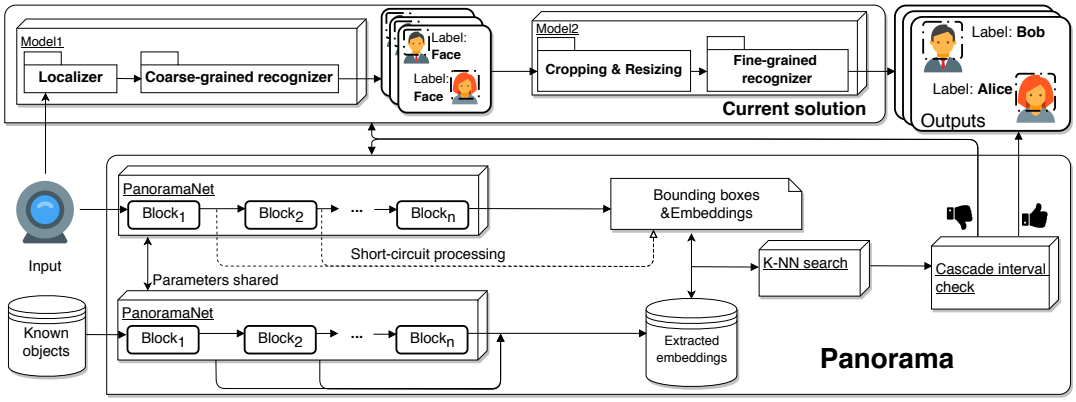

Figure 6: Detailed workflow of Panorama's internals for processing a recognition query. The deeply cascaded PanoramaNet can be short-circuited and is combined with nearest neighbor search for enabling unbounded vocabulary recognition in one pass. Also shown is a typical prior art solution; it takes a two-pass approach, with separate modules for coarse-grained and fine-grained recognition. The prior art solution also does not support unbounded vocabulary.

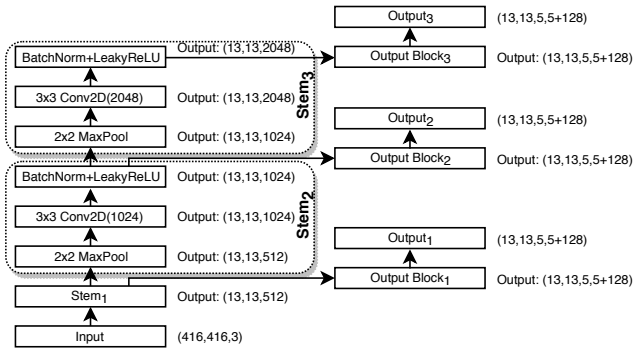

Figure 7: PanoramaNet deep cascade architecture with $n = 3$ blocks. An output has dimensions (grid, grid, number of bounding boxes, bounding box parameters+embedding dimension). All layers shown, including max-pooling, have a stride of 1 and are same-padded.

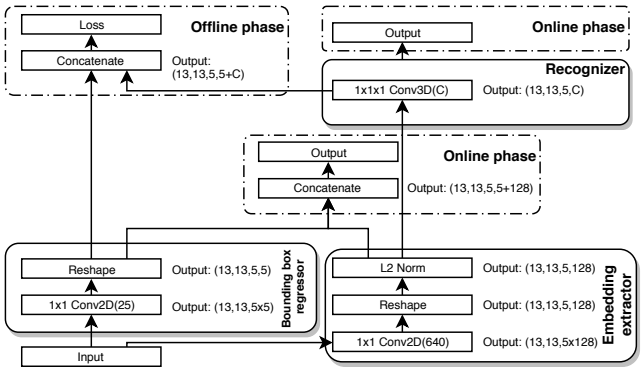

Figure 8: Detailed architecture of Output Blocks from Figure 7. All layers have a stride of 1 and are same-padded.

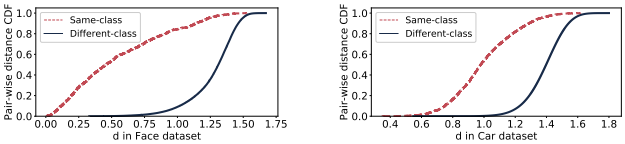

Figure 9: CDFs of pairwise Euclidean distances between the embeddings yielded by Block_2 of PanoramaNet.

to be 0.5.

4.1.1 *Answering Verification Queries*

We now explain how PanoramaNet answers verification queries. The model outputs embeddings from each of the two input frames/images. We then simply *threshold* on the L2 distances of the embeddings as follows. Given two frames f_i and f_j and their corresponding embedding sets $\{e_i\}$ and $\{e_j\}$, the verification answer is *yes* if the following holds (otherwise, it is *no*):

$$\min_{i,j} ||e_i - e_j|| \leq \gamma,$$

In the above, γ is a Panorama configuration threshold to distinguish embeddings of different entities in the metric space. This approach works because as explained in Section 2.3, well-trained embeddings offer us this geometric capability to roughly tell apart different entity classes. But how to set γ? We set γ based on a held-out validation set during the offline training process. This requires a balancing act between precision and recall. To achieve this balance, consider the CDFs of the pairwise distances for same-class (e.g., same person) embeddings and different-class embeddings in Figure 9. On the *Faces* dataset (explained more in Section 5), a threshold of $\gamma = 0.8$ reasonably separates same-class pairs from different-class pairs with high precision. Similarly, on the *Cars* dataset, $\gamma = 1.1$ is suitable. We prefer such high-precision thresholds, since overall recall can be enhanced through other means, e.g., have multiple different images for known objects.

4.1.2 *Answering Recognition Queries*

As mentioned earlier, we map a recognition query to multiple verification queries. Given a query image's embeddings (e.g., from a video frame), we perform a nearest neighbor search against the embeddings in the album. This is done as bulk matrix arithmetic on the GPU, which turned out to be much faster than indexing. Thresholding can be used

on top to ensure the retrieved neighbors are similar enough. Since recognition involves multiple queries, it is more prone to errors and harder to optimize. Thus, Panorama offers a configuration option of using the recognizer component in PanoramaNet for output labels directly for in-vocabulary recognition; note this is not possible for out-of-vocabulary recognition and only nearest neighbor search can be used.

4.2 Training with Deep Supervision

Since PanoramaNet has multiple output layers, we need to consider all of their loss functions during backpropagation. To this end, we use the "deep supervision" approach introduced in [41]. It was originally devised to tackle the vanishing gradients issue for accuracy and for better discriminative power of each layer. We repurpose it to enable our accuracy-throughput tradeoff; to the best of our knowledge, this is the first time deep supervision is exploited this way. The overall loss is as follows:

$$L = \sum \lambda_k l_k, \tag{1}$$

In the above, λ_k is the weight for output layer k and l_k is that layer's loss. Each l_k is backpropagated only through its parent layers. λ_k controls the trade-off between more opportunities of early-exit vs better over-all performance. We set each λ_k inversely proportional to the number of FLOPS to compute that layer's output. In particular, we set $(\lambda_1, \lambda_2, \lambda_3) = (8, 2, 1)$. We conduct experiments in Section 5 to study the effect of these weights. Note that our deeply cascaded architecture is generic; l_k can be any form of loss determined by the multi-task target goals. In particular, l_k in PanoramaNet is the same loss as in Yolov2; due to space constraints, we present the whole loss function in Appendix.

Given Block$_k$, B is the number of anchor boxes, S is the number of grids, (x_i, y_i, w_i, h_i) are the location of the centroid, and the width and height of anchor boxes. C_i is the "objectness" of the output, referred to earlier in Section 4.1. λ_{coord} and λ_{noobj} are weights to balance the parts of the loss; we use the default weights from [51]. Finally, $p_i(c)$ is the classification "confidence" for class c. We adapt the code from [5] to implement our loss function.

4.3 Automated Training Data Creation

PanoramaNet is domain-agnostic and meets our systems-oriented goals. But it still needs to be trained on a specific application's data. To this end, we create an automated training process to customize PanoramaNet to a given dataset in the offline phase. We first run the user-given reference model on a portion of the video (or subset of images) to create "weakly supervised" training data [70]. The reference model must provide both bounding boxes and labels for the corresponding bounded vocabulary task. We also support models that produce embeddings instead of labels; in this case, Panorama clusters the embeddings and assigns a label per cluster. We use HDBSCAN [12] for clustering; the user can set its hyper-parameters during configuration or use defaults. We also denoise the clustered data by removing outliers. Overall, the reference model "teaches" Panorama, which means the reference model caps its in-vocabulary accuracy. If one desires higher accuracy, or if a reference model is not available for an application, the user has to give PanoramaNet a whole labeled dataset; we used this approach for the *Cars* dataset in Section 5.

4.4 Configuration of Short-Circuiting

Goals and Basic Idea. A critical design decision in PanoramaNet is its multi-block architecture, which enables a graceful accuracy-throughput tradeoff by short-circuiting. But when to short-circuit? Recall that the user sets an *accuracy goal*. We need to satisfy this goal reliably at query time. Our basic idea is to compute a "score" for a given query at each block and compare it against a pre-computed "cascade interval" for that block. If the query's score at a block falls in its cascade interval, it means the model is not confident about this intermediate output and so, subsequent blocks need to be invoked. Otherwise, we short-circuit at the current output and return immediately. We first explain how we use cascade intervals and then explain how we set them, including how our approach ensures correctness.

Using Cascade Intervals. We pre-compute a cascade interval $[L_i, H_i]$ for Block$_i$ in the offline phase. In the online phase, we are given a verification query with two frames/images f and g. Let d_i denote the distance between the pair of embeddings output by Block$_i$ for these frames; this is our score for short-circuiting. We start processing both frames from the first block until we hit a Block$_i$ such that $d_i \in [L_i, H_i]$. If no block satisfies this, we invoke the reference model, which acts as the "pseudo ground truth" in our weakly supervised setup.

Let a_g denote the user's accuracy goal. Let the actual accuracy of Block$_i$ be a_i on a given labeled set of examples $\mathbb{D}_v = \{((f, g), y)\}$, wherein y is the ground truth (yes or no); denote $|\mathbb{D}_v|$ by N. So, Block$_i$ has correctly answered Na_i queries. If $a_g > a_i$, it means Block$_i$ has a deficit of $N(a_g - a_i)$ queries to meet the accuracy goal. Thus, for short circuiting to succeed at Block$_i$, the percentage of queries that should have been answered correctly within the set of wrongly answered queries is $\frac{N(a_g-a_i)}{N(1-a_i)} = \frac{a_g-a_i}{1-a_i} \equiv q_i$ (say).

Setting Cascade Intervals. In the offline phase, we plot the CDF of d_i for queries that did *not* get correctly answered at Block$_i$ using the labeled validation set $\mathbb{D}_v$. We set $[L_i, H_i]$ to match the above percentage q_i of these queries. A natural choice is to select an interval around the median:

$$L_i = P(0.5 - \frac{a_g - a_i}{2(1 - a_i)} - \delta_i, \mathbb{S}_e) \tag{2}$$

$$H_i = P(0.5 + \frac{a_g - a_i}{2(1 - a_i)} + \delta_i, \mathbb{S}_e) \tag{3}$$

In the above, $P(x, \mathbb{S})$ denotes the x percentile of the set $\mathbb{S}$. $\mathbb{S}_e$ is the set of d_i for all examples in $\mathbb{D}_v$ such that short-circuiting at Block$_i$ gives the wrong prediction (i.e., the output is the opposite of y). Under the assumption that the validation set and deployment data come from the same or similar distribution, the above values guarantee that the accuracy goal will be met, while short-circuiting as much as possible. To account for statistical differences between the deployment data and validation set, we also include a small slack variable δ_i.

Correctness Analysis. We now explain why our above approach guarantees that the accuracy goal a_g will be met. Let the accuracy of Block$_i$ be $a_i < a_g$. Queries that fall into the interval $[L_i, H_i]$ at Block$_i$ all get sent to the next block. Note that since we do not have ground truth in the online phase, we do not know if Block$_i$ answered any queries correctly; we can only rely on c_i for short-circuiting. But

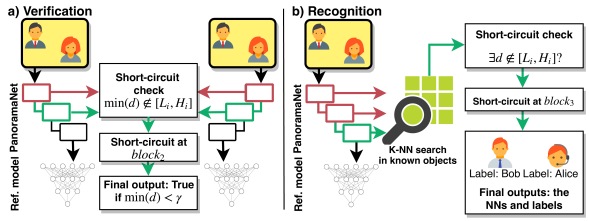

Figure 10: Examples of Panorama's inference execution. a): The verification query is short-circuited at $block_2$. The left and right models including PanoramaNet and the reference model share parameters, respectively. b). The recognition query is short-circuited at $block_3$. Embeddings from the known objects were pre-extracted and stored.

note that exactly q_i fraction of all wrongly answered queries (and unknown numbers of correctly answered queries) are sent by Block_i to a later block to be eventually answered correctly, perhaps ultimately by the reference model itself. Thus, the overall accuracy goes up from a_i to at least a_g by performing more inference computations (invoking more blocks) for queries that did not get short-circuited.

4.5 Query Cache

Intuition. Video with high frame rates lead to lots of queries, e.g., 40Hz means 40 queries for 1s. However, videos also have high *temporal redundancy*: most successive frames are similar. Thus, downsampling can raise efficiency without hurting accuracy much (e.g., 1 frame from 1s). But we can go further to exploit a key property of our target applications: objects typically do not appear and disappear too fast. Some objects may even last minutes, e.g., faces in news videos. This gives us to another systems insight: *cache recent query results to reduce computations for the same object.* Such a query cache skips the costly nearest neighbor search for successive recognition queries.

Mechanism. We create an approximate query cache with the embeddings since they exist in a metric space with Euclidean distance as an indicator of similarity. Denote $d(x, y)$ as the distance between embeddings x and y. Let e_a be the embedding from a recent frame. Let e' be the embedding of its nearest neighbor result from `known`. For a new frame with embedding e_b, we have one observation based on the triangle inequality; Suppose $d(e_a, e') \leq \gamma$, where γ is the threshold for same-class embeddings (Section 4.1.1). If $d(e_b, e_a) \leq d(e_a, e')$, return the label of e' as the result and skip the search. This approach is an approximation because a different embedding in the album may be nearer to e_b than e_a (although with low probability). Thus, our cache creates a runtime-accuracy tradeoff.

Corresponding to the observation, we cache the most recent several frames and evict in FIFO manner, the number of which is the cache size. Given a frame, we check the cache for hits and return the labels. We then take the misses and do a normal k-nn search to get labels for them. Finally we update the cache with all labels acquired in the above steps for this frame. Overall, this query cache can reduce runtimes significantly when the known objects set is massive.

4.6 Online Phase Inference Process

Figure 10 depicts how queries are processed in the online phase. For *verification*, both frames are passed to PanoramaNet for embedding extraction one block at a time. Pairwise distances between the embeddings are checked for short-circuiting. If short-circuiting succeeds (Section 4.4), we threshold the distance against γ for the final answer (yes or no). For *recognition* queries, Panorama extracts embeddings from given frames and compares against the embeddings (for the corresponding block) in the known object set via a nearest neighbor search. This search might potentially be skipped by the query cache (Section 4.5). Once again, if short-circuiting succeeds at some block, we stop and return the nearest result's label. As mentioned in section 4.4, the reference model is the fallback option in case none of the blocks of PanoramaNet can answer the query with high confidence.

5. EXPERIMENTS

We now evaluate Panorama with several real-world workloads and datasets for both verification and recognition queries. In summary, our results show the following:

(1) For in-vocabulary verification, Panorama offers between 2x and 8x higher throughput (lower latency) than a strong baseline, while offering competitive accuracy. For in-vocabulary recognition; the speedups are up to 20x.

(2) Panorama generalizes well for out-of-vocabulary queries, offering much higher accuracy than random guessing baselines, while still offering high throughput.

(3) Panorama configuration parameters enable a graceful tradeoff between accuracy and throughput.

(4) As the known objects set size scales up for recognition, Panorama's query cache helps raise throughput up to 6x.

We first describe the datasets and workloads used. We then present the end-to-end performance results followed by a drill-down study of the contributions of Panorama's techniques. Finally, we present the scalability test.

5.1 Experimental Setup

Datasets. Table 2 lists our datasets. *Faces*[14] and *Birds*[36] are videos recorded from online surveillance cameras at 30Hz frame rate. *Faces* is for recognizing people; *Birds*, for recognizing bird species. All videos are decoded and unpacked into frames. We sample 1 frame per second. Our baseline models also operates on the downsampled frames instead of the original video, which makes them already strong baselines for the throughput-accuracy tradeoff we study. *Cars* is an image dataset for car model recognition [64].

Table 2: Datasets and reference models for experiments.

Dataset	Source	|Voc.|	#Frames	Ref. model
Faces	CBSN[14]	60	5.4m	MTCNN[67]+FaceNet[56]
Birds	Bird Cam[36]	6	5.4m	Yolo[52]+Inceptionv3[16]
Cars	CompCars[64]	431	45k	Yolo[52]+GoogLeNetCars[65]

Reference Models. Each reference model has two submodels, as Table 2 shows. The reference model for *Faces* produces embeddings; thus, we create pseudo-labels after un-supervised clustering. Overall, the reference models operate on a bounded vocabulary. For *Faces* and *Birds*, Panorama

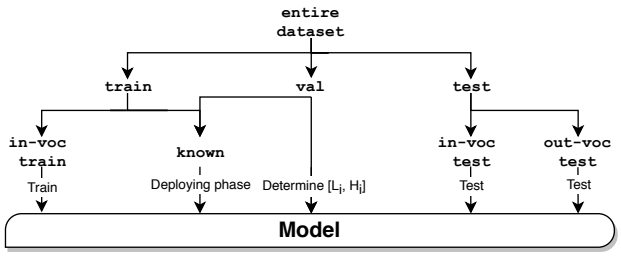

Figure 11: Schematic diagram about dataset split.

is weakly supervised by the respective reference model (Section 4.3), but for *Cars*, we used the CompCars [64] labeled dataset to show that Panorama can work on strong-supervised image data as well, not just week-supervised videos.

Data Split Methodology. Figure 11 shows how we split the datasets. We first split all examples into `train`, `val` and `test`. At the same time, we split the vocabulary into `in-voc` and `out-voc`. Then, `test` is further split into `in-voc test`, with `test` examples that have `in-voc` labels, and `out-voc test`, the rest of `test`. Only the `in-voc train` of `train` is used in the CNN training. `val` serves for the validation during training and short-circuiting configuration. Then at deploying time, we poll 5 best frames per class, based on Panorama's confidence score of object detection, from `train` and `val` to form `known` for subsequent recognition queries. Then `known` becomes the new vocabulary. For videos, we chunk the videos, instead of random order, into 60:20:20 ratio for the train-val-test split. The vocabulary is also chunked into 80:20 for in-out-voc split, sorted in descending order by the cardinality of each class. But for *Cars*, we reuse the pre-existing 70:30 train-test split in its original labeled dataset; however, its vocabulary is also split 80:20. The `val` split is 10% of `train`. Overall, PanoramaNet is trained only with `in-voc train`, which allows us to simulate the unbounded vocabulary scenario. At deployment time, we use `known` as the album for recognition queries.

Training Panorama. We train PanoramaNet with Adam optimizer. Adam is configured with an initial learning rate of 0.5×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. We use a batch size of 64 for *Faces* and *Birds* and 8 for *Cars*. The training is terminated if for 10 consecutive epochs the validation loss does not improve. Training for `face` terminates after 48 epochs taking 2 day 15 hours. Training for `bird` terminates after 109 epochs taking 1 day 3 hours. Training for `car` terminates after 208 epochs taking 8 days 6 hours.

Accuracy Methodology. As explained in Section 4.3, we use the denoised outputs of the reference model as labeled data for Panorama. Thus, all `in-voc test` accuracy is reported relative to the reference model. This methodology is fair because our focus is not on improving absolute accuracy but rather systems issues of functionality and efficiency.

Evaluation Metrics. We have three main metrics: *throughput*, *verification accuracy*, and *recognition accuracy*. Throughput is the number of queries answered per second; it is based on the wall-clock time taken for all queries put together. Since we have no batch processing or task parallelism, the higher the throughput, the lower the query latency. We omit all frame preprocessing time (e.g., decoding or resizing) for all compared approaches because they were minor.

Verification accuracy is defined based on standard practice [29] as the ratio of the number of queries that were correctly answered to the total number of queries. Given a verification query with a pair of frames/images (f, g), denote the sets of classes appearing in f and g by $\mathbb{O}_f$ and $\mathbb{O}_g$, respectively. The query with (f, g) returns yes if and only if $|\mathbb{O}_f \cap \mathbb{O}_g| \geq 1$; otherwise, the query returns no.

For recognition accuracy, we only evaluate it on frames on which the reference model gave output labels. On a frame containing l classes $\{Y_0, Y_1, ..., Y_i, ..., Y_l\}$, we ask the model to give at most m labels $\{z_0, z_1, ..., z_j ..., z_m\}$. This gives us a standard metric called "top-m" accuracy; we set $m = 5$ for our experiments. Recognition accuracy at the frame level is now defined as follows.

$$k_r = 1 - \frac{1}{l} \sum_j \min_i \mathbb{1}_{z_i = Y_j}, \tag{4}$$

The overall recognition accuracy is then defined as follows, wherein $\mathbb{Q}$ is the set of all recognition queries:

$$K_r = \frac{1}{|\mathbb{Q}|} \sum k_r, \tag{5}$$

Software and Hardware. Panorama is implemented entirely in Python. All CNNs and the nearest neighbor search are implemented using TensorFlow 1.4 and Keras 2.1.4 and use GPU acceleration with CUDA 7.0. We used OpenCV 2.0 with FFmpeg backend and PIL 1.1.7 for image preprocessing. All experiments were run on a machine with an NVIDIA GeForce GTX 1080Ti GPU, 8 Intel Xeon E5-2630 v4 cores, and 32 GB RAM.

5.2 End-to-end Accuracy and Throughput

We start with the end-to-end performance results, both accuracy, and throughput. Since Panorama is a first-of-its-kind system, prior video querying systems are not quantitatively comparable (more details in Section 6). Thus, we compare Panorama against the reference model, which works only for in-vocabulary queries. On `in-voc test`, we report Panorama's test results relative to the reference model. We then report Panorama's absolute test results on `out-voc test` to show how well it generalizes beyond its supervision vocabulary. We disable the query cache in Panorama for all the experiments in this subsection to let us focus on its main accuracy-throughput tradeoffs. Last we include a strongly supervised video clips dataset *Youtube* on faces to measure Panorama's capability of generalization. The dataset has a vocabulary size of 1595 and over 620k frames in total. We use this dataset to see if Panorama can even generalize beyond its supervision to distinct videos. We test PanoramaNet trained with *Faces* on this dataset. We do not split *Youtube* as it is only used for tests. We poll `known` and simply treat the rest as `out-voc test`.

5.2.1 Verification Queries

Query Set. We randomly sample pairs of frames from `in-voc test` (resp. `out-voc test`) for the in-vocabulary (resp. out-of-vocabulary) verification tests. For all tests, we produce 10^4 pairs each with a 50:50 split for yes and no. Since `out-voc test` in *Birds* is relatively small, we produce only 500 pairs on this but still with a 50:50 yes-no split. We compare two settings for Panorama's accuracy goal configuration parameter: $a_g = 0.9$ and $a_g = 0.99$. All slack parameters (δ_i) are set to 0. Recall that a reference model answers `in-voc` verification via recognition of the objects

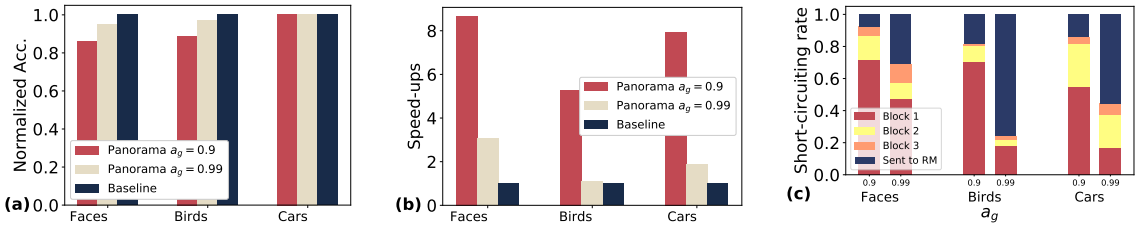

Figure 12: End-to-end `in-voc` verification results. (a) Verification accuracy. (b) Relative throughput. Baseline represents the corresponding reference model, the absolute values for three baselines are 7.3, 18.0 and 2.7, respectively; Panorama's results are normalized with respect to them. (c) Fractions of queries short-circuited at each block. "Sent to RM" means those queries were handled by the reference model.

Table 3: `out-voc` verification results. *P: Panorama. †RG: random guessing.

	Faces	*Youtube*	*Birds*	*Cars*
Thrpt. (frames/s)	130	120	96	101
P* accuracy	**81.6%**	**79.6%**	50.0%	**69.7%**
RG† accuracy	50.0%	50.0%	50.0%	50.0%

Table 4: `out-voc` recognition results. *P: Panorama. †RG: random guessing. ‡: Top-1 accuracy.

	Faces	*Youtube*	*Birds*	*Cars*
Thrpt. (frames/s)	107	105	97	63
P* Accuracy	**74.5%**	**46.4%**	**73.9%**‡	**49.6%**
RG† accuracy	38.5%	0.3%	50%‡	5.7%

in both images and comparing their labels, but it does not support `out-voc` verification.

In-Vocabulary Results. Figure 12 shows the accuracy and throughput results. We see that Panorama achieves substantially higher throughput while yielding competitive accuracy on *Faces* and *Cars*. For instance, on *Faces*, Panorama with $a_g = 0.99$ has 96% accuracy but is 2x faster; with a 14% drop in accuracy, the other setting is 8x faster. Interestingly, on *Cars*, we found that Panorama's accuracy was slightly higher than the reference model (skipped in the figure, which is capped at 1); recall that we had trained both approaches from scratch on the original labeled dataset in this case. Panorama is also up to 8x faster on *Cars*. On *Birds*, $a_g = 0.9$ is 5x faster while giving 92% of accuracy.

Figure 12(c) explains the above results. Panorama's short-circuit processing worked well, with many queries stopping at earlier blocks. In fact, with $a_g = 0.99$, on *Faces*, over half of queries were short-circuited at block 1 and 2. But on *Birds*, more queries were sent to the reference model, yielding a lower average speedup. *Cars* is in between these two extremes. These results validate two of our key design decisions: make PanoramaNet a multi-block architecture that can short-circuit itself and automatically customize it for a dataset to pick an appropriate point in the accuracy-throughput tradeoff.

Out-of-Vocabulary Results. In reality the reference models do not work on out-voc queries. So, we compare Panorama against a random guessing baseline, which is 50% for this binary task. We use $a_g = 0.9$ and no δ_i for all tests. Table 3 presents the absolute results. Panorama successfully generalizes beyond its supervision vocabulary to support out-of-vocabulary verification. On *Faces*, the lift is a substantial 33%. *Birds* turns out to be more challenging, while *Cars* falls in between. Panorama's throughput is also well above real-time in all cases. It generalizes well to *Youtube*, which contains distinct videos (e.g. different resolutions, illumination, angles and distances to camera) from *Faces*.

5.2.2 Recognition Queries

Query Set. We compare two settings for Panorama: Cas 1 and Cas 2; Cas 2 represents a stricter accuracy goal than Cas 1, but we vary the configuration parameters across each dataset because they exhibited different properties on the verification tests. For `Faces`, Cas 1 uses $(a_g, \delta_i) = (0.95, 0)$; Cas 2 uses $(a_g, \delta_i) = (0.99, 0.1)$. For `Birds`, Cas 1 uses $(a_g, \delta_i) = (0.9, 0)$; Cas 2 uses $(a_g, \delta_i) = (0.99, 0)$. Finally, for `Cars`, Cas 1 uses $(a_g, \delta_1, \delta_2, \delta_3) = (0.9, \infty, 0, 0)$; Cas 2 uses $(a_g, \delta_1, \delta_2, \delta_3) = (0.99, \infty, 0, 0)$. Note that setting $\delta_j = \infty$ means short-circuiting is not allowed at Block$_j$.

In-Vocabulary Results. Figure 13 shows the results. On *Faces*, Cas 1 is 17x faster, while offering almost 80% relative accuracy, Cas 2 is 2x faster while yielding 92% accuracy. On *Cars*, both settings match (or slightly surpass) the reference model's accuracy, while being up to 20x faster. Compared to *Faces*, *Birds* offers a slightly more modest speedups but with higher accuracy. Figure 13(c) explains these results in terms of the short-circuiting results. We see similar behaviors as in the `in-voc` verification tests.

Out-of-Vocabulary Results. Once again, since the reference models are not applicable here, we compare Panorama to a random guessing baseline. We use $a_g = 0.9$ and no slacks for all tests. Recognition is effectively multi-class, not binary. So, the accuracy of random guessing depends on the sizes of the vocabularies in `known`; these sizes are 14, 2, 87, 1595 for *Faces*, *Birds*, *Cars*, *Youtube*, respectively. We report top-1 accuracy for *Birds* (since the vocabulary size is only 2) and top-5 accuracy for the rest. Table 4 presents the absolute results. Once again, we see that Panorama successfully generalizes beyond its supervision vocabulary to support out-of-vocabulary recognition queries too. On *Cars*, the lift is a substantial 44%. It generalizes well to *Youtube*, offering 46% lift on accuracy with high throughput.

5.3 Drill-down Analysis

Factor Analysis. We now drill into Panorama's behavior to show the effects of its various components on the throughput-accuracy tradeoff. We expand `in-voc` recognition tests on

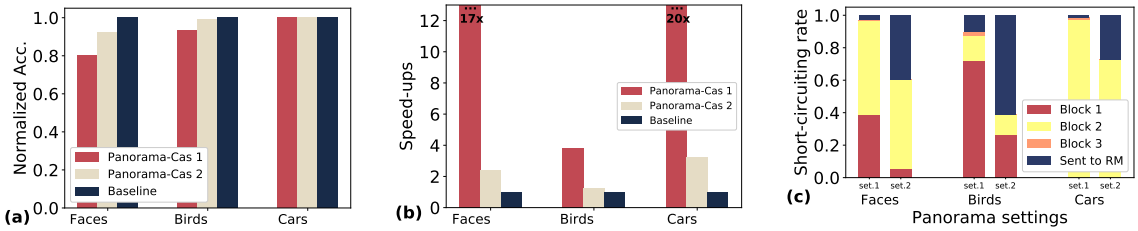

Figure 13: End-to-end `in-voc` recognition results. (a) Recognition accuracy. (b) Relative throughput. Baseline represents the corresponding reference model, the absolute values are identical to Figure 12; Panorama's results are normalized with respect to them. (c) Fractions of queries short-circuited at each block. "Sent to RM" means being handled by the reference model.

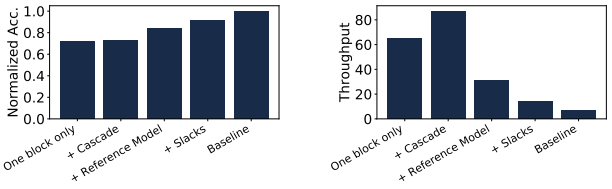

Figure 14: Factor analysis. Accuracy is normalized against the reference model.

Table 5: Impact of λ_k on block-wise verification accuracy.

λ_k	Acc. Block1	Acc. Block2	Acc. Block3
1:1:1	79.9%	87.8%	89.1%
8:2:1	83.2%	84.3	87.2%
100:10:1	61.4%	58.4%	65.5%

Faces for this purpose. We use Cas 2 described above for the cascade related configs. Figure 14 presents the results of each component's effect. We start by disabling short-circuiting and taking only the output of last block of PanoramaNet. This provides over 60 FPS but limits the accuracy to 72%. If we enable the rest blocks and cascaded processing, throughput boosts to over 80 FPS. This demonstrates the effeteness of our cascade. Next if we concatenate the reference model into the cascade, the accuracy further improves to 84% with the speedups drop to 30 FPS. The last element needed are the slacks to yield 92% accuracy, with a 2x speed-up compared to baseline still.

Impact of λ_k. We now investigate the impact brought by different settings of λ_k. We vary these weights and train three different models and report the raw verification performance of each block of the models on our *Faces* `val` split. Table 5 summarizes the results. Compared to no weights (1:1:1), setting a higher weights on the first block (8:2:1) does improve the individual performance of block1, however, the subsequent blocks loses some accuracy. These weights provide a trade-off between early and later block discrimination power. On the other hand, too large weights (100:10:1) interferes with the training process and fails to converge.

5.4 Query Cache and Scalability Test

We now stress test Panorama's throughput by raising the size of the known objects for recognition queries. Some real-world video monitoring applications could indeed have to deal with millions of objects, e.g., identifying faces in mall security surveillance, and the database for faces can be excessively large. We pick the same setting on the in-voc recognition test of *Faces* and use cascade setting 1.

Table 6: Impact of query cache on recognition accuracy.

Cache size	Relative accuracy	Cache hit rate
0(No cache)	80%	0%
1	80%	43%
10	80%	80%
100	80%	89%

Table 7: Results of the scalability test. W/O cache means Panorama without cache, and Cache X means Panorama with cache size X. All values in the right four columns are throughputs reported in frames/sec.

$\lvert$`known`$\rvert$	Baseline	W/O cache	Cache 1	Cache 100
10^5	7.2	24.0	33.0	**45.9**
5×10^5	6.1	9.0	11.4	**20.7**
10^6	4.6	5.1	6.7	**14.7**

Impact of query cache on accuracy. We now enable the cache and investigate the impact brought by the query cache with varying cache size. Table 6 summarizes the results. As the size of the cache goes up, the cache hit rate rises, while the accuracy remains relatively constant, meaning this cache does not influence accuracy much. Although the video is after downsampling, the cache hit rate can still be as high as 89%. This demonstrates the temporal redundancy characteristics of video.

Scalability test. To simulate the case where the `known` set is at scale, we enlarge the existing `known` set with duplicates. For this experiment, we run Panorama in three modes: without the query cache, with size-1 cache and size-100 cache. We compare the results to the *Faces* reference model, which also yields embeddings and uses k-nn for recognition. Table 7 shows the results for throughput when `known` is at scale. In this scenario the k-nn search becomes a bigger bottleneck compared to CNN inference. Without the cache, the throughput of Panorama will eventually join baseline as the known object set expands. However, Panorama with size-100 cache still offers 3x∼6x speedups depending on $\lvert$`known`$\rvert$. Cache-100 also outperforms Cache-1, as the former has much higher cache hit rate and skips more searches. This validates the benefits of the query cache for large-scale recognition queries.

5.5 Limitations and Discussion

Panorama generalizes beyond a given finite vocabulary to

unseen objects of the same type in a given domain. This is a form of subclassing, i.e., Panorama does not generalize to new types of objects or new domains. We now offer some insights on when Panorama may or may not be applicable. It applies to fine-grained visual tasks, and the granularity is determined by the supervision provided. The viability of a task depends on the availability of large-scale datasets and/or high-quality reference models and the degree of difficulty of the task itself. Faces are most viable because of their relatively well-understood properties: mostly 2-D and simple geometric layout. There are also many large datasets for faces. For cars, the datasets are decent; so, the accuracy is good. As for other domains, as long as there exists large-enough fine-grained datasets and/or good reference models, we believe Panorama is applicable.

6. RELATED WORK

Vision and Label-efficient ML. We already explained how Panorama relates to prior works in vision, including task definitions (Section 2.1), CNN-based vision (Section 2.2), how PanoramaNet is based on lessons of recent CNNs (Section 4.1), deep supervision (Section 4.2), and which CNNs act as reference models (Section 5). Thus, we now only discuss a key aspect of Panorama's goal that is related to several lines of work in ML. Deep CNNs typically need large labeled datasets, but many applications may not have so much labeled data. To meet this challenge, the ML community has long worked on label-efficient learning schemes, including *zero-shot* [39, 38, 4, 22, 46], *one-shot* [37, 19, 20, 6, 21, 39], and *open-set* [54, 55, 23, 8] learning. Zero-shot learning asks models to recognize new unseen classes by transferring semantic knowledge learned from training classes, often via auxiliary metadata for retraining. One-shot learning relaxes this assumption by asking for one or a few labeled examples per new class. Open-set learning also aims to remove the closed-world vocabulary assumption, but it does so by retraining models to recognize both old and new classes. Such alternative learning schemes are sometimes collectively called *life-long* learning [15, 48, 60].

All these previous efforts in ML inspire our formulation of the *unbounded vocabulary* problem, but our goal is *not* proposing new learning schemes, vision tasks, or more accurate CNNs. Our focus in Panorama has a crucial system functionality difference aimed at benefiting users of video monitoring applications.

Cascaded Classification. Cascaded models have long been used in multimedia systems to improve accuracy and efficiency. Introduced in the Viola-Jones object detection framework [61], recent works have extended this idea to deep CNNs [10, 25, 42, 58, 62, 11, 28]. These works inspired our design decision of making PanoramaNet cascaded, but our approach extends this idea along two lines: we fuse it with multi-task learning for unified processing instead of disparate bespoke models and we use deeply supervised training (originally designed to improve accuracy [41]) to make this fusion possible. Our short-circuiting configuration also supports a more tunable accuracy-throughput tradeoff.

Multimedia Databases. The multimedia database community has long studied content-based image retrieval (CBIR), whose goal is to retrieve images or videos from a database that have the same "content" as a query image [3, 32, 63, 68, 31, 49, 30, 57]. The notion of content is application-specific. Early CBIR works used hand-crafted vision features (e.g., SIFT) but recent ones showed that CNN features improve accuracy. Panorama's focus is *not* on CBIR but rather video monitoring applications. That said, our design decision of using embedding extraction to tackle unbounded vocabularies is inspired by work on CBIR. To the best of our knowledge, ours is the first work to exploit this connection between multimedia DB techniques and video monitoring.

Video Querying Systems. Video monitoring systems have seen a resurgence of interest in the DB and systems literature. NoScope [34] creates a model cascade with simple filters and a specialized cheaper CNN to improve querying efficiency compared to a larger reference CNN. Focus [27] splits video monitoring into ingesting and query stages to enable more accuracy-efficiency tradeoffs, including indexing objects offline and using them to speed up queries. BlazeIt [33] proposes an SQL-like language for selection and aggregation queries over frames and uses approximation techniques to improve efficiency. All these systems support only binary or finite multi-class vocabularies, which make them complementary to Panorama. Nevertheless, our work on Panorama was inspired by these systems, and we fundamentally expand video monitoring functionality to unbounded vocabularies while ensuring system efficiency.

Among video analytics systems, CaTDet [45] reduces inference costs by computing regions of interests based on historic detections. FilterForward [13] uses constrained edge nodes better. VideoStorm [66] and Optasia [44] are large-scale video analytics systems that aim to reduce latency. RAM^{3}3S[7], KDEDisStrOut[69], and other research[59] aim at real-time video analytics from massive multimedia streams. All these systems are orthogonal to Panorama, since they focus on better resource management and parallelism for analytics queries, not enabling unbounded vocabularies for monitoring queries. We believe Panorama can be integrated with such systems in the future.

7. CONCLUSION AND FUTURE WORK

The success of deep CNNs presents new opportunities for querying video data. However, most off-the-shelf models and video querying systems assume the prediction vocabulary is bounded, impeding many emerging video monitoring applications. In response, we present a new data system architecture, Panorama, for unbounded vocabulary querying of video. Panorama saves users the hassle of retraining models post deployment as the vocabulary grows. It is based on a multi-task and unified architecture, and its deployment is end-to-end automated and domain-agnostic. Relative to bespoke domain-specific models, Panorama's unified system offers competitive accuracy but with higher throughput. As for future work, we plan to support more kinds of video analytics tasks beyond verification and recognition.

APPENDIX

A. STEM$_1$

Figure 15 shows Stem$_1$.

B. YOLOV2 LOSS

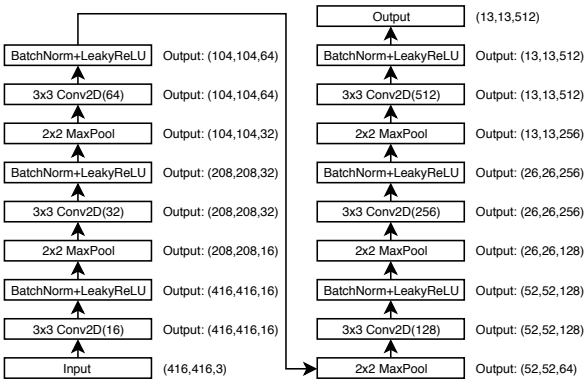

Figure 15: Detailed architecture of Stem_1 block from Figure 7. Max pooling layers have a stride of 2 and are valid-padded; other layers have a stride of 1 and are same-padded.

The Yolov2 loss is:

$$
\begin{aligned}
l_k = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 . \quad (6)
\end{aligned}
$$

C. REFERENCES

[1] NPR: Facial Recognition In China Is Big Business As Local Governments Boost Surveillance. https://www.npr.org/sections/parallels/2018/04/03/598012923/facial-recognition-in-china-is-big-business-as-local-governments-boost-surveilla, 2018. [Online; accessed May 2019].

[2] The Economic Times: Computer Vision for Crowd Control at India's Kumbh Mela. https://economictimes.indiatimes.com/news/politics-and-nation/higher-budget-and-bigger-ground-this-years-kumbh-mela-is-set-to-begin-with-a-bang/articleshow/67397579.cms, 2019. [Online; accessed May 2019].

[3] D. A. Adjeroh and K. C. Nwosu. Multimedia database management requirements and issues. In *IEEE MultiMedia*, volume 4, pages 24–33, 1997.

[4] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid. Label embedding for attribute-based classification. In *CVPR*, 2013.

[5] H. N. Anh. keras-yolo2. https://github.com/experiencor/keras-yolo2, 2018. [Online; accessed 20-March-2018].

[6] E. Bart and S. Ullman. Cross-generalization: learning novel classes from a single example by feature replacement. In *CVPR*, 2005.

[7] I. Bartolini and M. Patella. A general framework for real-time analysis of massive multimedia streams. *Multimedia Systems*, 24(4):391–406, Jul 2018.

[8] A. Bendale and T. Boult. Towards open world recognition. In *CVPR*, 2015.

[9] B. Brouwer. YouTube Now Gets Over 400 Hours Of Content Uploaded Every Minute. https://www.tubefilter.com/2015/07/26/youtube-400-hours-content-every-minute/, 2015. [Online; accessed 19-Jan-2019].

[10] Z. Cai, M. J. Saberian, and N. Vasconcelos. Learning complexity-aware cascades for deep pedestrian detection. In *ICCV*, pages 3361–3369, 2015.

[11] Z. Cai and N. Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *CVPR*, 2018.

[12] R. J. G. B. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, editors, *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[13] C. Canel, T. Kim, G. Zhou, C. Li, H. Lim, D. G. Andersen, M. Kaminsky, and S. R. Dulloor. Scaling video analytics on constrained edge nodes. In *SysML*, 2019.

[14] CBS. CBS News Live. https://www.cbsnews.com/live/, 2019. [Online; accessed 08-October-2019].

[15] X. Chen, A. Shrivastava, and A. Gupta. Neil: Extracting visual knowledge from web data. In *ICCV*, pages 1409–1416, 2013.

[16] Y. Cui, Y. Song, C. Sun, A. Howard, and S. J. Belongie. Large scale fine-grained categorization and domain-specific transfer learning. *CVPR*, pages 4109–4118, 2018.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. FeiFei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[19] L. Fei-Fei, R. Fergus, and P. Perona. A bayesian approach to unsupervised one-shot learning of object categories. In *ICCV*, 2003.

[20] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. In *IEEE TPAMI*, 2006.

[21] F. Fleuret and G. Blanchard. Pattern recognition from one example by chopping. In *NIPS*, 2005.

[22] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013.

[23] Y. Fu and L. Sigal. Semi-supervised vocabulary-informed learning. In *CVPR*, 2016.

[24] I. Goodfellow et al. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[25] S. Han, H. Shen, M. Philipose, S. Agarwal, A. Wolman, and A. Krishnamurthy. Mcdnn: An

approximation-based execution framework for deep stream processing under resource constraints. In *MobiSys*, pages 123–136, 2016.

[26] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, Oct 2017.

[27] K. Hsieh, G. Ananthanarayanan, P. Bodík, S. Venkataraman, P. Bahl, M. Philipose, P. B. Gibbons, and O. Mutlu. Focus: Querying large video datasets with low latency and low cost. In *13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018.*, pages 269–286, 2018.

[28] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations*, 2018.

[29] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

[30] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *Proceedings of the 10th European Conference on Computer Vision: Part I*, ECCV '08, pages 304–317, Berlin, Heidelberg, 2008. Springer-Verlag.

[31] Y. Jing, D. Liu, D. Kislyuk, A. Zhai, J. Xu, J. Donahue, and S. Tavel. Visual search at pinterest. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1889–1898, New York, NY, USA, 2015. ACM.

[32] O. Kalipsiz. Multimedia databases. In *IEEE Conference on Information Visualization. An International Conference on Computer Visualization and Graphics*, 2000.

[33] D. Kang, P. Bailis, and M. A. Zaharia. Blazeit: Fast exploratory video queries using neural networks. *CoRR*, abs/1805.01046, 2018.

[34] D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia. Noscope: optimizing neural network queries over video at scale. In *VLDB*, volume 10, pages 1586–1597, 2017.

[35] A. Karpathy. Software 2.0. `https://medium.com/@karpathy/software-2-0-a64152b37c35/`, 2017. [Online; accessed 13-March-2019].

[36] C. Lab. Cornell Lab FeederWatch Cam at Sapsucker Woods. `http://cams.allaboutbirds.org/channel/40/Cornell_Lab_FeederWatch_Cam/`, 2019. [Online; accessed 08-October-2019].

[37] B. M. Lake and R. Salakhutdinov. One-shot learning by inverting a compositional causal process. In *NIPS*, 2013.

[38] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009.

[39] C. H. Lampert, H. Nickisch, and S. Harmeling. Attribute-based classification for zero-shot visual object categorization. In *IEEE TPAMI*, pages 453–465, 2013.

[40] Y. LeCun et al. Deep learning. *nature*, 521(7553):436, 2015.

[41] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeply supervised nets. In *AISTATS*, 2015.

[42] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *CVPR*, pages 5325–5334, 2015.

[43] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *ECCV 2014*, pages 740–755, 2014.

[44] Y. Lu, A. Chowdhery, and S. Kandula. Optasia: A relational platform for efficient large-scale video analytics. In *SoCC*, 2016.

[45] H. Mao, T. Kong, and W. J. Dally. Catdet: Cascaded tracked detector for efficient object detection from video. *CoRR*, abs/1810.00434, 2018.

[46] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. In *ICLR*, 2014.

[47] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *Proceedings of the British Machine Vision*, volume 1, pages 41.1–41.12, 2015.

[48] A. Pentina and C. H. Lampert. A pac-bayesian bound for life-long learning. In *ICML*, pages II–991–II–999, 2014.

[49] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[50] A. J. Ratner, B. Hancock, and C. Ré. The role of massively multi-task and weak supervision in software 2.0. In *CIDR*, 2019.

[51] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. pages 779–788, 06 2016.

[52] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In *CVPR*, pages 6517–6525, 2017.

[53] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IJCV*, pages 211–252, 2015.

[54] W. J. Scheirer, L. P. Jain, and T. E. Boult. Probability models for open set recognition. In *IEEE TPAMI*, 2014.

[55] W. J. Scheirer, A. Rocha, A. Sapkota, and T. E. Boult. Towards open set recognition. In *IEEE TPAMI*, 2013.

[56] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823, 2015.

[57] Sivic and Zisserman. Video google: a text retrieval approach to object matching in videos. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1470–1477 vol.2, Oct 2003.

[58] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *CVPR*,

pages 3476–3483, 2013.

[59] M. Tang, S. Pongpaichet, and R. Jain. Research challenges in developing multimedia systems for managing emergency situations. In *ACM Multimedia*, 2016.

[60] S. Thrun and T. M. Mitchell. Lifelong robot learning. In *Robotics and Autonomous Systems*, volume 15, pages 25 – 46, 1995.

[61] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, volume 1, pages I–I, 2001.

[62] X. Wan, Y. Luo, D. Crankshaw, A. Tumanov, and J. E. Gonzalez. Idk cascades: Fast deep learning by learning not to overthink. In *UAI*, 2018.

[63] F. Yang, A. Kale, Y. Bubnov, L. Stein, Q. Wang, H. Kiapour, and R. Piramuthu. Visual search at ebay. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 2101–2110, New York, NY, USA, 2017. ACM.

[64] L. Yang, P. Luo, C. C. Loy, and X. Tang. A large-scale car dataset for fine-grained categorization and verification. In *CVPR*, pages 3973–3981, 2015.

[65] L. Yang, P. Luo, C. C. Loy, and X. Tang. A large-scale car dataset for fine-grained categorization and verification(tech report). Technical Report CNS-TR-2011-001, The Chinese University of Hong Kong, 2015.

[66] H. Zhang, G. Ananthanarayanan, P. Bodik, M. Philipose, P. Bahl, and M. J. Freedman. Live video analytics at scale with approximation and delay-tolerance. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pages 377–392, Boston, MA, 2017. USENIX Association.

[67] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, 23:1499–1503, Oct. 2016.

[68] Y. Zhang, P. Pan, Y. Zheng, K. Zhao, Y. Zhang, X. Ren, and R. Jin. Visual search at alibaba. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 993–1001, New York, NY, USA, 2018. ACM.

[69] Z. Zheng, H.-Y. Jeong, T. Huang, and J. Shu. Kde based outlier detection on distributed data streams in multimedia network. *Multimedia Tools and Applications*, 76:18027–18045, 2016.

[70] Z.-H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 08 2017.