Selectivity Estimation for Semantic Filters on Image Data

Matthias Urban
Technical University of Darmstadt
Darmstadt, Hesse, Germany
matthias.urban@tu-darmstadt.de

Vu Huy Nguyen
Technical University of Darmstadt
Darmstadt, Hesse, Germany
vu_huy.nguyen@stud.tu-darmstadt.de

Gabriele Sanmartino
EURECOM
Biot, France
gabriele.sanmartino@eurecom.fr

Paolo Papotti
EURECOM
Biot, France
papotti@eurecom.fr

Carsten Binnig
TU Darmstadt, DFKI, HessianAI
Darmstadt, Hesse, Gemany
carsten.binnig@tu-darmstadt.de

ABSTRACT

Semantic data systems integrate Large Language Models (LLMs) and Vision-Language Models (VLMs) directly into database query execution, enabling expressive queries on multi-modal data. However, optimizing these queries requires accurate selectivity estimates to determine the most efficient operator execution order. Contemporary systems rely on online sample-based profiling, a process that incurs severe latency overheads and struggles with low-selectivity queries. In this paper, we introduce *Semantic Histograms*, a novel selectivity estimator for semantic filters on image data that leverages shared embedding spaces to bypass traditional profiling. We realize that all semantic filters are implicit range queries, as they match a range of different images. Some filter predicates are more general, yielding a wide range, while others are more specific, yielding a smaller range. To address the challenge of implicit ranges, we propose two approaches to estimate the queries' specificity, with an ensemble of the two performing best. The evaluation shows that Semantic Histograms can reduce the end-to-end runtime overhead of query optimization and execution by up to 86%.

VLDB Workshop Reference Format:
Matthias Urban, Vu Huy Nguyen, Gabriele Sanmartino, Paolo Papotti, and Carsten Binnig. Selectivity Estimation for Semantic Filters on Image Data. VLDB 2026 Workshop: Applied AI for Database Systems and Applications (AIDB 2026).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/DataManagementLab/semantic_histograms.

1 INTRODUCTION

Query Optimization in Semantic Data Systems. Recently, semantic data systems have emerged in both academia [4, 8, 9, 13, 16, 18–20, 22, 25–27, 29, 31, 32] and industry [1, 2, 5]. Fundamentally, these systems directly integrate Large Language Models (LLMs) and Vision Language Models (VLMs) into database query execution, enabling them to query large collections of multi-modal data. For instance, in an e-commerce dataset, an analyst might search for product listings where a customer-uploaded image shows both a *broken screen* and a *warranty void sticker*. This query requires two distinct semantic filters to be evaluated by a VLM.

When optimizing such a query, the execution order of these filters significantly impacts runtime. If 90% of the images feature a warranty void sticker, but only 0.1% of images show broken screens, applying the *broken screen* filter first is optimal as it minimizes the number of expensive model calls. The optimal execution depends on two main factors: (1) the latency of the VLM and (2) the selectivity of each semantic filter. While latency can be profiled offline, selectivity is highly query-dependent and must be determined at runtime.

The Overhead of Online Profiling. Contemporary semantic data systems usually rely on an online profiling phase to estimate selectivity [18, 25, 26, 32]. In this phase, a small sample of data points (e.g., 1% of images) is processed by the model to extrapolate the filter's behavior. However, even a 1% sample induces substantial overhead on large datasets; profiling a single filter on 1 million images would require 10,000 VLM calls. Furthermore, sampling struggles to provide accurate estimates when the data is skewed or the target selectivity is low, leading to suboptimal query optimization decisions and additional overhead.

Semantic Histograms via Embedding Spaces. To avoid these runtime overheads, traditional RDBMSs utilize histograms to estimate operator selectivity. Histograms are compact, offline-computed structures that capture a column's distribution, allowing the system to estimate filter selectivity without scanning the table. In this paper, we explore how to build a *Semantic Histogram* for image data: a data structure that provides accurate selectivity estimates for visual semantic filters with low latency.

Traditional one-dimensional partitioning is inapplicable to inherently high-dimensional visual data. A naive approach would precompute filter outcomes offline to populate standard histograms. However, in semantic data systems, filter predicates are rarely known in advance; thus, this approach fails once the user applies new, unseen predicates. Therefore, we base our Semantic Histogram on embedding spaces provided by models like CLIP or SigLIP [24, 30, 37]. These models map images and text into a shared, high-dimensional vector space where semantic similarity corresponds to vector proximity. By calculating the cosine distance between a text filter predicate embedding (e.g., *broken screen*) and

Proceedings of the VLDB Endowment. ISSN 2150-8097.

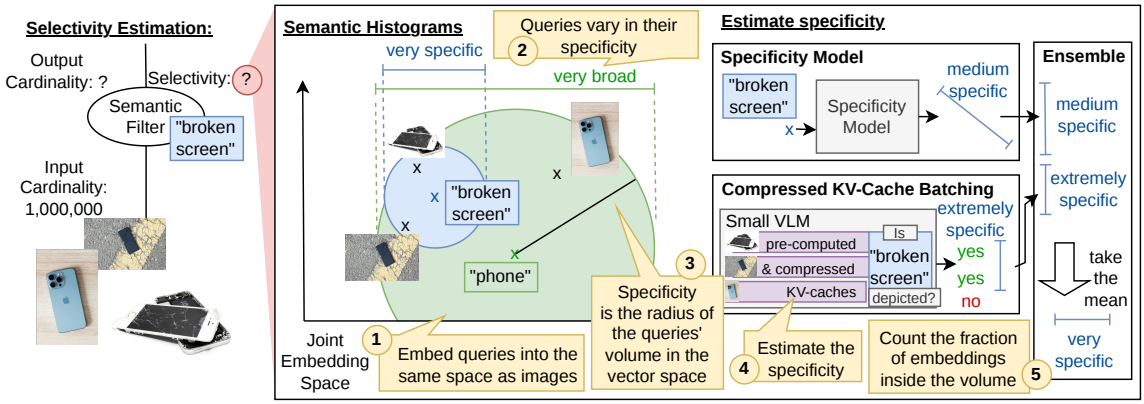

Figure 1: (Left) The semantic filter selects those pictures that depict a *broken screen*. Thus, the filter selectivity is the fraction of images in the dataset that depict a broken screen. (Right) Semantic Histograms for selectivity estimation: ① we embed all images (offline) and the filter predicate *broken screen* into a joint embedding space. High proximity in the vector space means a match is likely. ② Some predicates are more specific, intuitively covering a smaller volume in the vector space, while others are broader, covering a larger volume, like e.g., *phone*. To estimate the specificity (= the volume's radius ③) of a filter predicate, we propose three methods: ④ The specificity model takes a predicate embedding as input and outputs a cosine distance threshold that is equivalent to the radius of the volume; if the distance between an image and a predicate embedding is smaller than it, they count as a match. The compressed KV-Cache batching approach optimizes LLM inference on a large set of images to obtain the actual VLM response for these images, which we use to compute a threshold as well. The ensemble averages the two thresholds to improve robustness. ⑤ Finally, we count the matches and compute the selectivity.

the distribution of image embeddings, we can identify which and how many images most closely match the filter predicate.

Filter Predicates as Semantic Ranges. However, turning these distances into accurate selectivities is non-trivial. If we could define a strict distance threshold such that only images within the threshold match the filter predicate, we could easily estimate selectivity. Yet, an optimal threshold depends heavily on the specific filter predicate. A helpful way to frame this is to view all filter predicates as **range queries** within the embedding space. A broad filter predicate, such as *phone*, conceptually covers a larger semantic volume (a wider range), meaning images further from the embedding are still relevant, thus requiring a higher distance threshold. Conversely, a highly specific filter predicate, such as *IPhone X with broken screen*, maps much closer to its target images due to the contrastive training of embedding models.

Calibrating Thresholds for Semantic Histograms. To overcome the challenge of unknown specificity, we propose two methods for estimating the threshold. In the evaluation, we demonstrate that an ensemble of the two methods works most robustly across datasets.

The first leverages a lightweight *specificity model* trained on hierarchical visual labels to generalize across diverse filter predicates. This model can directly map from an input filter-predicate embedding to a threshold with a fraction of the cost of a VLM call, yielding low-latency selectivity estimates that are often of high quality in our experiments.

The second estimates the threshold empirically. It computes the actual VLM response on a large predefined sample and heavily optimizes the inference using a single, massive forward pass over pre-computed and compressed KV-caches of the images. Using these intermediate image representations, the system can empirically "peek" at the data distribution at a fraction of the cost of standard inference. Moreover, instead of using the selectivity on the sample, we use it to compute a distance threshold, enabling more accurate predictions even when the sample contains no positive examples. We show that this approach has slightly higher latency than the first method, but yields more robust estimates.

Contributions and outline. In summary, this paper makes the following contributions: (1) *Semantic Histograms:* We introduce a novel selectivity estimator for semantic filters on image data that leverages shared embedding spaces. To accurately capture filter-predicate specificity, we propose several threshold calibration techniques: a low-latency specificity model, a highly efficient compressed KV-cache batching method, and a robust ensemble of the two. All bypass the overhead of traditional online profiling.
(2) *Empirical Evaluation:* We evaluate our proposed approaches across three image datasets from Caesura [31] and Sembench [15], demonstrating that Semantic Histograms yield highly accurate selectivity estimates while significantly reducing query optimization latency compared to standard sample-based profiling.

The remainder of this paper is organized as follows: Section 2 provides an overview of Semantic Histograms, and Section 3 details how we deal with predicates of different specificity. Section 4 presents our empirical evaluation, while related and future work, as well as the conclusion, are presented in Sections 5 and 6.

2 OVERVIEW

In this section, we provide an overview of how we accurately estimate the selectivities of semantic filters on image data with low latency. These selectivity estimates can be used to optimize semantic queries involving multiple semantic operators, as we demonstrate in Section 4. The selectivity estimation is done in two phases: (1) offline, we pre-compute a synopsis of the available image data using embedding models such as CLIP [24] or SigLIP [30, 37], and (2) online, we use these embeddings for selectivity estimation.

2.1 Offline Embedding Computation

Offline, we need to capture the semantics of the images in our dataset in a way that allows for arbitrary filter predicates. For instance, the user could query for *phone with broken screen*, and we need to estimate the selectivity even when we have never seen that or a similar filter predicate before. To do so, we use SigLIP2 [30] to first map all images into a high-dimensional embedding space that captures their semantics. These enable us to predict the selectivity of arbitrary filter predicates, because filter predicates can be mapped into the same embedding space, where semantically similar images are close to the filter-predicate embedding.

In our Semantic Histogram, there is no concept of buckets as in regular histograms; we simply store all pre-computed embeddings as-is. However, we did experiment with bucketizing the embeddings by fitting density estimators, such as Gaussian Mixture Models, to the embedding vectors, or by clustering them. However, we found that simply keeping all embeddings yielded the best results: Their storage footprint is minimal (4.5 kB/embedding) compared to the original images (which need to be stored anyway), and computing the cosine similarity between a large number of embeddings is orders of magnitude faster than LLM calls (which happen anyway during query execution). We found that bucketizing can sometimes reduce the accuracy of selectivity estimates, leading to suboptimal query optimization choices and runtime overhead.

2.2 Online Selectivity Estimation

Figure 1 shows an overview of how Semantic Histograms compute selectivity estimates of semantic filters. Selectivity is the fraction of images in the dataset that match the filter's predicate, e.g., the fraction of images that depict a *broken screen* (Figure 1, left). To estimate the selectivity, we extract the filter's predicate (e.g., *broken screen*), and embed it into the same embedding space as the images, using SigLIP2 [30] again (See ① in Figure 1).

However, to decide how many image embeddings match the predicate embedding, we have to consider that the filter predicates differ in their *specificity* (See ②). For instance, a filter predicate such as *broken screen* is more specific than a broader one such as *phone*. Intuitively, we want to map the filter predicates to a volume in the vector space (rather than a single point), and have all image embeddings within the volume match the predicate. The specificity then corresponds to the radius of that volume (See ③); a highly specific predicate corresponds to a smaller radius, and a less specific predicate corresponds to a larger radius. Thus, to decide how many image embeddings are inside the volume, we compute the cosine distances from the predicate embedding, which can be done orders of magnitude faster than calling an LLM. An image embedding is inside the volume if it is smaller than a distance threshold equivalent to the volume's radius.

To estimate the distance threshold (i.e., the radius) of a filter predicate, we present two approaches (See ④ in Figure 1). Moreover, as we show in our evaluation, an ensemble of the two approaches yields particularly robust selectivity estimates across all datasets we tested.

The first approach is a simple, small neural network trained to estimate the specificities of filter predicates. Its input is a filter predicate text embedding, and it directly outputs the distance threshold. We find that this approach yields very fast estimates that are very accurate as long as the data at hand is somewhat similar to the specificity model's training data.

The second approach, compressed KV-cache batching, is more robust. The idea is to select a large, diverse sample of images and optimize the inference on this sample as much as possible, so we can quickly decide which images in the sample match the filter predicate. However, sampling notoriously struggles with low-selectivity filters. Thus, instead of simply using the selectivity on the sample, which would be 0 if no image in the sample matches, we instead use it to compute a threshold. That way, when there are no sample matches, we know it is a very specific filter predicate, resulting in a small distance threshold. If any images match the filter predicate, they will likely be inside the volume defined by the threshold, leading to a strictly positive, and often more accurate, selectivity estimate. We explain the details of the specificity model and the compressed KV-cache batching in Section 3.

Finally, after estimating the distance threshold, we can use it to compute the filter selectivity (See ⑤ in Figure 1). We simply return the fraction of image embeddings that are closer to the filter predicate embedding than the threshold.

3 THE SPECIFICITY CHALLENGE

Estimating selectivities of semantic filters comes with a unique challenge: Some filter predicates are more specific, like for instance *IPhone X with broken screen*, and others are less specific, like for instance *phone*. In this section, we explain how to estimate the specificity of a filter predicate, which is essential for computing filter selectivities, as explained before. All approaches output a cosine distance threshold as a measure of specificity. A small distance threshold leads to the predicate covering a smaller volume in the vector space, and thus corresponds to more specific filter predicates. Conversely, a large distance threshold corresponds to less specific predicates. The presented approaches differ in their strengths. However, our final ensemble approach, in particular, yields very accurate and robust selectivity estimates, as shown in our evaluation.

3.1 Specificity Model

The specificity model is a small neural network that maps a predicate embedding to its distance threshold.

Model Training. However, training the model is challenging. For training it, we need a large dataset that pairs text embeddings with distance threshold labels. To this end, we use ImageNet [6], which is a large collection of labeled images. Importantly, each image has not only a single label but a hierarchy of labels, inherited from Wordnet [21], ranging from specific concepts (e.g., cellular telephone) to broader concepts (e.g., telephone, electronic device). We use a subset consisting of 150,000 photographs divided into 1000 categories [12]. This allows us to construct a specificity dataset as follows: First, we randomly sample a subset of the data and several WordNet concepts that appear in it. For instance, we might have sampled the concepts of *cellular phone* and *electronic device*. Then we compute the label threshold for each concept. For instance, for *cellular phone*, we have 10 images in our subset that match that label. Then we compute the threshold label for training, such that exactly 10 image embeddings in the subset are closer to the embedding of

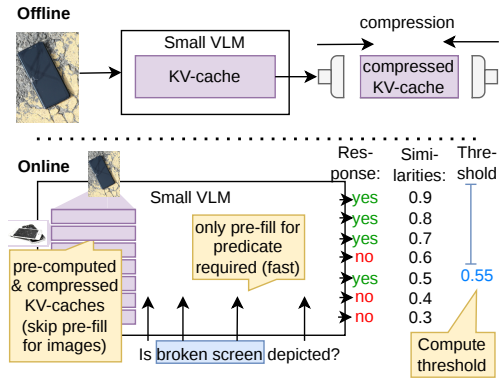

Figure 2: (Top) Offline, we feed a selected number of images into a small VLM (e.g., LLaVA-NeXT 8B [17]) and extract and compress the KV-cache. Afterward, we load the compressed KV-caches onto the GPU. (Bottom) Online, a filter predicate comes in. We can skip the prefill phase for the images and quickly compute the responses for all selected images at once. We also compute the similarities between the selected images and the predicate to compute a similarity threshold, which we then use for selectivity estimation.

cellular phone than the threshold. We repeat this process until we have more than 5000 training samples.

Limitations. While this leads to very fast threshold estimates, it has limitations stemming from the dataset choice. ImageNet focuses heavily on animals (especially different dog breeds), and the concepts are often scientific. In the future, we want to fine-tune the embedding models themselves to return not only embeddings but also the corresponding similarity thresholds, using a much broader dataset spanning more domains than ImageNet.

3.2 Compressed KV-cache Batching

While the first method yields fast estimates that are accurate when the dataset at hand does not deviate too much from ImageNet, we also offer a more robust method. To do so, we compute the actual VLM responses on a well-chosen subset of images to determine the threshold and make the VLM inference as fast as possible.

Threshold estimation. To estimate the threshold, we ask the VLM whether, e.g., a broken screen is depicted on a sample of images. We also compute the similarities of *broken screen* with all images in the sample. If, for instance, 10 images depict a broken screen, we choose the distance threshold so that exactly 10 images in the sample have a cosine distance smaller than the threshold. If no image in the sample matches the filter predicate, we set the threshold to the smallest observed distance, allowing us to still estimate selectivity for very low-selectivity predicates.

Making inference fast. However, if done naively, this does not lead to faster estimates than the online profiling phase typically used, which we want to eliminate. Thus, we optimize the inference to get many responses (in our case, up to 128) in a fraction of the time that typical inference would normally take for a single response. To do so, we first fix the sample to 128 images as explained in the next paragraph. Then, inspired by Stretto [26], we feed these images offline into the LLM and precompute and compress the model's KV cache, as shown in Figure 2 (top). Online, this allows us to skip image processing with the VLM by omitting the so-called prefill phase, see Figure 2 (bottom). Moreover, we compress these KV-caches using the Expected Attention press [7], so that all 128 embeddings take about 4 GB of space with 90% compression in our experiments. Thus, they can be preloaded onto the GPU before any query enters the system. Compression is lossy and degrades prediction quality, but allows for a large sample to be pre-loaded onto the GPU, leading to lower estimation latencies, which is beneficial.

When the query, e.g., with the filter *broken screen*, arrives, we only need two more VLM passes to obtain responses for all 128 images in a batch. First, we need to finish the prefill phase by computing the KV-cache of the prompt (i.e., Is *broken screen* depicted?), which usually consists of only a few tokens and is thus very fast. Afterward, we generate a single yes/no answer token for all images at once. Experiments show that this method yields robust threshold estimates with latencies smaller than a single LLM call.

Sample Selection. Finally, it is important that the sample used to estimate the threshold is diverse and representative, capturing many of the concepts present in the dataset. To select a diverse sample, we cluster the image embeddings using K-Means. We set the number of clusters to the sample size we want to obtain, e.g., up to 128 in our case. Then, we pick the images whose embeddings are closest to each centroid. Finally, we want to mention that this method also allows for larger sample sizes. However, it would consume more memory (about 30 MB per KV-cache using LLaVA-NeXT 8B [17] with 90% compression), potentially leaving some KV-caches not pre-loaded onto the GPU, requiring them to be loaded online.

3.3 Ensemble

Finally, we found that the two threshold estimation approaches make orthogonal mistakes. Combining the two approaches by averaging the two predicted thresholds often yields better selectivity estimates than either approach alone, as shown in our experiments.

4 EVALUATION

In our experiments, we show that the Semantic Histogram variants presented in this paper are both faster and more accurate than estimating selectivity via sampling in an online profiling phase, as is done by contemporary semantic data systems. Especially the ensemble yields robust estimates across all datasets with low latency, resulting in significantly faster end-to-end runtimes.

4.1 Experimental Setup

Datasets. As our datasets, we use three datasets introduced by prior work [15, 31] to evaluate semantic data systems. All contain a column of image data, which we can use to apply semantic filters to images and, more importantly, estimate the selectivities of these filters. *(1) Artwork:* A dataset constructed from Wikidata, that contains 1000 pictures of artworks from different epochs. The dataset was first introduced by Caesura [31]. *(2) Wildlife:* A dataset containing wildlife photographs of different animals from a conservancy in Kenya. We randomly select a subset of 1000 images. The dataset was first introduced by SemBench [15]. *(3) E-Commerce:* A dataset containing pictures of different fashion items. We select a subset of 1000 images. First introduced by Sembench [15].

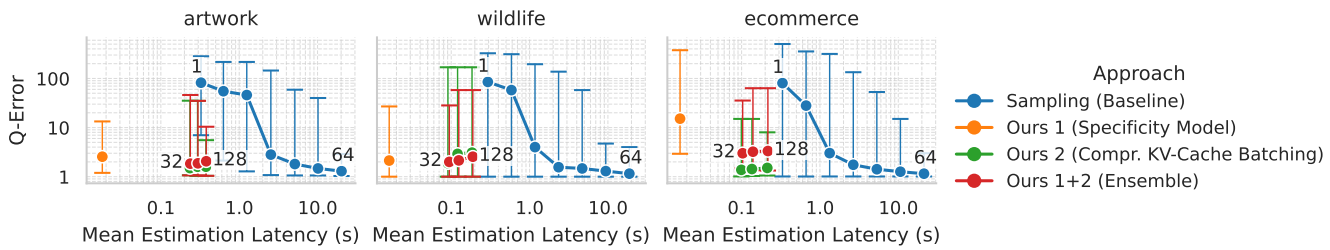

Figure 3: Semantic Histograms are much faster than sampling, while being very accurate. We plot the Q-Error (Y-axis) of different selectivity estimation approaches and configurations against their latency (X-axis). Good approaches are at the bottom (low Q-Error) and on the left (low latency) of each plot. For latency, we plot the mean across all filter predicates and seeds; for Q-Error, we plot the median and the 5th and 95th percentiles (as error bars). For sampling, we use sample sizes of 1, 2, 4, …, 64; for compressed KV-Cache batching, we use sample sizes of 32, 64, 128 (with compression rates of 0.6, 0.8, 0.9, respectively, to keep memory consumption approximately the same). We show which configuration is used next to the markers.

Queries and Operators. For each dataset, we generate a large set of filter predicates (14-26 per dataset) based on the original queries. We also ensure that we have predicates of different specificity; for instance, in the wildlife dataset, we have filter predicates such as *animal*, *prey animal*, *antelope*, and *impala*.

For executing the semantic filters, we use the Qwen 2.5 VL 7B model [3, 33], and the prompt *Is <filter predicate> depicted?* We deploy the model using ollama on an A100 GPU.

Metrics. To evaluate the quality of selectivity estimates, we use the Q-Error, which is the ratio of the predicted selectivity to the actual selectivity. For instance, if the true selectivity is 2% but a method predicts 20% or 0.2%, the Q-Error is 10 in both cases. To avoid undefined Q-errors when the predicted selectivity is 0, we set the predictions to $\frac{1}{\text{dataset\_size}}$ in this case. Moreover, we measure the latency of the different selectivity estimation methods and the time to execute semantic queries in seconds.

Approaches and Baselines. We test the different Semantic Histogram approaches presented and compare them with sampling in an online profiling phase for selectivity estimation. For sampling, we process a data sample with the VLM and estimate selectivity based on its responses. Note that we picked a small 7B VLM; a larger one would exacerbate sampling costs. We test a sample size of 1 (0.1% of dataset size), 2, 4, …, 64 (6.4% of dataset size).

Moreover, for the compressed KV-cache batching approach, we test several configurations that use the same amount of GPU memory: (1) we pre-compute the KV-cache of 128 images and compress them with a compression ratio of 90% (2) we pre-compute 64 images and use a compression ratio of 80%, and (3) we pre-compute 32 KV-caches and use a compression ratio of 60%. We use LLaVA-NeXT 8B [17], where the KV-caches for each setting are about 4 GB. We run all experiments with 20 seeds and average the results.

4.2 Fast and Accurate Selectivity Estimates

Summary. In our first experiment, we show that Semantic Histograms yield better selectivity estimates in a fraction of the estimation latency. The ensemble, in particular, yields excellent Q-errors across all datasets, with latencies lower than a single LLM call. The other Semantic Histogram variants are less robust but excel on particular datasets; for instance, the specificity model performs well on the wildlife dataset, likely because it contains pictures of animals, which also appear frequently in ImageNet. Compressed KV-Cache batching works excellently on e-commerce and is also quite robust.

Details. In Figure 3, we plot the Q-Error of selectivity estimation (Y-axis) against the latency to obtain the estimates (X-axis) of the different approaches. Low latency is important as any latency for estimating selectivity directly translates into runtime overhead. Low Q-Error is important for enabling good decisions when selectivity estimates are used, for instance, in query optimization. Even minor mispredictions can cause the query optimizer to pick the wrong operator order, resulting in runtime overhead.

First, we observe that all our approaches are substantially faster than naive *sampling* during an online profiling phase, depicted in blue. In particular, the *specificity model*, which does not involve any LLM call for selectivity estimation, is significantly faster than all other approaches. On average, it needs only 17 ms to make a prediction. But also, the *compressed KV-cache batching* approach and the *ensemble* are faster than sampling. For instance, when pre-loading 128 KV-caches onto the GPU and computing the LLM response for 128 samples in a single batch, we get these 128 responses in approximately the same time as it takes to evaluate a single sample.

At the same time, our predictions are often more accurate than sampling with reasonable sample sizes. In particular, we want to highlight the ensemble, which achieves good median Q-error scores of 1.8, 2.0, and 3.0 on the datasets and is particularly robust. Sampling needs to process about 1% of data with the VLM to obtain the same median (and p95 Q-error), which incurs substantial runtime overhead. The other approaches excel on particular datasets. For instance, the specificity model works very well on wildlife, likely because it is similar to ImageNet, which also contains many animals. On the other hand, the compressed KV-cache batching excels on the e-commerce dataset. We speculate that this is because these images are low-complexity, each showing a single product, enabling accurate predictions even with KV-cache compression.

As expected, the specificity model does not work well on all datasets. In particular, on e-commerce, we observe a high median Q-Error of 15.2, which is due to threshold mispredictions. However, ensembling helps to improve the Q-Error. Another observation worth noting is that the compressed KV-Cache Batching often predicts very low selectivity for wildlife, even for rather broad concepts such as *prey animal*, leading to a high p95 Q-Error. Upon investigation, we found that this is a problem with the underlying VLM we

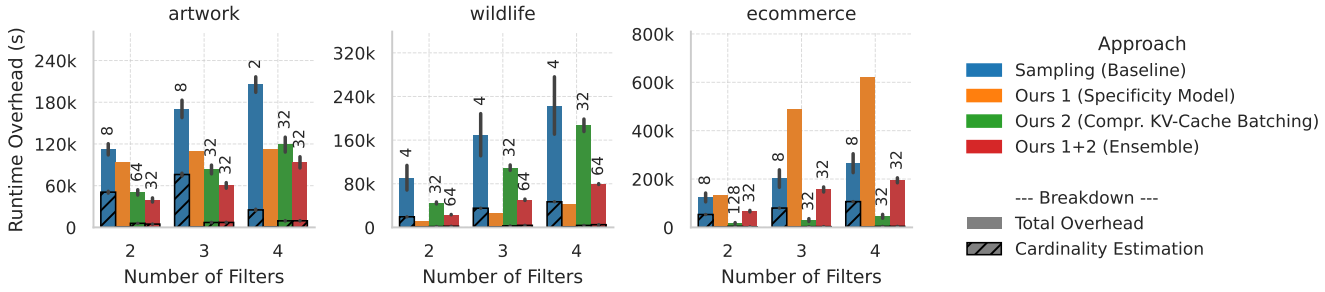

Figure 4: Using Semantic Histograms for query optimization leads to faster end-to-end runtime than sampling. We generate semantic queries with 2, 3, and 4 semantic filters (Y-axis) and plot the absolute runtime overhead of query optimization + execution when using the selectivity estimation methods. For each approach, we plot the mean overhead and its 95% confidence interval (error bars) for the best-performing configuration (e.g., the best-performing sample size, annotated above each bar) across different seeds. The overhead is the difference from a perfect baseline that uses a zero-latency oracle selectivity estimation. The ensemble approach for Semantic Histograms is most robust across datasets and works better than sampling on all datasets.

use (LLaVA-NeXT 8B), which is not very accurate on these images (potentially because the animals are often very small and in the distance), rather than a side-effect of KV-cache compression loss. Thus, we suspect that implementing this technique with more accurate models will yield more accurate selectivity estimates. However, again, ensembling helps reduce the p95 Q-Error.

4.3 End-to-End Runtime

Summary. In our second experiment, we show that the improved selectivity estimates with low latency lead to better end-to-end runtimes when used for query optimization. The ensemble yields the lowest overall end-to-end runtimes, with statistically significant speedups over naive sampling on all datasets.

Details. To understand the trade-off between accuracy and latency in selectivity estimation for semantic data systems, we use the selectivity estimates for query optimization and measure end-to-end runtime. In this setting, there are two types of overhead a bad selectivity estimator can have: (1) the latency for producing the selectivity estimate (2) the latency induced by producing bad estimates leading to a suboptimal plan. Thus, to achieve low end-to-end latency, the selectivity estimator must be both fast and accurate.

The experiment setup is as follows: For each dataset, we generate 300 semantic queries - 100 each of 2, 3, or 4 consecutive semantic filters, randomly sampled from the available filter predicates. During query optimization, we must determine the filter order. Ideally, we want to have the most selective filter first, so that we minimize the number of images the second filter must process, and so on. Thus, for each query and selectivity estimator, we estimate the selectivity of each filter in a query. Then we sort the filters by this value, putting the most selective filter first. Afterward, we run all filters on the full dataset in this order.

In Figure 4, we plot the sum of overheads over all queries of the end-to-end runtime versus an optimal query optimizer that has access to oracle selectivity estimates with zero latency. Overall, we see that our Semantic Histogram approaches are typically faster than sampling, with only a few exceptions. Moreover, the ensemble method is better than sampling across all datasets and filter counts. Usually, the error bars, which denote the confidence intervals of the mean overhead across different seeds, do not overlap with those of sampling, indicating that the result is statistically significant. Again, the other Semantic Histogram variants excel on individual datasets: the specificity model excels on wildlife, and the compressed KV-cache batching on e-commerce. Finally, the error bars of sampling are much larger than those of Semantic Histograms, indicating that the runtime of systems relying on sampling is highly unpredictable.

5 RELATED WORK

Selectivity Estimation. Traditionally, histograms or sketches have been used for selectivity estimation [e.g., 10, 23, 28]. More recently, also machine learning-based approaches have been explored [e.g., 11, 14, 35]. However, these approaches require the fixed schema provided by traditional RDBMSs. Concurrent work by Xu et al. [34] has explored how to build selectivity estimators for text data in semantic data systems. It does not address image data.

Semantic Data Systems. Recently, many semantic data systems have been proposed [1, 2, 4, 5, 8, 9, 13, 16, 18, 20, 22, 25, 26, 29, 31, 32]. Many include query optimizers that pick the best operator order [1, 5, 18, 20, 25, 26, 32], pick the model or combination of models with the best quality-runtime trade-off [1, 5, 20, 22, 25, 26], or pick the best prompting strategies [25], or rewrite the queries for better quality or lower latency [1, 29, 36]. However, these approaches rely on sampling when they estimate selectivities of semantic filters.

6 CONCLUSION AND FUTURE WORK

In this paper, we show that image embeddings can be used to build selectivity estimators for image data that yield low-latency and high-quality estimates. These can be used by semantic data systems for query optimization to achieve a lower end-to-end runtime than previous approaches that use sampling for selectivity estimation. In the future, we want to extend the approach towards other modalities, such as text or audio, and also other semantic operators, such as semantic joins. Moreover, some semantic data systems rely on sampling to estimate the *quality* of different operator implementations (e.g., using different LLM sizes). Therefore, future work must also look into how sampling can be replaced for these purposes.

ACKNOWLEDGMENTS

This work was funded by the DFG/ANR Project MAgiQ (ANR-24-CE92-0077; DFG, German Research Foundation – Project No. 545611510), the LOEWE Spitzenprofessur programme (III 5-519/05.00.003-(0005)), by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy (EXC-3057/1 "Reasonable Artificial Intelligence", Project No. 533677015), and by the French government, through the 3IA Côte d'Azur Investments in the IA-cluster project managed by the National Research Agency (ANR-23-IACL-0001). We also thank DFKI and hessian.AI.

REFERENCES

[1] Paritosh Aggarwal, Bowei Chen, Anupam Datta, Benjamin Han, Boxin Jiang, Nitish Jindal, Zihan Li, Aaron Lin, Pawel Liskowski, Jay Tayade, Dimitris Tsirogiannis, Nathan Wiegand, and Weicheng Zhao. 2025. Cortex AISQL: A Production SQL Engine for Unstructured Data. *CoRR* abs/2511.07663 (2025). https://doi.org/10.48550/ARXIV.2511.07663 arXiv:2511.07663

[2] Eric Anderson, Jonathan Fritz, Austin Lee, Bohou Li, Mark Lindblad, Henry Lindeman, Alex Meyer, Parth Parmar, Tanvi Ranade, Mehul A. Shah, Benjamin Sowell, Dan Tecuci, Vinayak Thapliyal, and Matt Welsh. 2025. The Design of an LLM-powered Unstructured Analytics System. In *15th Conference on Innovative Data Systems Research, CIDR 2025, Amsterdam, The Netherlands, January 19-22, 2025*. www.cidrdb.org. https://vldb.org/cidrdb/2025/the-design-of-an-llm-powered-unstructured-analytics-system.html

[3] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Ming-Hsuan Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. 2025. Qwen2.5-VL Technical Report. *CoRR* abs/2502.13923 (2025). https://doi.org/10.48550/ARXIV.2502.13923 arXiv:2502.13923

[4] Shu Chen, Deepti Raghavan, and Ugur Çetintemel. 2025. Continuous Prompts: LLM-Augmented Pipeline Processing over Unstructured Streams. *CoRR* abs/2512.03389 (2025). https://doi.org/10.48550/ARXIV.2512.03389 arXiv:2512.03389

[5] Yeounoh Chung, Rushabh Desai, Jian He, Yu Xiao, Thibaud Hottelier, Yves-Laurent Kom Samo, Pushkar Khadilkar, Xianshun Chen, Sam Idicula, Fatma Özcan, Alon Y. Halevy, and Yannis Papakonstantinou. 2026. 100x Cost & Latency Reduction: Performance Analysis of AI Query Approximation using Lightweight Proxy Models. *CoRR* abs/2603.15970 (2026). https://doi.org/10.48550/ARXIV.2603.15970 arXiv:2603.15970

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[7] Alessio Devoto, Maximilian Jeblick, and Simon Jégou. 2025. Expected Attention: KV Cache Compression by Estimating Attention from Future Queries Distribution. *CoRR* abs/2510.00636 (2025). https://doi.org/10.48550/ARXIV.2510.00636 arXiv:2510.00636

[8] Anas Dorbani, Sunny Yasser, Jimmy Lin, and Amine Mhedhbi. 2025. Beyond Quacking: Deep Integration of Language Models and RAG into DuckDB. *Proc. VLDB Endow.* 18, 12 (2025), 5415–5418. https://doi.org/10.14778/3750601.3750685

[9] Uélison Jean Lopes dos Santos, Alessandro Ferri, Szilard Nistor, Riccardo Tommasini, Carsten Binnig, and Manisha Luthra. 2026. Towards Multimodal Stream Processing Systems. In *Proceedings 29th International Conference on Extending Database Technology, EDBT 2026, Tampere, Finland, March 24-27, 2026*, Wolfgang Lehner, Vanessa Braganholo, Kostas Stefanidis, Zheying Zhang, Alexander Krause, and João Felipe Nicolaci Pimentel (Eds.). OpenProceedings.org, 627–633. https://doi.org/10.48786/EDBT.2026.51

[10] Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. 2007. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. *Discrete mathematics & theoretical computer science* Proceedings (2007).

[11] Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. 2019. DeepDB: Learn from Data, not from Queries! *CoRR* abs/1909.00607 (2019). arXiv:1909.00607 http://arxiv.org/abs/1909.00607

[12] Addison Howard, Eunbyung Park, and Wendy Kan. 2018. ImageNet Object Localization Challenge. https://kaggle.com/competitions/imagenet-object-localization-challenge. Kaggle.

[13] Saehan Jo and Immanuel Trummer. 2024. ThalamusDB: Approximate Query Processing on Multi-Modal Data. *Proc. ACM Manag. Data* 2, 3 (2024), 186. https://doi.org/10.1145/3654989

[14] Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter Boncz, and Alfons Kemper. 2019. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. In *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13-16, 2019, Online Proceedings*. www.cidrdb.org. https://vldb.org/cidrdb/2019/learned-cardinalities-estimating-correlated-joins-with-deep-learning.html

[15] Jiale Lao, Andreas Zimmerer, Olga Ovcharenko, Tianji Cong, Matthew Russo, Gerardo Vitagliano, Michael Cochez, Fatma Özcan, Gautam Gupta, Thibaud Hottelier, H. V. Jagadish, Kris Kissel, Sebastian Schelter, Andreas Kipf, and Immanuel Trummer. 2025. SemBench: A Benchmark for Semantic Query Processing Engines. *CoRR* abs/2511.01716 (2025). https://doi.org/10.48550/ARXIV.2511.01716 arXiv:2511.01716

[16] Alexander W. Lee, Benjamin Han, Shayak Sen, Samuel Yeom, Ugur Çetintemel, and Anupam Datta. 2026. Evergreen: Efficient Claim Verification for Semantic Aggregates. *CoRR* abs/2604.26180 (2026). https://doi.org/10.48550/ARXIV.2604.26180 arXiv:2604.26180

[17] Bo Li, Kaichen Zhang, Hao Zhang, Dong Guo, Renrui Zhang, Feng Li, Yuanhan Zhang, Ziwei Liu, and Chunyuan Li. 2024. LLaVA-NeXT: Stronger LLMs Supercharge Multimodal Capabilities in the Wild. https://llava-vl.github.io/blog/2024-05-10-llava-next-stronger-llms/

[18] Zequn Li, Yuanhao Zhong, Chengliang Chai, Zhaoze Sun, Yuhao Deng, Ye Yuan, Guoren Wang, and Lei Cao. 2025. DocDB: A Database for Unstructured Document Analysis. *Proc. VLDB Endow.* 18, 12 (2025), 5387–5390. https://doi.org/10.14778/3750601.3750678

[19] Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeighami, Aditya G. Parameswaran, and Eugene Wu. 2024. Towards Accurate and Efficient Document Analytics with Large Language Models. *CoRR* abs/2405.04674 (2024). https://doi.org/10.48550/ARXIV.2405.04674 arXiv:2405.04674

[20] Chunwei Liu, Matthew Russo, Michael J. Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael J. Franklin, Tim Kraska, Samuel Madden, Rana Shahout, and Gerardo Vitagliano. 2025. Palimpzest: Optimizing AI-Powered Analytics with Declarative Query Processing. In *15th Conference on Innovative Data Systems Research, CIDR 2025, Amsterdam, The Netherlands, January 19-22, 2025*. www.cidrdb.org. https://vldb.org/cidrdb/2025/palimpzest-optimizing-ai-powered-analytics-with-declarative-query-processing.html

[21] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41. https://doi.org/10.1145/219717.219748

[22] Liana Patel, Siddharth Jha, Melissa Z. Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. 2025. Semantic Operators and Their Optimization: Towards AI-Based Data Analytics with Accuracy Guarantees. *Proc. VLDB Endow.* 18, 11 (2025), 4171–4184. https://doi.org/10.14778/3749646.3749685

[23] Viswanath Poosala, Yannis E. Ioannidis, Peter J. Haas, and Eugene J. Shekita. 1996. Improved Histograms for Selectivity Estimation of Range Predicates. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, H. V. Jagadish and Inderpal Singh Mumick (Eds.). ACM Press, 294–305. https://doi.org/10.1145/233269.233342

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research)*, Marina Meila and Tong Zhang (Eds.). PMLR, 8748–8763. http://proceedings.mlr.press/v139/radford21a.html

[25] Matthew Russo, Chunwei Liu, Sivaprasad Sudhir, Gerardo Vitagliano, Michael J. Cafarella, Tim Kraska, and Samuel Madden. 2026. Abacus: A Cost-Based Optimizer for Semantic Operator Systems. *Proc. VLDB Endow.* 19, 5 (2026), 1060–1073. https://www.vldb.org/pvldb/vol19/p1060-russo.pdf

[26] Gabriele Sanmartino, Matthias Urban, Paolo Papotti, and Carsten Binnig. 2026. The Stretto Execution Engine for LLM-Augmented Data Systems. *CoRR* abs/2602.04430 (2026). https://doi.org/10.48550/ARXIV.2602.04430 arXiv:2602.04430

[27] Dario Satriani, Enzo Veltri, Donatello Santoro, Sara Rosato, Simone Varriale, and Paolo Papotti. 2025. Logical and Physical Optimizations for SQL Query Execution over Large Language Models. *Proc. ACM Manag. Data* 3, 3 (2025), 181:1–181:28. https://doi.org/10.1145/3725411

[28] Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. 1979. Access Path Selection in a Relational Database Management System. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, USA, May 30 - June 1*, Philip A. Bernstein (Ed.). ACM, 23–34. https://doi.org/10.1145/582095.582099

[29] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. 2025. DocETL: Agentic Query Rewriting and Evaluation for Complex Document Processing. *Proc. VLDB Endow.* 18, 9 (2025), 3035–3048. https://doi.org/10.14778/3746405.3746426

[30] Michael Tschannen, Alexey A. Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier J. Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. 2025. SigLIP 2: Multilingual Vision-Language Encoders with Improved Semantic Understanding, Localization, and Dense Features. *CoRR* abs/2502.14786

(2025). https://doi.org/10.48550/ARXIV.2502.14786 arXiv:2502.14786

[31] Matthias Urban and Carsten Binnig. 2024. CAESURA: Language Models as Multi-Modal Query Planners. In *14th Conference on Innovative Data Systems Research, CIDR 2024, Chaminade, HI, USA, January 14-17, 2024*. www.cidrdb.org. https://vldb.org/cidrdb/2024/caesura-language-models-as-multi-modal-query-planners.html

[32] Jiayi Wang and Guoliang Li. 2025. AOP: Automated and Interactive LLM Pipeline Orchestration for Answering Complex Queries. In *15th Conference on Innovative Data Systems Research, CIDR 2025, Amsterdam, The Netherlands, January 19-22, 2025*. www.cidrdb.org. https://vldb.org/cidrdb/2025/aop-automated-and-interactive-llm-pipeline-orchestration-for-answering-complex-queries.html

[33] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. 2024. Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. *CoRR* abs/2409.12191 (2024). https://doi.org/10.48550/ARXIV.2409.12191 arXiv:2409.12191

[34] Shihui Xu, Jiayi Wang, and Guoliang Li. 2026. Bridging the Gap: Cardinality Estimation for Semantic Queries on Unstructured Data. *Proceedings of the ACM on Management of Data* 4, 3 (SIGMOD (2026), 1–26.

[35] Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. 2019. Deep Unsupervised Cardinality Estimation. *Proc. VLDB Endow.* 13, 3 (2019), 279–292. https://doi.org/10.14778/3368289.3368294

[36] Sepanta Zeighami, Shreya Shankar, and Aditya G. Parameswaran. 2025. Cut Costs, Not Accuracy: LLM-Powered Data Processing with Guarantees. *Proc. ACM Manag. Data* 3, 6 (2025), 1–26. https://doi.org/10.1145/3769776

[37] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. 2023. Sigmoid Loss for Language Image Pre-Training. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*. IEEE, 11941–11952. https://doi.org/10.1109/ICCV51070.2023.01100