Accurate ML Processing under Real-Time Constraints

Phillip Hilliard, Zack Ives, Rajeev Alur, University of Pennsylvania

Penn Engineering
University of Pennsylvania

Streaming Queries with ML

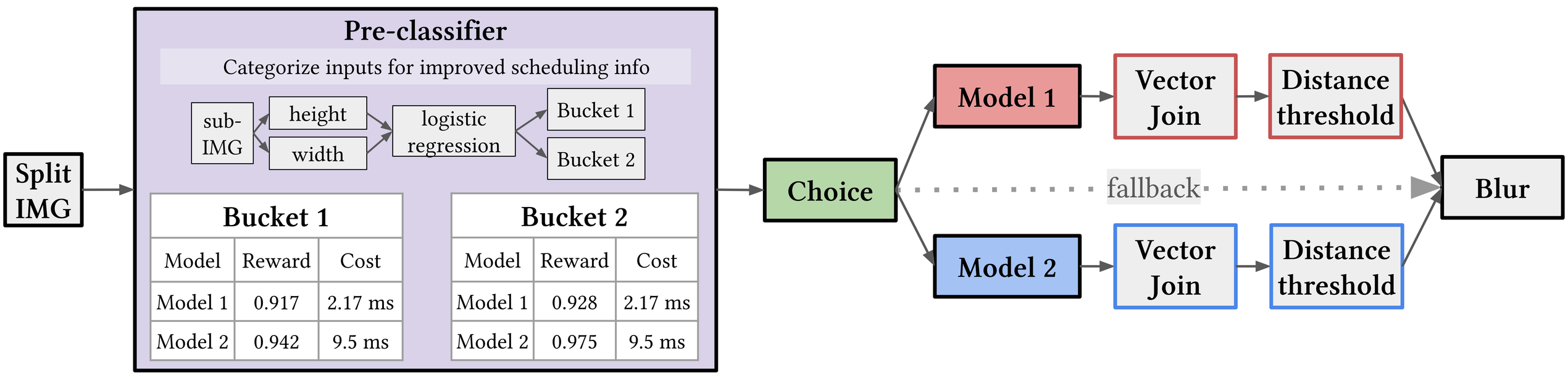
Modern applications use ML models to process data as part of a larger query. Tasks have multiple available models that trade *accuracy* for *runtime cost*. **No single model** can provide the best accuracy over varied data and at various throughput requirements.

Adaptive Query Processing

Scenario: An image stream splits into cropped faces. Faces are *embedded* and *joined* against an index. Matches are blurred afterwards. Adaptive Query Processing can help pick the best model in the ML operations, but the operator will be **on the critical path**.
Goal: Improve accuracy *without* exceeding deadlines and falling behind.

Successful Adaptivity

Forecasts of future inputs reveal short-term vs long-term opportunities. An *ultra-lightweight pre-classifier* gives very fast info and lets us meet deadlines! We adapt to remain competitive with the best model choice under varied input distributions, volumes, and velocities.



Problem Concerns

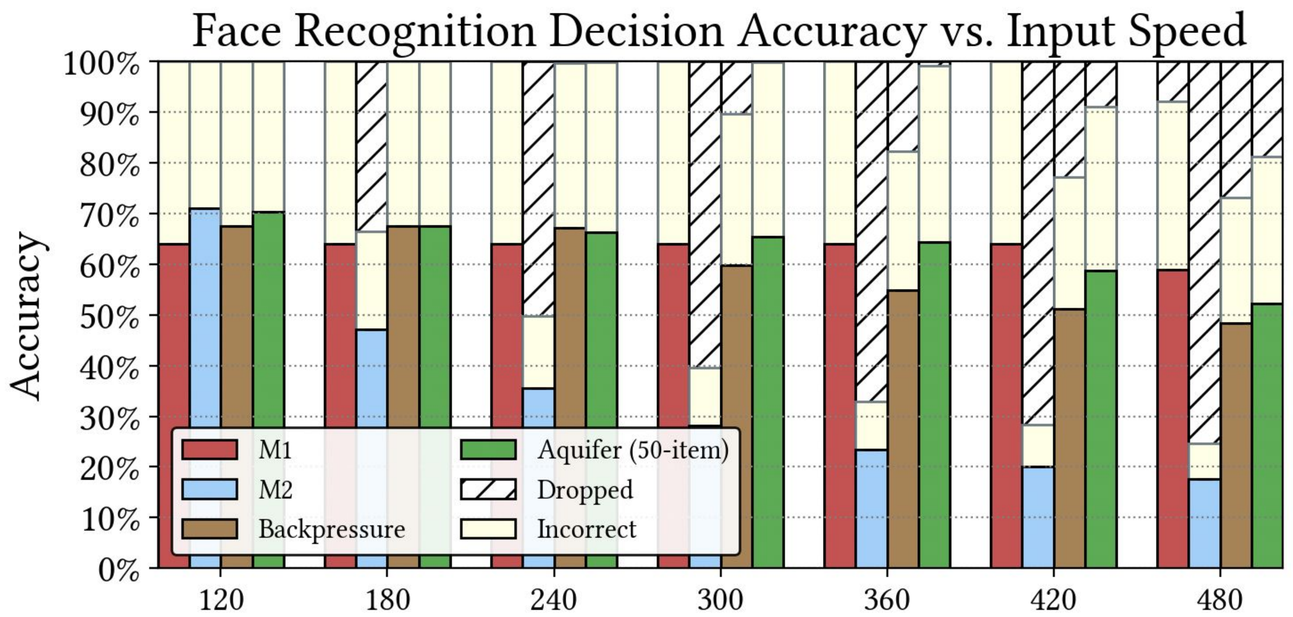
$$\text{Cost}_{\text{pre-model}} + \text{Cost}_{\text{model choice}} + \text{Cost}_{\text{post-model}} < \text{Deadline}$$

- Multiple model choices, accuracy vs cost
- Input processing deadline
- Streaming throughput requirement
- Fallback strategy reduces accuracy
- Model accuracy is not uniform over distribution

Solution Insights

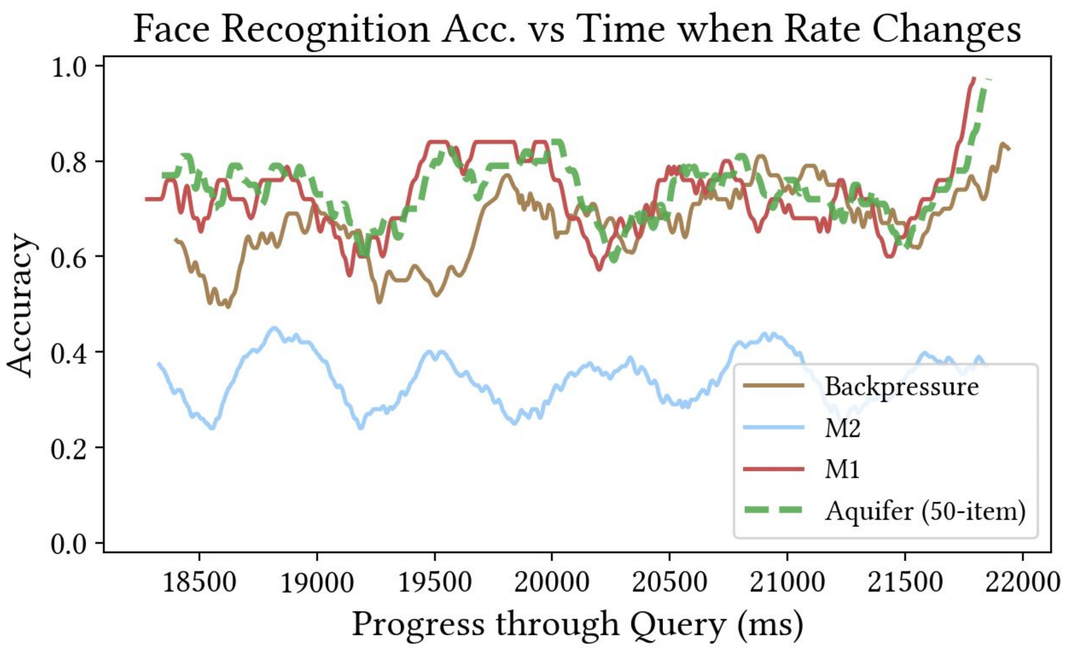
Simple & fast factors can still yield optimization opportunities!

- Simple metrics + logistic regression offline
- **Performance buckets** with meaningful differences
  - expected **reward** and **cost** if a model runs processes an input
  - information for scheduling adaptive decisions
- **Forecasting** a future window using past buckets + rates
- Greedy algorithm assigns model routes for inputs by bucket



Results and Takeaways

- Our approach (green) matches the optimal strategy across a variety of scenarios
- Forecasting+scheduling almost always provides more benefit than overhead
- Reduces “fallback” (0% acc.) cases while allowing calls to Model 2



What’s next?

More domains. We have results for at least two other modalities: time-series health data and text-based question-answering. We also intend to expand this to more queries.

Scheduling implementation. Bucket-based scheduling leaves opportunities to change the parameters. We currently make a new forecast and compute a new schedule for every input. What if we chose a different window size? What if we schedule multiple current inputs together? What if we reuse a schedule for subsequent inputs?

Quantifying uncertainty. Learned methods can offer measures of confidence, from distance thresholds to bucketing logits. Can queries make greater use of those probabilities, using lessons learned here in streaming and from elsewhere in probabilistic databases? Can we provide bounds based on what information has gone through the system so far? Can bounds be used as constraints in optimization problems like this one, and for optimizing other query processing problems like approximate query processing?

Penn Engineering
University of Pennsylvania