The University of Texas at Austin
Department of Computer Science
College of Natural Sciences

In Proceedings of ICML 2024

Online Cascade Learning for Efficient Inference over Streams

Lunyiu Nie, Zhimin Ding, Erdong Hu, Christopher Jermaine, Swarat Chaudhuri

Univerity of Texas at Austin, Rice University
lynie@utexas.edu

TL;DR

Reduces LLM inference costs by up to 90% while maintaining high accuracy and reliability in stream processing, by intersecting imitation learning & online learning.

Motivation

LLMs are widely used in commercial products for processing **data streams**.

Content Filtering

e.g. moderating user-generated content, email spam filtering, flagging hate speech

Decision Making

e.g. financial investment analysis, healthcare diagnosis support, HR and recruitment

Recommendation System

e.g., e-commerce product recommendation, streaming platforms, travel and hospitality

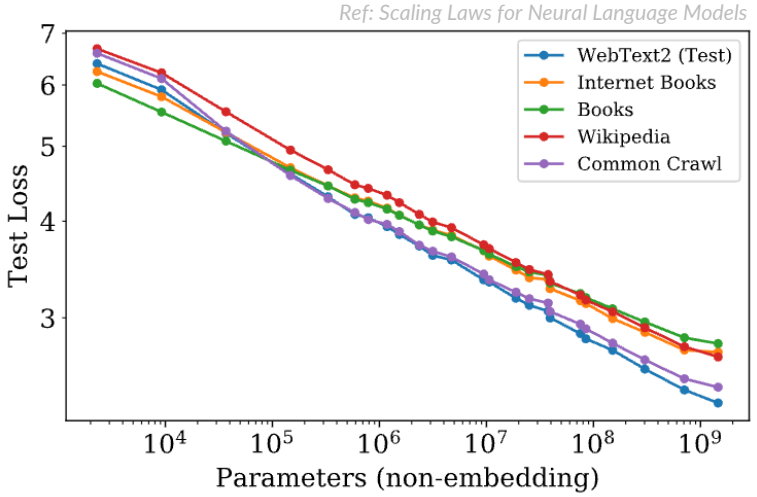
Content Creation

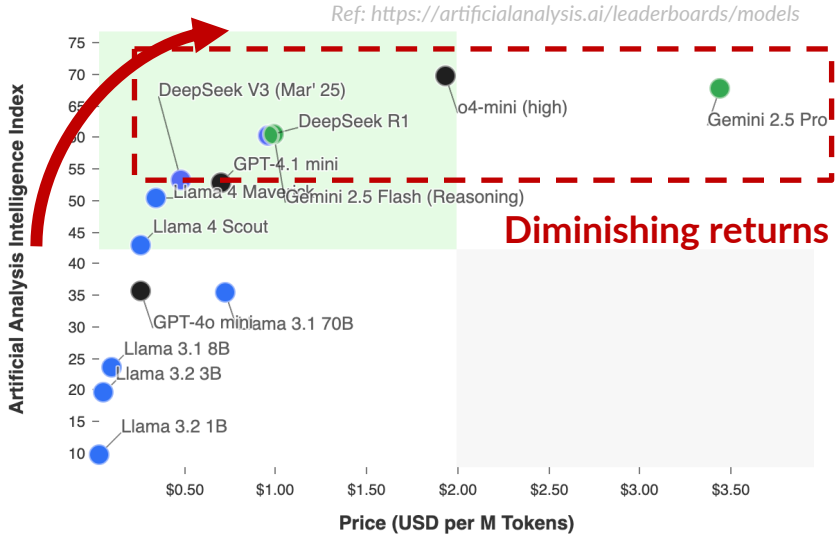
e.g., marketing copywriting, video script writing, personalized email, story creation

Cost management and latency control are critical.

Motivation

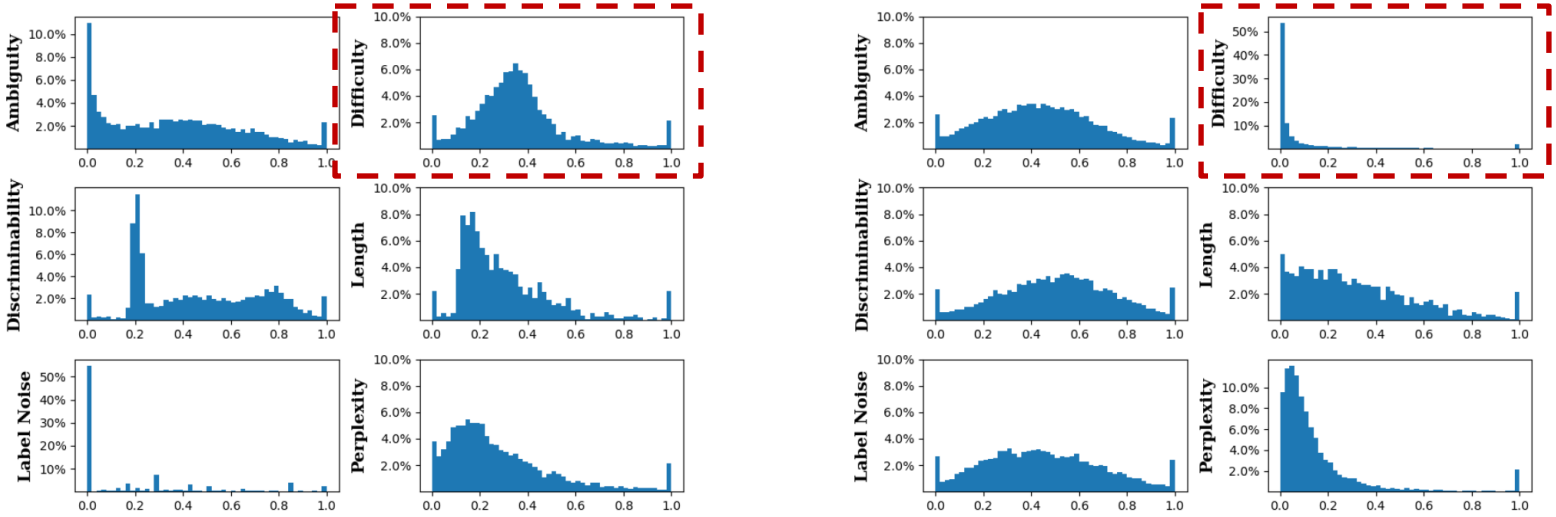
LLMs are advancing with varied **cost-performance** profiles.





Motivation

Complexity distribution of input queries.

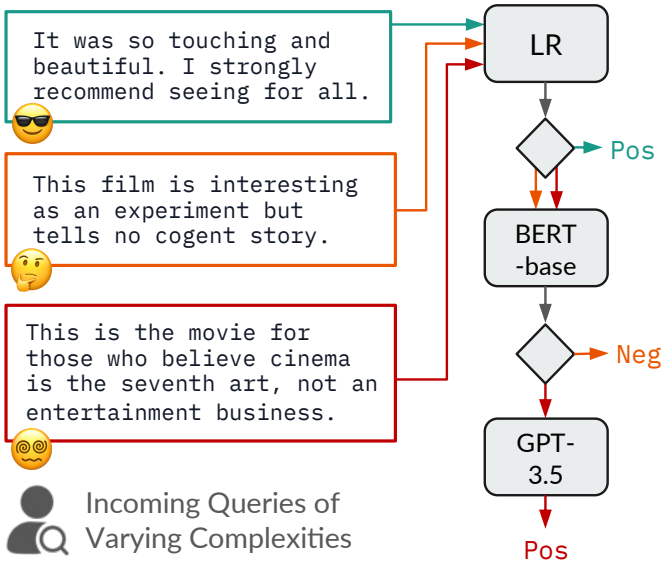


(a) SQUAD

(b) MultiNLI

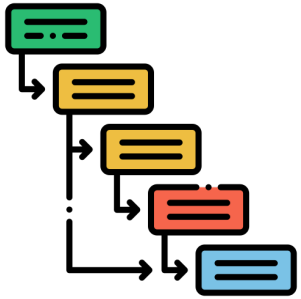
Ref: Benchmark Transparency: Measuring the Impact of Data on Evaluation, NAACL 24

Motivation

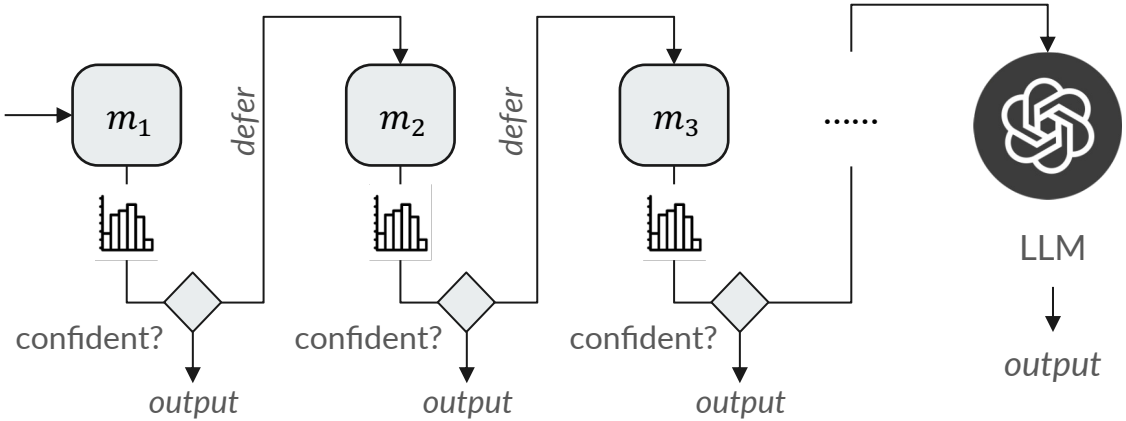


*One can **save resources** by using low-capacity models for simpler tasks and reserving larger models for more complex ones.*

Model Cascade



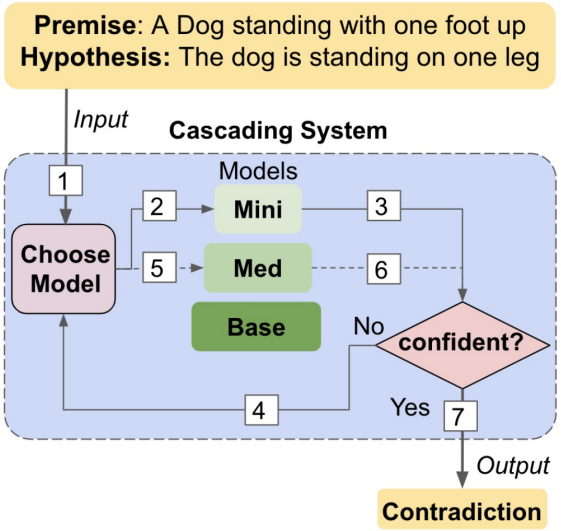
Input Stream



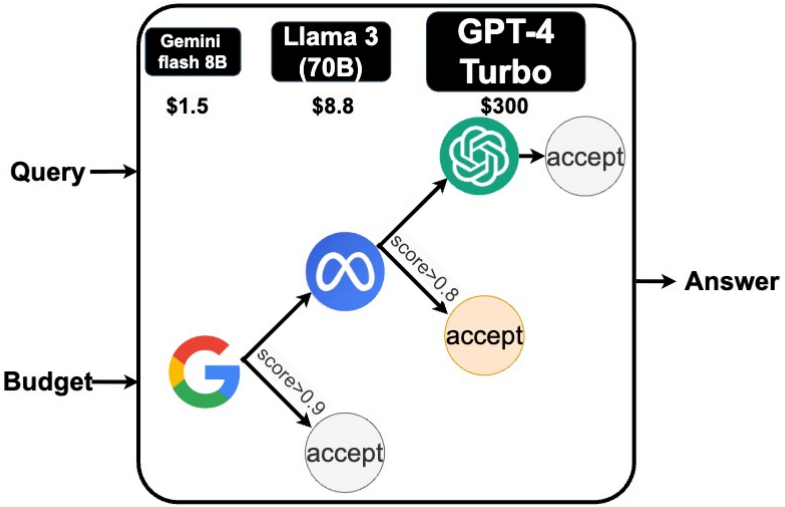
$\text{Cost}(m_1) < \text{Cost}(m_2) < \text{Cost}(m_3) < \ldots < \text{Cost}(\text{LLM})$

Exisiting Works

- Model Cascading

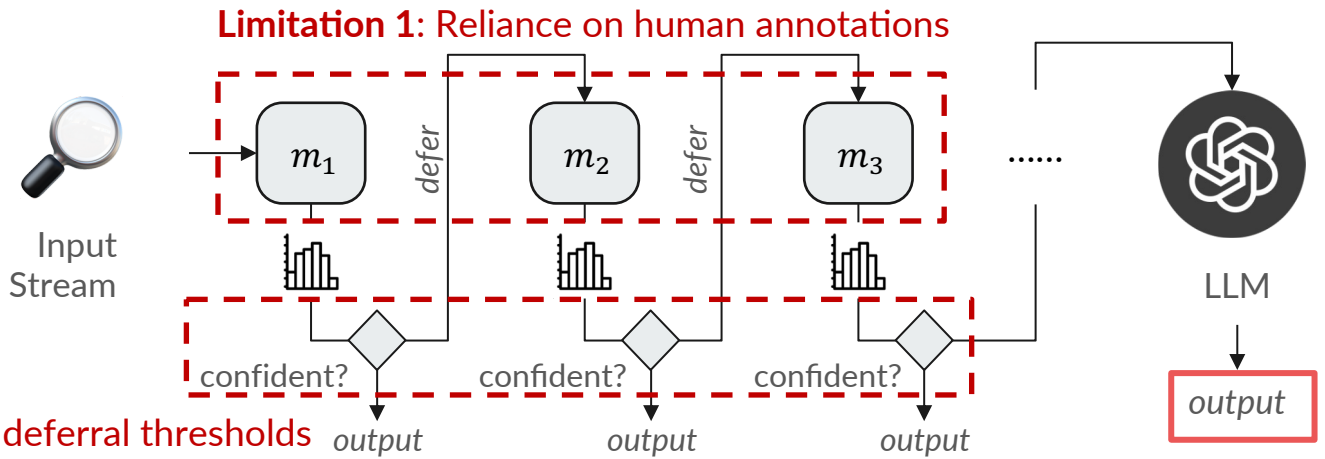


- FrugalGPT



References:
- Model Cascading: Towards Jointly Improving Efficiency and Accuracy of NLP Systems, EMNLP 22
- FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance, TMLR 24

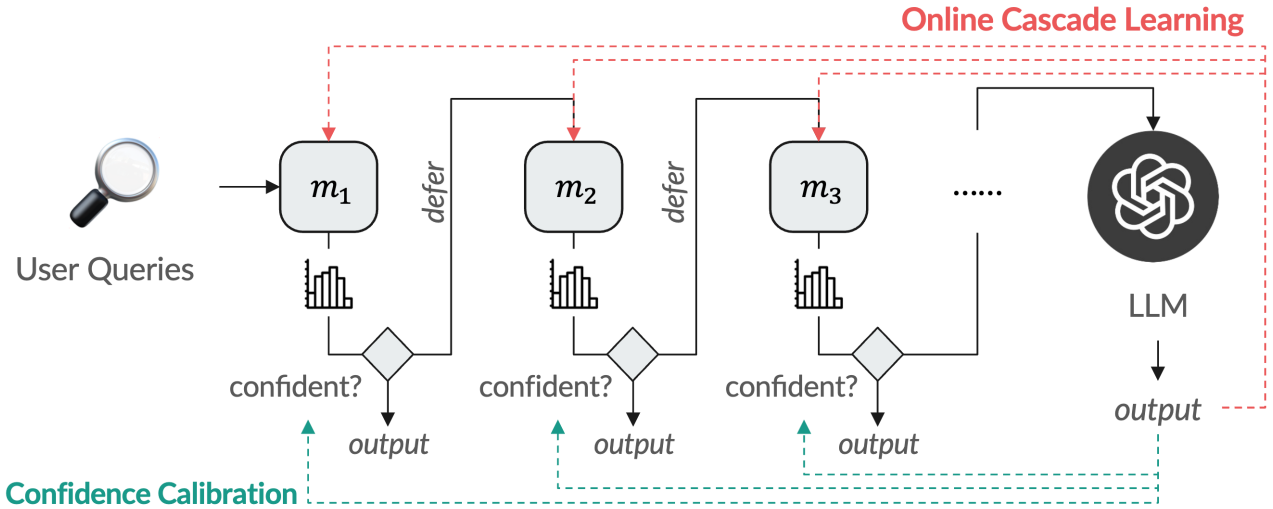
Exisiting Works - Limitations



1. *How to make full use of the LLM outputs?*
2. *How to make accurate deferral decisions?*

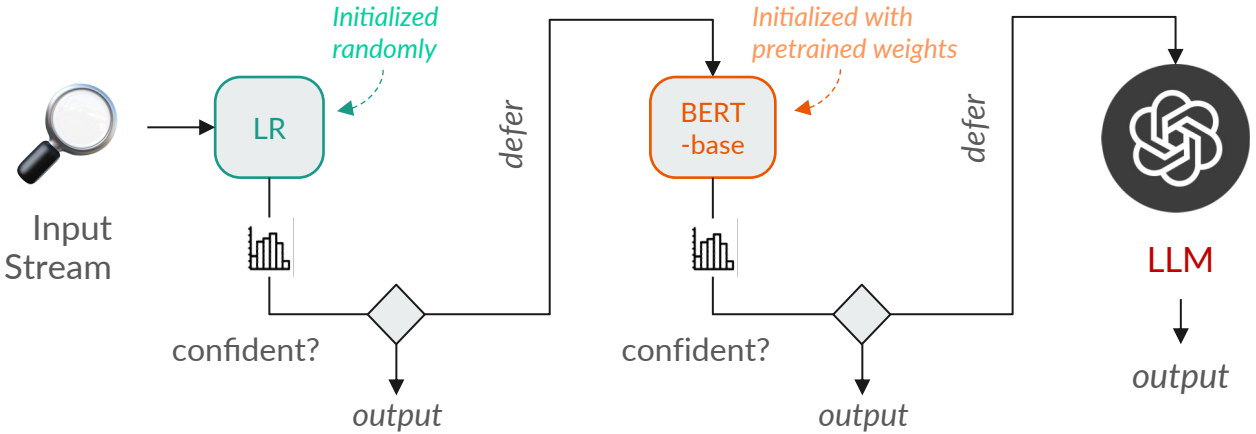
Proposed Method: Online Cascade Learning

Intersecting **Imitation Learning** and **Online Learning**: Smaller models $\langle m_1, \ldots, m_N \rangle$ in the cascade learn from LLM expert's behaviors (red arrows) and calibrate the deferral functions $\langle f_1, \ldots, f_{N-1} \rangle$ in real-time (green arrows).



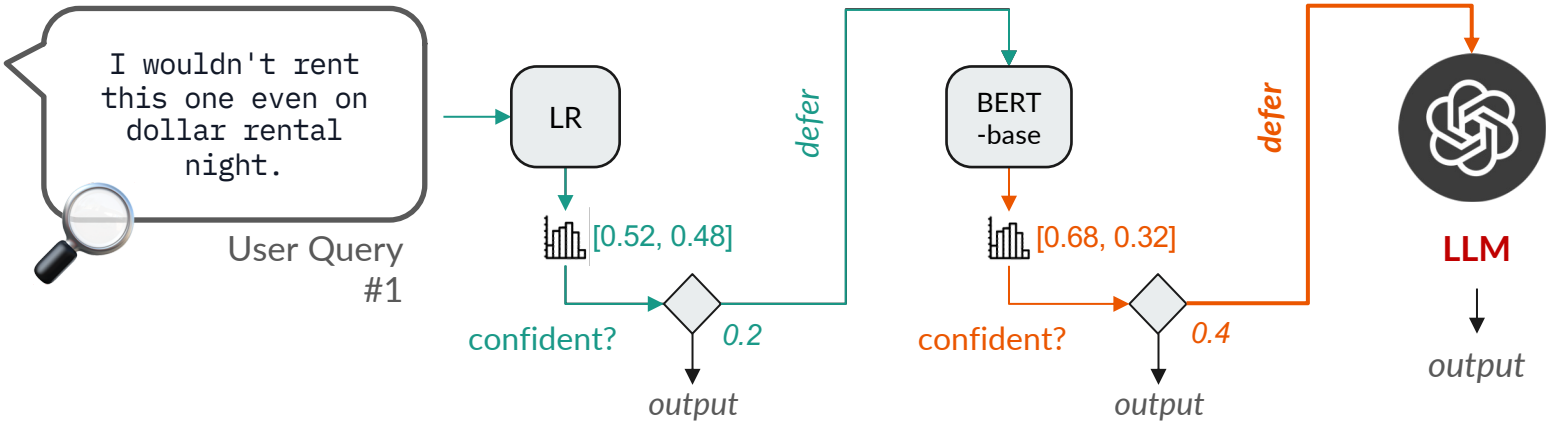
Online Cascade Learning

Assume we have a cascade consisting of a Logistic Regression, a BERT-base, and an LLM:



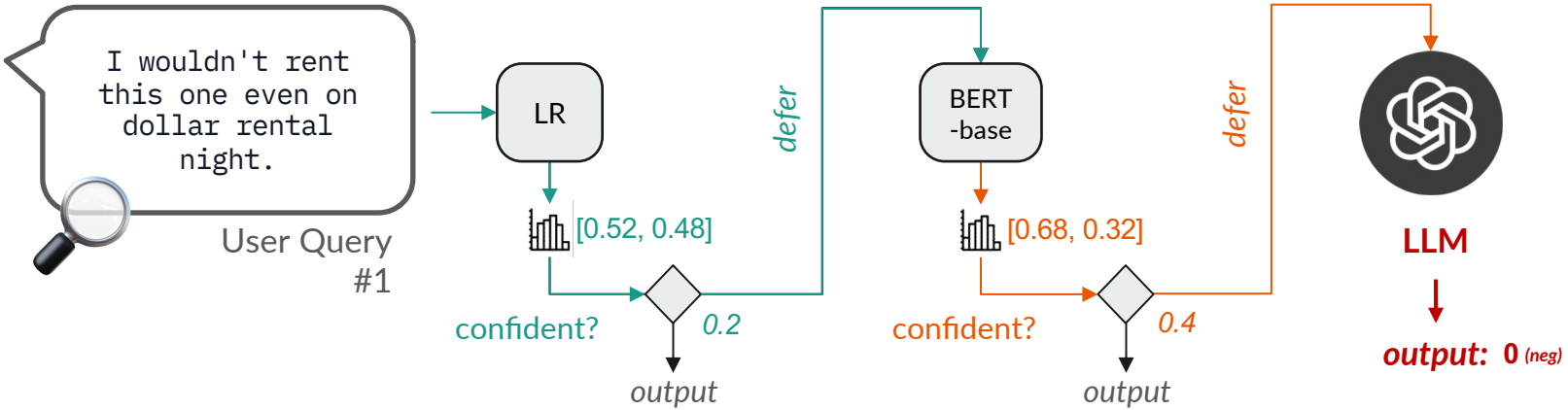
Online Cascade Learning – Warm-up Stage

At startup, the policy keeps its “gates” open, allowing all initial inputs to flow through the cascade



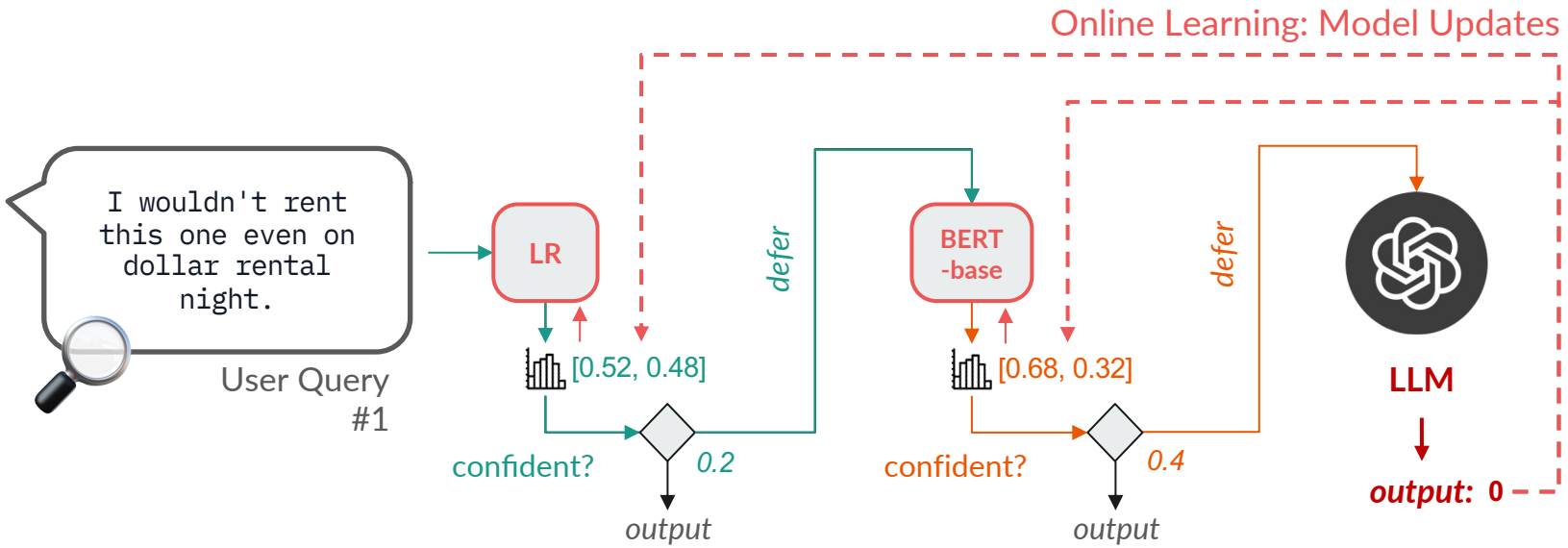
Online Cascade Learning – Warm-up Stage

All initial queries are processed by the most expensive model (an LLM) to collect annotations:



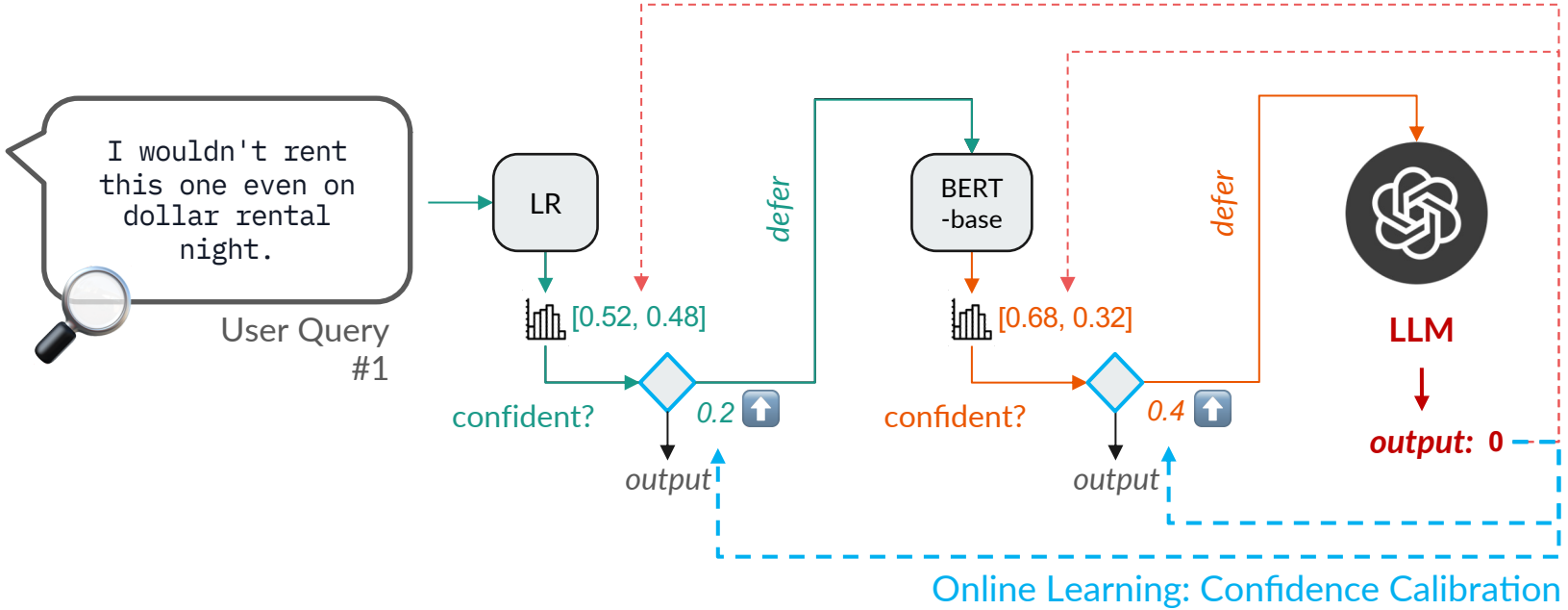
Online Cascade Learning – Model Updates

The smaller models' parameters are updated on the collected annotations.



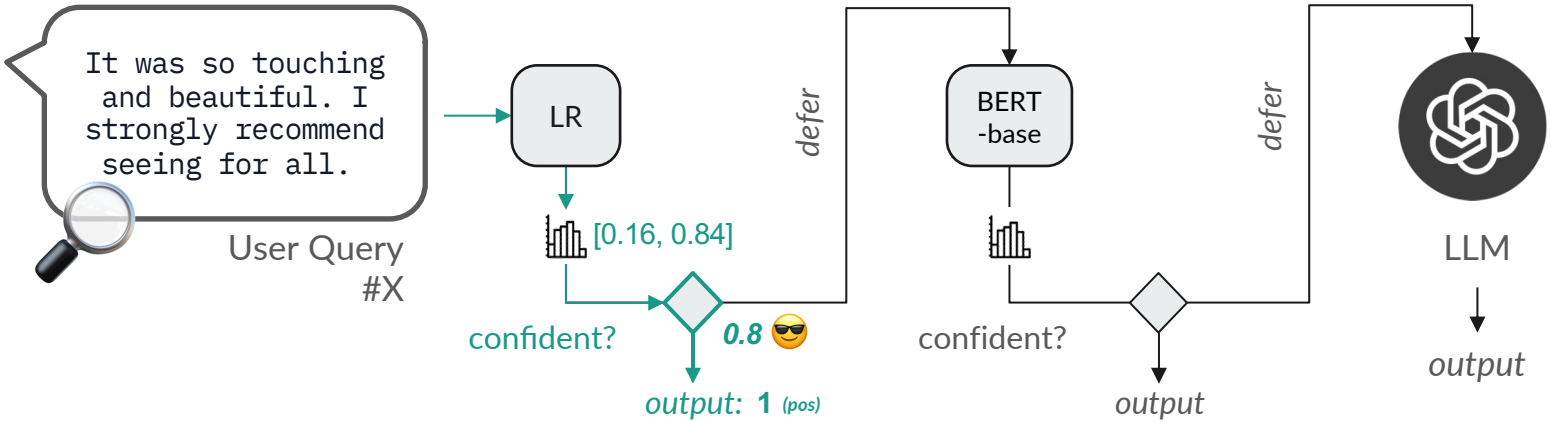
Online Cascade Learning – Confidence Calibration

as well as the deferral functions:



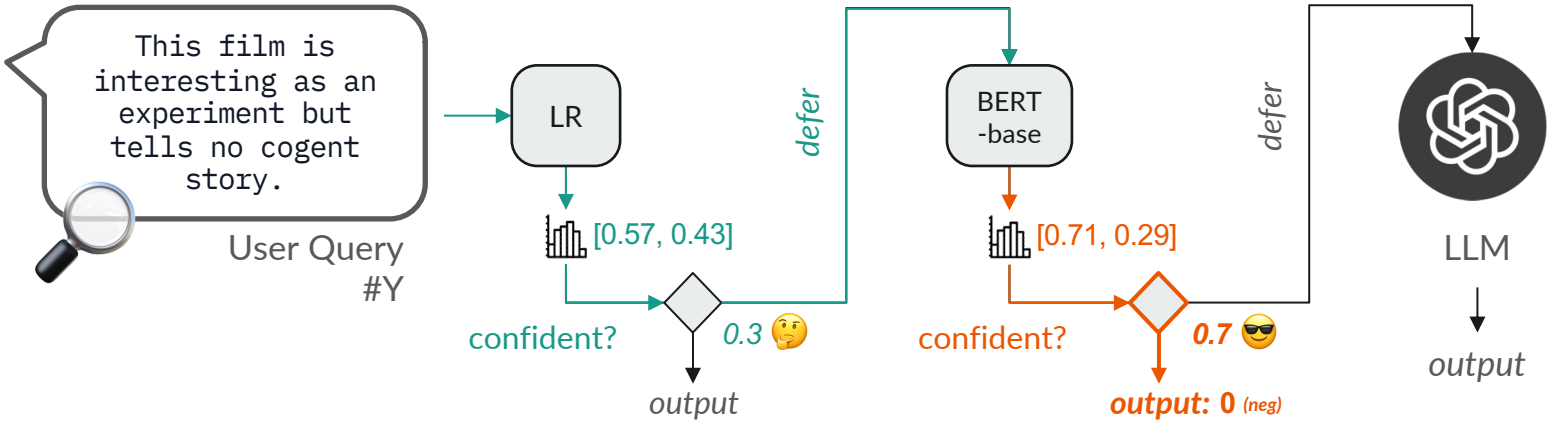
Online Cascade Learning - Stabilized Stage

As the system stabilizes, simple queries can be effectively handled by the smallest model (LR)



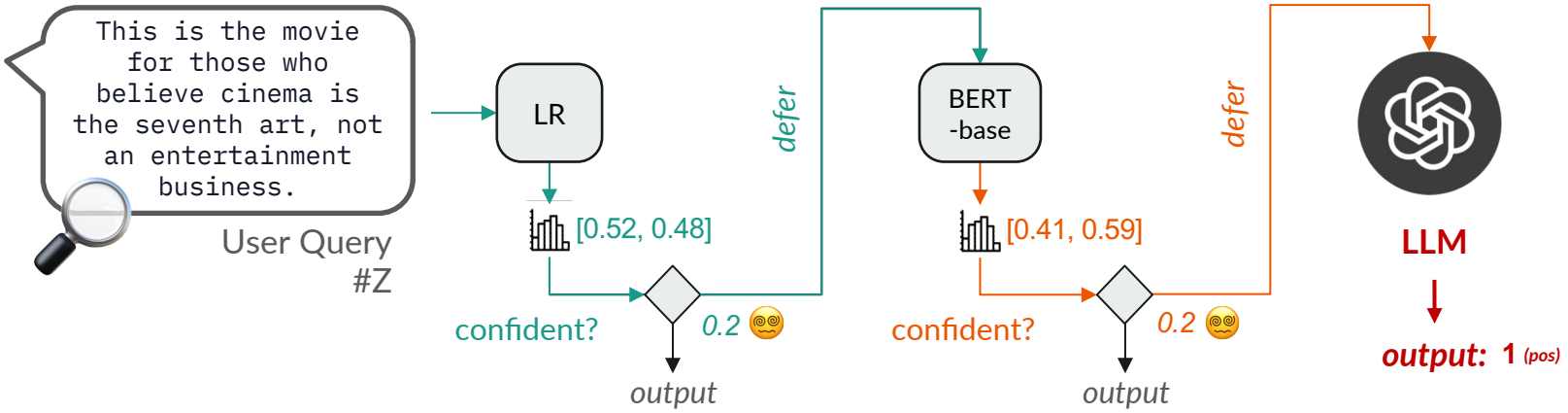
Online Cascade Learning - Stabilized Stage

Harder queries are smartly deferred to the more complex model (BERT-base)



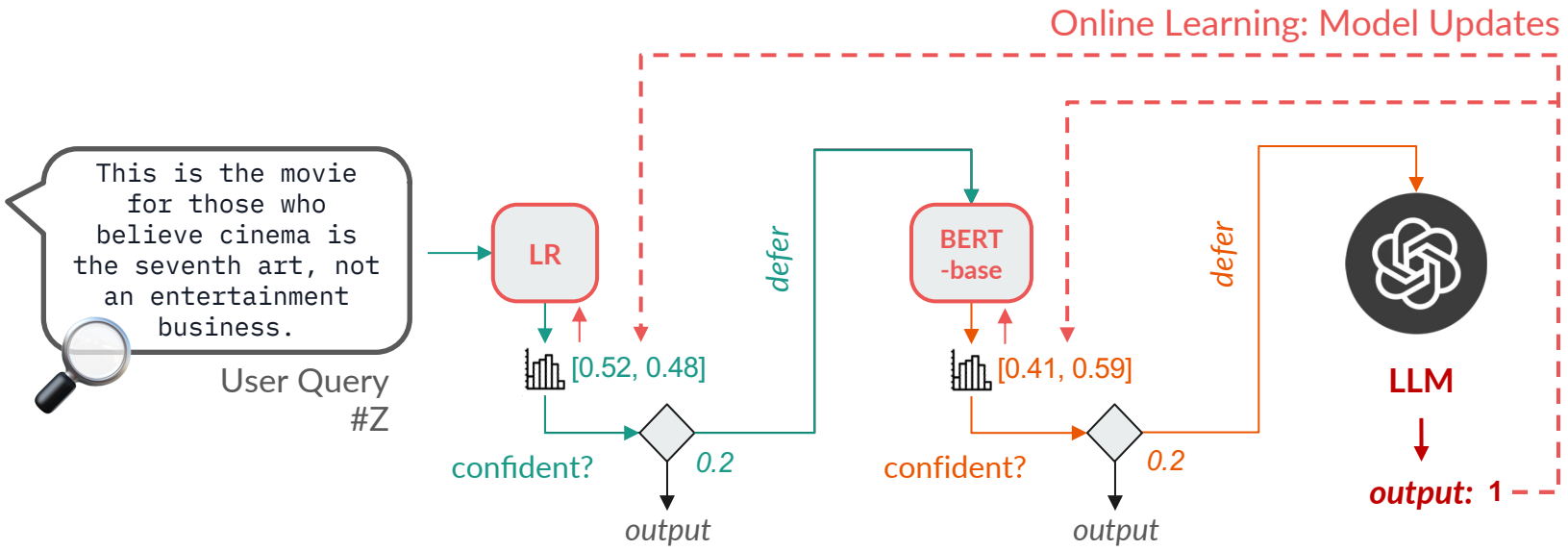
Online Cascade Learning - Stabilized Stage

The most difficult queries are deferred to the LLM to collect annotations for online learning



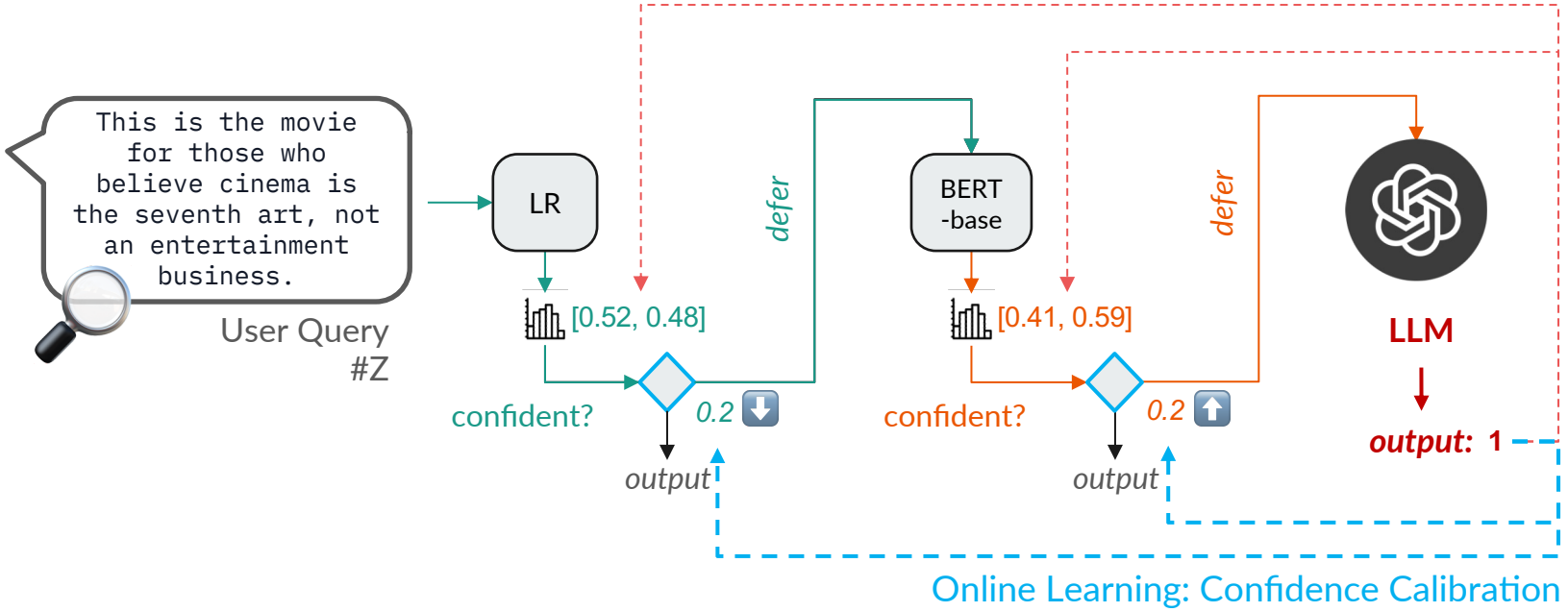
Online Cascade Learning – Model Updates

The smaller models' parameters are updated on the collected annotations.



Online Cascade Learning – Confidence Calibration

as well as the deferral functions:



Problem Formulation – an Episodic MDP

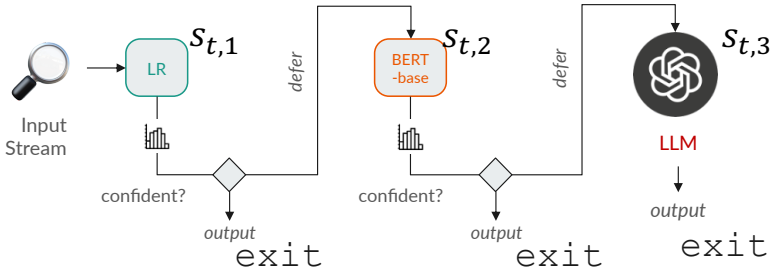
We consider a streaming processing task where input queries are a fixed infinite stream $X = \langle x_1, \ldots, x_t, \ldots \rangle$. Each query x_t is associated with a ground truth y_t from label set Y. Our goal is to predict the label for each x_t using an N-level model cascade cost-effectively.

Now we formulate this as an episodic MDP:

- States (S): Includes states $s_{t,i} = \langle x_t, i \rangle$ where x_t is the user query at time t and i indicates the current cascade level, and a terminal state `exit`.

Let's say, we have 3 models in a cascade for binary classification:

- States: $s_{t,1}$, $s_{t,2}$, $s_{t,3}$, `exit`



Problem Formulation – an Episodic MDP

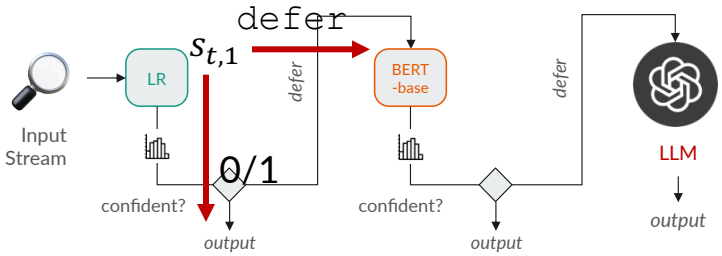
We consider a streaming processing task where input queries are a fixed infinite stream $X = \langle x_1, \dots, x_t, \dots \rangle$. Each query x_t is associated with a ground truth y_t from label set Y. Our goal is to predict the label for each x_t using an N-level model cascade cost-effectively.

Now we formulate this as an episodic MDP:

- Actions (A): Consists of label set Y, representing the potential predictions if the cascade choose to output at the current state, and a special action `defer` that activates the next level of the cascade.

Let's say, we have 3 models in a cascade for binary classification:

- Actions: $Y = \{0,1\}$, `defer`



Problem Formulation – an Episodic MDP

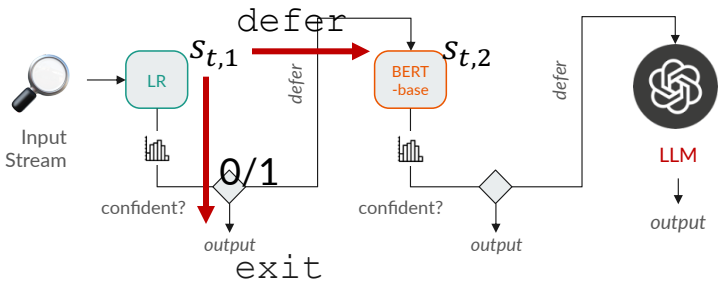
We consider a streaming processing task where input queries are a fixed infinite stream $X = \langle x_1, \ldots, x_t, \ldots \rangle$. Each query x_t is associated with a ground truth y_t from label set Y. Our goal is to predict the label for each x_t using an N-level model cascade cost-effectively.

Now we formulate this as an episodic MDP:

- Transitions ($\mathcal{T}$): A determinisitic transition function, consisting of transitions of the form:
  - $\mathcal{T}(s_{t,i}, a) = \texttt{exit}$ for $a \in Y$
  - $\mathcal{T}(s_{t,i}, \texttt{defer}) = s_{t,i+1}$

Let's say, we have 3 models in a cascade for binary classification, and we are now at $s_{t,1}$:

- Transitions:
  - $\mathcal{T}(s_{t,1}, a) = \texttt{exit}$ if $a \in \{0,1\}$
  - $\mathcal{T}(s_{t,1}, \texttt{defer}) = s_{t,2}$



Problem Formulation – an Episodic MDP

We consider a streaming processing task where input queries are a fixed infinite stream $X = \langle x_1, \ldots, x_t, \ldots \rangle$. Each query x_t is associated with a ground truth y_t from label set Y. Our goal is to predict the label for each x_t using an N-level model cascade cost-effectively.

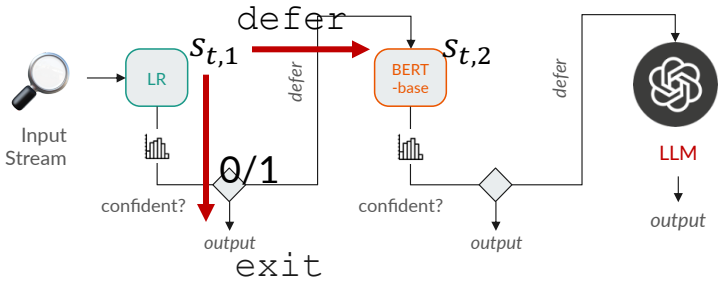
Now we formulate this as an episodic MDP:

- Cost Function (C): Balances predictive loss ($\mathcal{L}$) for prediction actions and computational overheads of next model (c_{i+1}) for deferral actions. Factor μ adjusts the trade-off between computational **cost** and **accuracy**.

$$C(s_{t,i}, a) = \begin{cases} \mathcal{L}(a|y_t) & \text{if } a \in Y, \\ \mu c_{i+1} & \text{if } a = \texttt{defer}. \end{cases}$$

Let's say, we have 3 models in a cascade for binary classification, and we are now at $s_{t,1}$:

- Cost at $s_{t,1}$:
  - $C(s_{t,1}, a) = \mathcal{L}(a|y_t)$ if $a \in \{0,1\}$
  - $C(s_{t,1}, a) = \mu c_2$ if $a = \texttt{defer}$, where c_2 represents the inference costs of BERT-base



Method – Policies and Learning Objective

Policy Representation: We represent policies in a factorized way using a set of *classification models* $\langle m_1, \ldots, m_N \rangle$ that constitute the different levels of the cascade, and *deferral functions* $\langle f_1, \ldots, f_{N-1} \rangle$ that decide deferral.

Probability of deferring: $\pi(s_{t,i}, \texttt{defer}) = f_i(m_i(x_t))$

Probability of output label y: $\pi(s_{t,i}, y) = \left(1 - f_i(m_i(x_t))\right) m_i(x_t)[y]$ for $y \in Y$

Objective function: Minimize the combined cost of **computational costs** and **prediction loss.**

$$J(\pi, T) = \sum_{t=1}^{T} \left[\sum_{i=1}^{N} p_\pi^{s_{t,i}} C_\pi(s_{t,i}) \right]$$

$$C_\pi(s_{t,i}) = \pi(s_{t,i}, \texttt{defer}) \cdot \mu c_{i+1} + (1 - \pi(s_{t,i}, \texttt{defer})) \cdot \sum_{y \in Y} \pi(s_{t,i}, y) \cdot \mathcal{L}(y | y_t).$$

probability of reaching state $s_{t,i}$

$$p_\pi^{s_{t,i}} = \prod_{j=1}^{i-1} \pi(s_{t,j}, \texttt{defer})$$

Experiments

- **Models:**
  - Logistic Regression
  - BERT-base, ~110 M (and BERT-large, ~340M for N=4)
  - GPT-3.5 Turbo / Llama 2-70b-chat
- **Benchmarks:**
  - **IMDB**: a binary sentiment classification benchmark with 50,000 movie reviews
  - **HateSpeech**: class-imbalanced (1:7.95) hate speech detection with 10,703 samples
  - **ISEAR**: multi-class emotion detection benchmark with 7,666 samples in 7 categories
  - **FEVER**: a fact-checking dataset with 6,512 claims requiring complex reasoning and information verification.
- **Baselines**: Single LLMs, Knowledge Distillation, Online Ensemble Learning

Experimental Results

	IMDB			HateSpeech (Accuracy | Recall)						ISEAR			FEVER		
	$\mathcal{N}$=1300	$\mathcal{N}$=3800	$\mathcal{N}$=5200	$\mathcal{N}$=600		$\mathcal{N}$=2700		$\mathcal{N}$=4900		$\mathcal{N}$=1200	$\mathcal{N}$=1500	$\mathcal{N}$=2700	$\mathcal{N}$=700	$\mathcal{N}$=2000	$\mathcal{N}$=2800
GPT-3.5 Turbo		94.15			83.34		83.28				70.34			79.98	
Distilled LR	82.61	83.60	87.01	80.18	37.94	82.23	49.25	**85.03**	45.59	44.97	47.46	48.92	56.51	57.80	57.13
Distilled BERT-base	85.28	90.18	90.19	80.49	64.39	80.71	73.88	79.35	77.37	**61.49**	62.62	63.37	61.70	63.64	70.82
Online Ensemble Learning	86.73	88.80	89.95	82.61	76.75	77.48	76.89	81.55	80.30	56.56	60.42	61.78	61.69	69.78	76.67
Online Cascade Learning	**87.95**	**92.48**	**93.01**	**82.66**	**82.36**	**85.35**	**77.20**	83.26	**81.03**	60.78	**65.34**	**69.75**	**61.95**	**71.86**	**78.49**
Llama 2 70B Chat		93.33			77.81		82.19				68.23			77.15	
Distilled LR	82.17	85.80	86.88	67.94	66.56	**79.71**	61.73	**81.46**	49.91	46.78	47.56	51.76	57.46	61.24	58.42
Distilled BERT-base	85.39	85.59	85.44	75.84	**78.87**	79.18	75.54	80.27	72.21	**62.18**	61.84	65.12	**65.88**	65.66	67.54
Online Ensemble Learning	87.14	88.66	89.61	75.99	60.36	70.79	**79.16**	76.82	81.84	54.74	57.35	60.19	63.48	71.27	76.46
Online Cascade Learning	**87.58**	**92.14**	**92.63**	**78.30**	63.06	78.32	76.54	78.32	**82.03**	59.24	**63.34**	**67.25**	63.81	**72.47**	**77.73**

Table: Comparison of accuracy (and recall for HateSpeech dataset) among different methods under various cost budgets (i.e., the maximum allowable LLM calls, denoted as N)

1. *Overall, Online Cascade Learning can achieve better **cost-performance trade-off**.*

Experimental Results

	IMDB			HateSpeech (Accuracy | Recall)						ISEAR			FEVER		
	$\mathcal{N}$=1300	$\mathcal{N}$=3800	$\mathcal{N}$=5200	$\mathcal{N}$=600		$\mathcal{N}$=2700		$\mathcal{N}$=4900		$\mathcal{N}$=1200	$\mathcal{N}$=1500	$\mathcal{N}$=2700	$\mathcal{N}$=700	$\mathcal{N}$=2000	$\mathcal{N}$=2800
GPT-3.5 Turbo		94.15			83.34		83.28				70.34			79.98	
Distilled LR	82.61	83.60	87.01	80.18	37.94	82.23	49.25	**85.03**	45.59	44.97	47.46	48.92	56.51	57.80	57.13
Distilled BERT-base	85.28	90.18	90.19	80.49	64.39	80.71	73.88	79.35	77.37	**61.49**	62.62	63.37	61.70	63.64	70.82
Online Ensemble Learning	86.73	88.80	89.95	82.61	76.75	77.48	76.89	81.55	80.30	56.56	60.42	61.78	61.69	69.78	76.67
Online Cascade Learning	**87.95**	**92.48**	**93.01**	**82.66**	**82.36**	**85.35**	**77.20**	83.26	**81.03**	60.78	**65.34**	**69.75**	**61.95**	**71.86**	**78.49**
Llama 2 70B Chat		93.33			77.81		82.19				68.23			77.15	
Distilled LR	82.17	85.80	86.88	67.94	66.56	**79.71**	61.73	**81.46**	49.91	46.78	47.56	51.76	57.46	61.24	58.42
Distilled BERT-base	85.39	85.59	85.44	75.84	**78.87**	79.18	75.54	80.27	72.21	**62.18**	61.84	65.12	**65.88**	65.66	67.54
Online Ensemble Learning	87.14	88.66	89.61	75.99	60.36	70.79	**79.16**	76.82	81.84	54.74	57.35	60.19	63.48	71.27	76.46
Online Cascade Learning	**87.58**	**92.14**	**92.63**	**78.30**	63.06	78.32	76.54	78.32	**82.03**	59.24	**63.34**	**67.25**	63.81	**72.47**	**77.73**

Table: Comparison of accuracy (and recall for HateSpeech dataset) among different methods under various cost budgets.

2. *Model cascade can even **outperform LLM** by taking advantage of fine-tuned models.*

Experimental Results

	IMDB			HateSpeech (Accuracy | Recall)						ISEAR			FEVER		
	$\mathcal{N}$=1300	$\mathcal{N}$=3800	$\mathcal{N}$=5200	$\mathcal{N}$=600		$\mathcal{N}$=2700		$\mathcal{N}$=4900		$\mathcal{N}$=1200	$\mathcal{N}$=1500	$\mathcal{N}$=2700	$\mathcal{N}$=700	$\mathcal{N}$=2000	$\mathcal{N}$=2800
GPT-3.5 Turbo		94.15			83.34		83.28				70.34			79.98	
Distilled LR	82.61	83.60	87.01	80.18	37.94	82.23	49.25	**85.03**	45.59	44.97	47.46	48.92	56.51	57.80	57.13
Distilled BERT-base	85.28	90.18	90.19	80.49	64.39	80.71	73.88	79.35	77.37	**61.49**	62.62	63.37	61.70	63.64	70.82
Online Ensemble Learning	86.73	88.80	89.95	82.61	76.75	77.48	76.89	81.55	80.30	56.56	60.42	61.78	61.69	69.78	76.67
Online Cascade Learning	**87.95**	**92.48**	**93.01**	**82.66**	**82.36**	**85.35**	**77.20**	83.26	**81.03**	60.78	**65.34**	**69.75**	**61.95**	**71.86**	**78.49**
Llama 2 70B Chat		93.33			77.81		82.19				68.23			77.15	
Distilled LR	82.17	85.80	86.88	67.94	66.56	**79.71**	61.73	**81.46**	49.91	46.78	47.56	51.76	57.46	61.24	58.42
Distilled BERT-base	85.39	85.59	85.44	75.84	**78.87**	79.18	75.54	80.27	72.21	**62.18**	61.84	65.12	**65.88**	65.66	67.54
Online Ensemble Learning	87.14	88.66	89.61	75.99	60.36	70.79	**79.16**	76.82	81.84	54.74	57.35	60.19	63.48	71.27	76.46
Online Cascade Learning	**87.58**	**92.14**	**92.63**	**78.30**	63.06	78.32	76.54	78.32	**82.03**	59.24	**63.34**	**67.25**	63.81	**72.47**	**77.73**

Table: Comparison of accuracy (and recall for HateSpeech dataset) among different methods under various cost budgets.

3. As cost buget increases, distilled models may be bounded by their capacity, whereas model cascade can always defer to LLMs for complex queries.

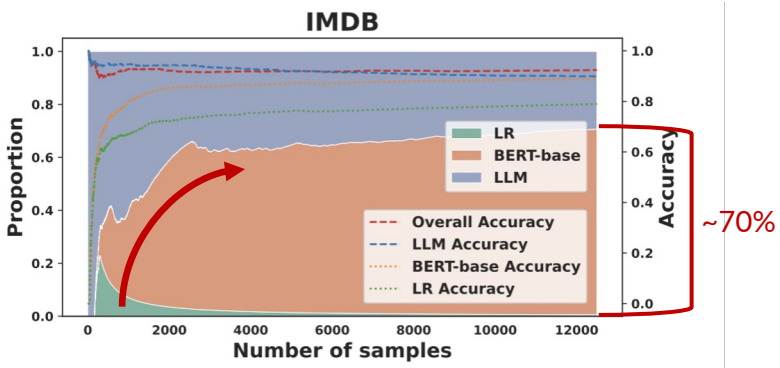


Figure 5: Inference results on IMDB when $\mathcal{N} = 3671$. Online cascade learning system performs similarly to GPT-3.5 Turbo while saving ∼70% of the inference costs.

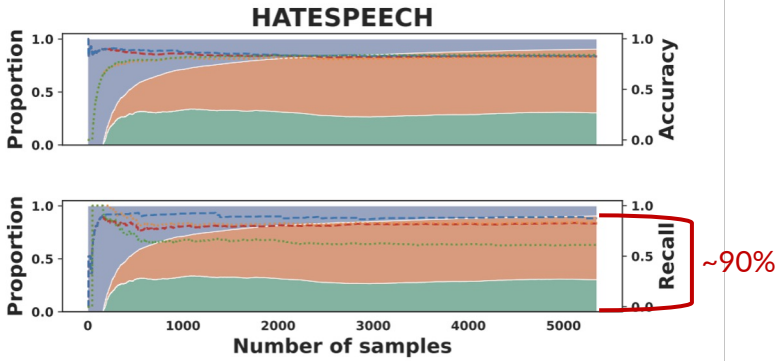


Figure 6: Inference results on HateSpeech when $\mathcal{N} = 507$. Online cascade learning system performs similarly to GPT-3.5 Turbo while saving ∼90% of the inference costs.

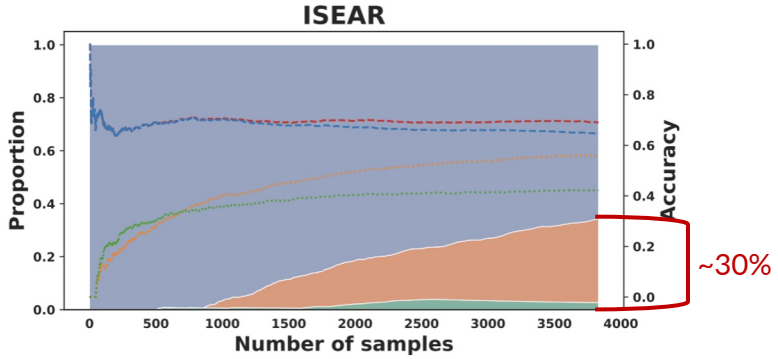


Figure 7: Inference results on ISEAR when $\mathcal{N} = 2517$. Online cascade learning system performs very close to GPT-3.5 Turbo while saving ∼30% of the inference costs.

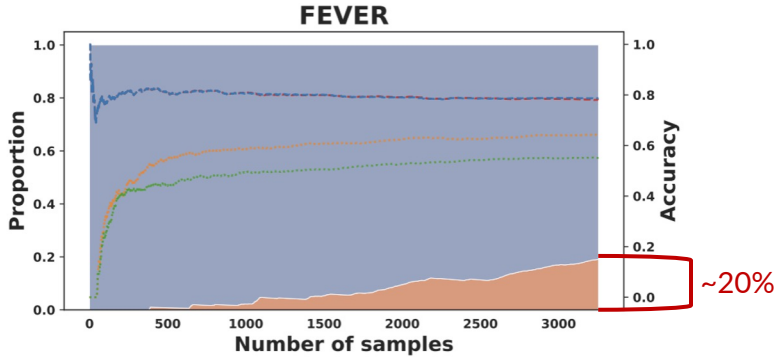


Figure 8: Inference results on FEVER when $\mathcal{N} = 2635$. Online cascade learning system performs similarly to GPT-3.5 Turbo while saving ∼20% of the inference costs.

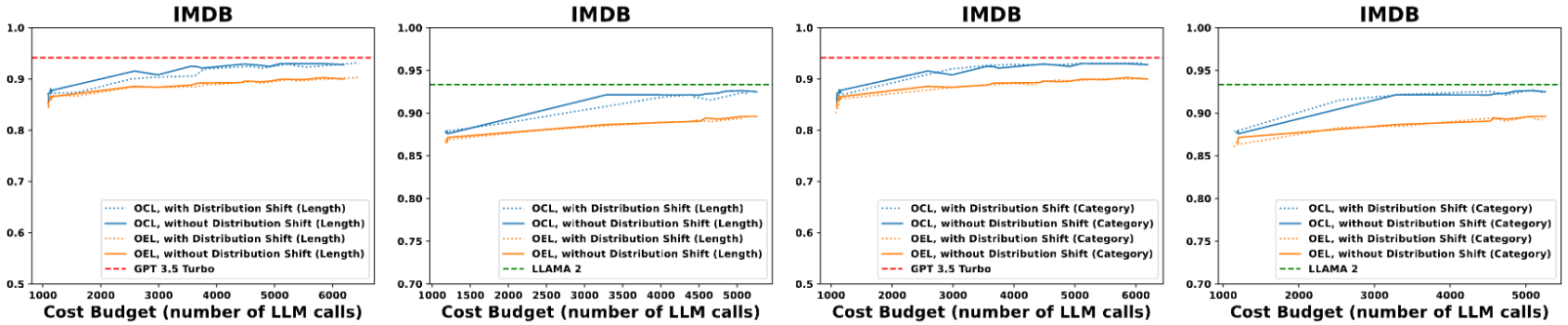
Robustness againt Input Distribution Shift

a. Distribution Shift in Input Length.

- IMDB benchmark rearranged in length ascending order, simulating a distribution shift over the **input complexity**.

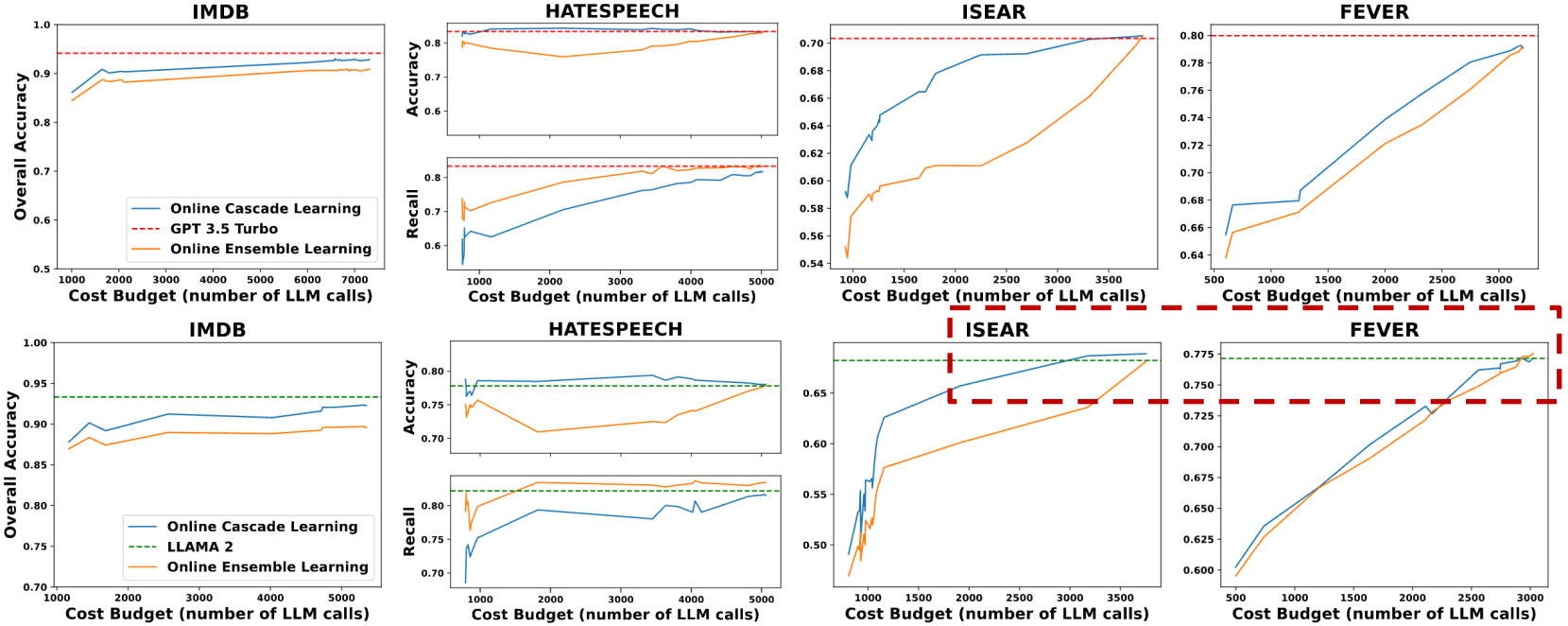
b. Distribution Shift in Input Category.

- IMDB benchmark rearranged by filtering all reviews regarding “Comedy” movies, then feeding them as the last part of the input stream, simulating a distribution shift over the **input semantics**.



Our method can quickly adapt to unseen inputs, perform robustly against distribution shifts in the data streams.

Adaptability to Larger Cascade ($N = 4$)

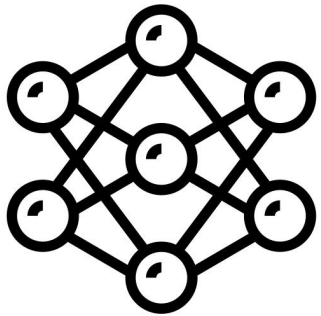


Our method can flexibly accommodate and efficiently scale up a larger cascade with more models.

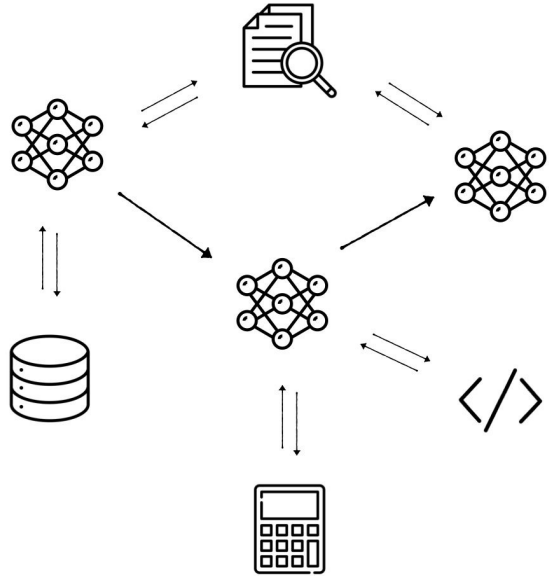
Key Findings

- **Cost-Efficiency**: Significant cost savings (up to 90%) with an accuracy comparable to LLMs alone.
- **Adaptability**: Smartly adapts across tasks of different difficulties and scaled-up cascade setups.
- **Robustness**: Utilizing the advantages of online learning, quickly adapts to unseen inputs, and performs robustly against distribution shifts in the data streams.

Future Work



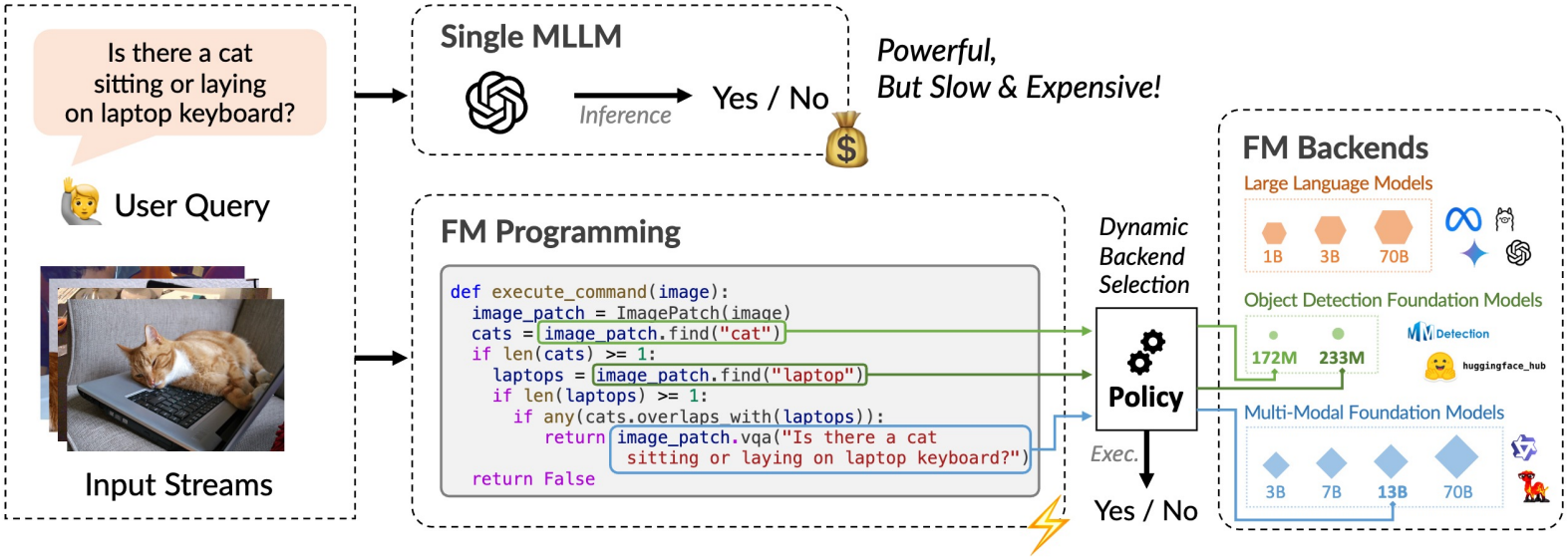
Single Model



Compound AI System
(e.g., Foundation Model Agent with tool use)

Future Work

Cost-efficient Inference with Foundation Model Programs



By exploiting **task structure** and tailoring submodules to each **input's complexity**, *Foundation Model Programming* can trace an optimal tradeoff curve on the Pareto frontier of resource usage v.s. performance.

Thank you!

Q & A

Project Site

Paper

Framework

Nie, Lunyiu, Zhimin Ding, Erdong Hu, Christopher Jermaine, and Swarat Chaudhuri.
"Online Cascade Learning for Efficient Inference over Streams."
In International Conference on Machine Learning, pp. 38071-38090. PMLR, 2024.