

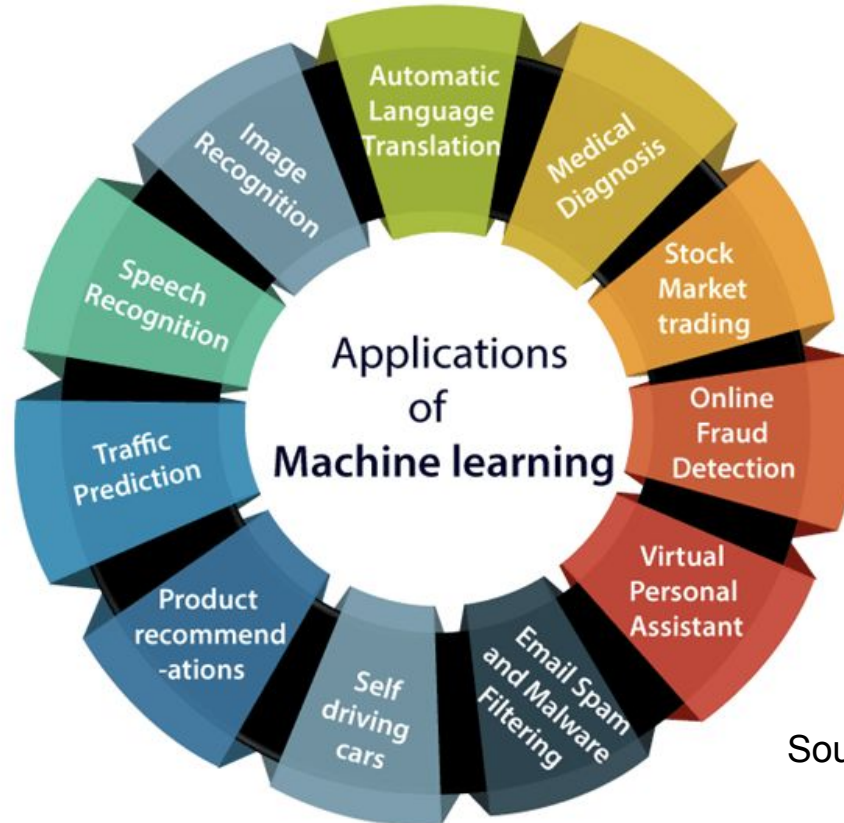
# Optimisation of Inference Queries

Ziyu Li, TU Delft

*with Marios Fragkoulis, Alessandro Bozzon, Asterios Katsifodimos*

01.10.2021, DSDSD

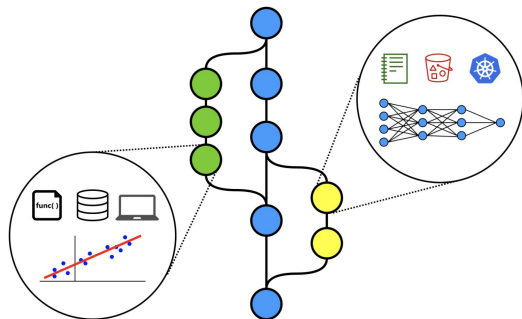
# Machine Learning is widely adopted



Source: Javapoint

# Management for explosion of ML models

## MLOps



ModelDB [2]

## Model Hub



HuggingFace [3]  
17,144

TensorFlow  
Hub

[4]  
>1000

PYTORCH  
HUB

[5]  
42+

[2] Vartak M, Subramanyam H, Lee W E, et al. ModelDB: a system for machine learning model management[C]//Proceedings of the Workshop on Human-In-the-Loop Data Analytics. 2016: 1-3.

[3] HuggingFace <https://huggingface.co/models>

[4] TensorFlow Hub <https://tfhub.dev/>

[5] PyTorch Hub <https://pytorch.org/hub/>

# Example: HuggingFace

- Model description
- Intended uses & limitations
- Training procedure & data
- Evaluation results (accuracy)

## Limitations

- Speed also matters
  - Lack of information regarding inference cost, e.g., FLOPs, execution time
- Lack of necessary metadata
  - Input & output
  - Performance across object classes



Models 123

Search Models

google/vit-base-patch16-224

Image Classification • Updated 17 days ago • 126k • 4

facebook/deit-base-distilled-patch16-224

Image Classification • Updated Apr 9 • 4.95k

microsoft/beit-base-patch16-224-pt22k-ft22k

Image Classification • Updated 17 days ago • 4.34k • 1

microsoft/beit-base-patch16-224

Image Classification • Updated 17 days ago • 3.03k

google/vit-large-patch16-224

Image Classification • Updated Jun 10 • 2.54k

microsoft/beit-base-patch16-224-pt22k

Image Classification • Updated 17 days ago • 1.98k

google/vit-base-patch16-384

Image Classification • Updated Jun 10 • 876

google/vit-base-patch16-224

like 4

Image Classification • PyTorch • JAX • Transformers • imagenet • imagenet-21k • arxiv:2010.11929

Model card

Files and versions

### Vision Transformer (base-sized model)

Vision Transformer (ViT) model pre-trained on ImageNet-21k (14 million images, 21,843 classes) at resolution 224x224, and fine-tuned on ImageNet 2012 (1 million images, 1,000 classes) at resolution 224x224. It was introduced in the paper [An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale](#) by Dosovitskiy et al. and first released in [this repository](#). However, the weights were converted from the [timm repository](#) by Ross Wightman, who already converted the weights from JAX to PyTorch. Credits go to him.

Disclaimer: The team releasing ViT did not write a model card for this model so this model card has been written by the Hugging Face team.

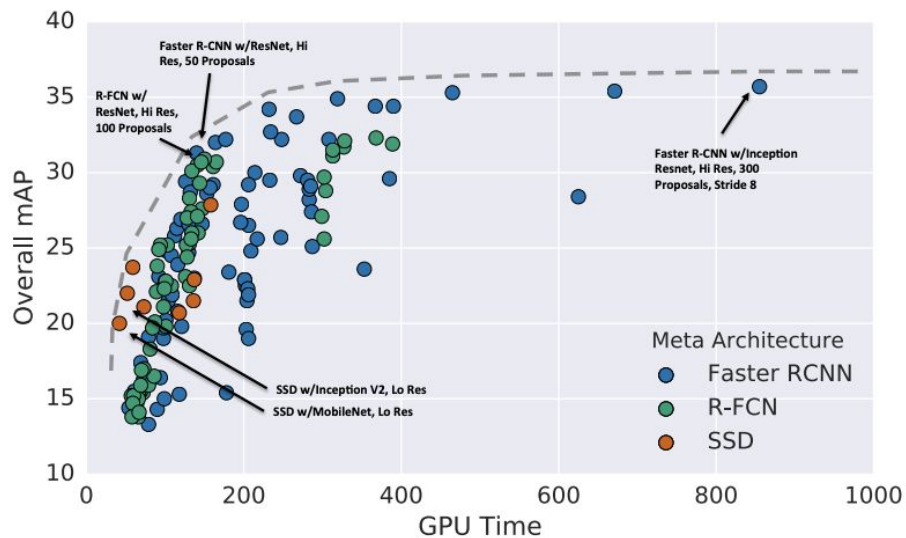
### Model description

The Vision Transformer (ViT) is a transformer encoder model (BERT-like) pretrained on a large collection of images in a supervised fashion, namely ImageNet-21k, at a resolution of 224x224 pixels. Next, the model was fine-tuned on ImageNet (also referred to as ILSVRC2012), a dataset comprising 1 million images and 1,000 classes, also at resolution 224x224.

Images are presented to the model as a sequence of fixed-size patches (resolution 16x16), which are linearly embedded. One also adds a [CLS] token to the beginning of a sequence to use it for classification tasks. One also adds absolute position embeddings before feeding the sequence to the layers of the Transformer encoder.

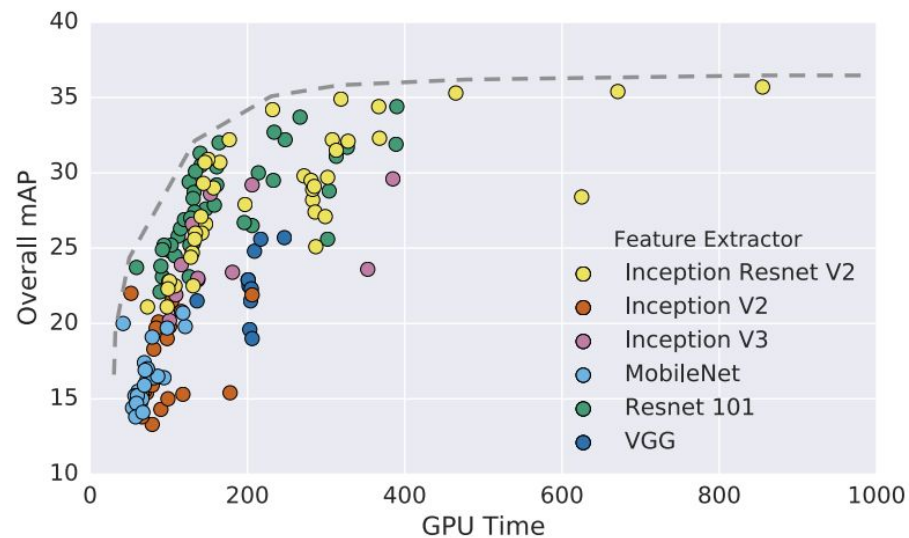
By pre-training the model, it learns an inner representation of images that can then be used to extract features useful for downstream tasks: if you have a dataset of labeled images for instance, you can train a standard classifier by placing a linear layer on top of the pre-trained encoder. One typically places a linear layer on top of the [CLS] token, as the last hidden state of this token can be seen as a

# Tradeoff between accuracy and execution time



(a)

source: [2]

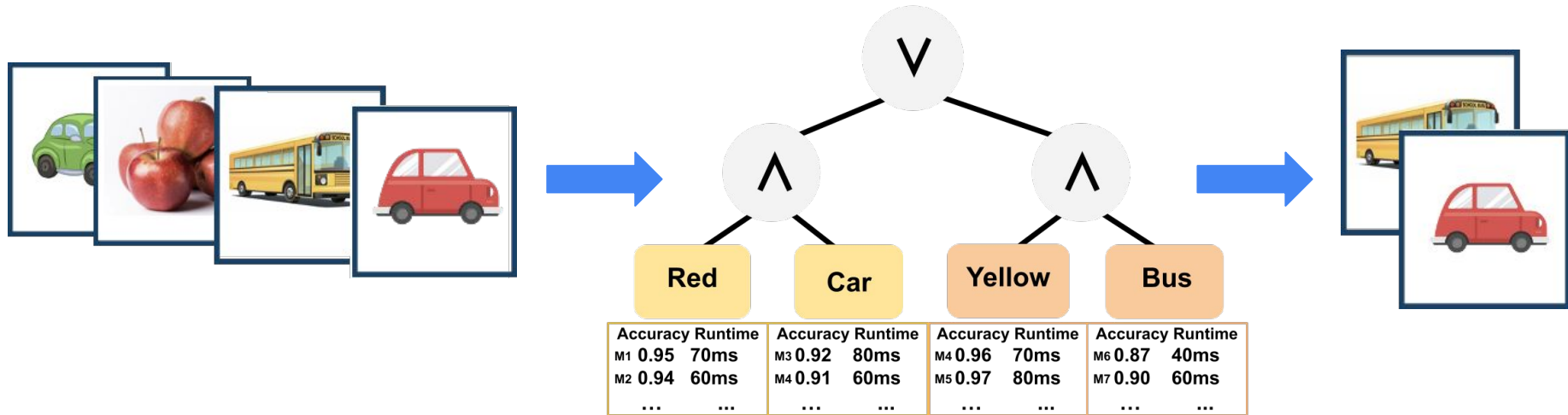


(b)

# Increasingly complex to select optimal ML models

For a specific inference task:

e.g.,  $(P_{car} \wedge P_{red}) \vee (P_{bus} \wedge P_{yellow})$



# Formalizing model repository

- Query
  - $(P_{car} \wedge P_{red}) \vee (P_{bus} \wedge P_{yellow})$
- Model repository
  - $\mathcal{R}(C\{M, P\}, A\{M, P\})$



Table 1: Example accuracy  $A$  of models in a repository.

	car	bus	red	yellow
Model 1	0.88	0	0	0
Model 2	0.98	0	0	0
Model 3	0	0.75	0	0
Model 4	0	0.95	0	0
Model 5	0	0	0.96	0.97
Model 6	0	0	0.97	0.98

Table 2: Example execution time  $C$  of models in a repository.

	car	bus	red	yellow
Model 1	15	$\infty$	$\infty$	$\infty$
Model 2	30	$\infty$	$\infty$	$\infty$
Model 3	$\infty$	20	$\infty$	$\infty$
Model 4	$\infty$	35	$\infty$	$\infty$
Model 5	$\infty$	$\infty$	5	5
Model 6	$\infty$	$\infty$	10	10

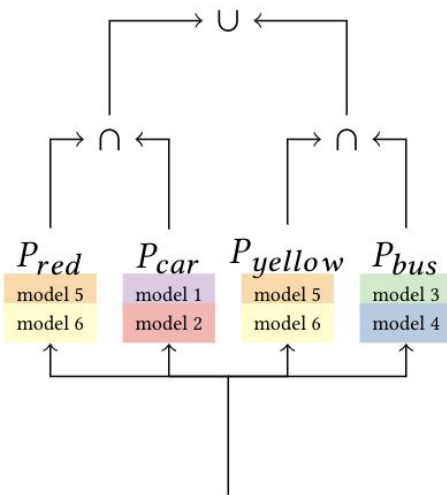
# Goal

- Generate query plans for inference queries defined on model repositories
  - Tackle the problem of optimal **model selection** and **predicate ordering**  
*under accuracy and execution time constraints*

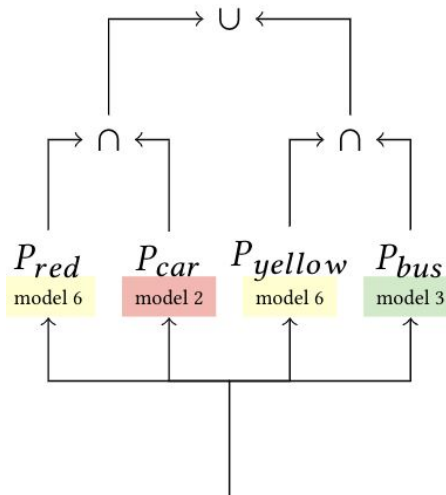


# We devise three approaches

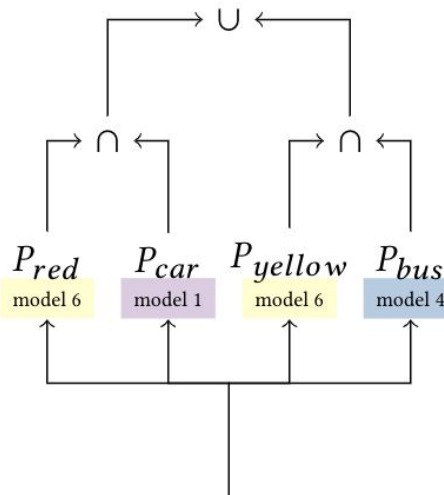
- Greedy (model selection)
- Model optimizer (model selection)
- Order optimizer (model selection + predicate ordering)



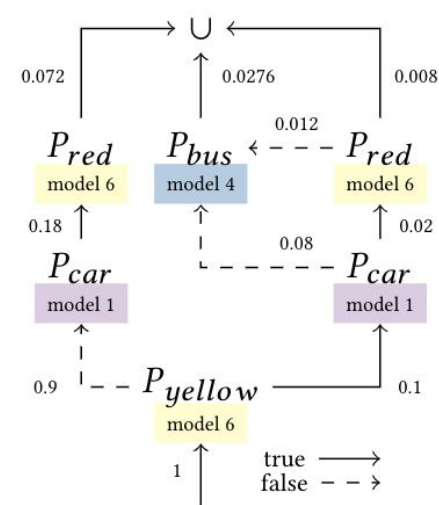
(a) Logical query plan



(b) Greedy query plan



(c) Model optimal query plan



(d) Order optimal query plan<sup>9</sup>

# We devise three approaches

- Greedy (model selection)
  - Model optimizer (model selection)
  - Order optimizer (model selection + predicate ordering)
- 
1. Select Pareto-optimal models
  2. Loop over predicates and greedily select model with most accurate / least execution time

# We devise three approaches

- Greedy (model selection)
- Model optimizer (model selection)
- Order optimizer (model selection + predicate ordering)

Apply Mixed Integer Programming:

- Model the accuracy of the query
- Model the execution time of the models
- Maximize accuracy / Minimize execution time

	car	bus	red	yellow	
Model 1	$x_{m1,car}$	$x_{m1,bus}$	$x_{m1,red}$	$x_{m1,yellow}$	$B_{m1} = \{0, 1\} \cdot c_{m1}$
Model 2					$B_{m2} = \{0, 1\} \cdot c_{m2}$
Model 3	...	...	...	...	$B_{m3} = \{0, 1\} \cdot c_{m3}$
Model 4					$B_{m4} = \{0, 1\} \cdot c_{m4}$
Model 5					$B_{m5} = \{0, 1\} \cdot c_{m5}$
Model 6	...	...	...	...	$B_{m6} = \{0, 1\} \cdot c_{m6}$
	$\sum = 1$	$\sum = 1$	$\sum = 1$	$\sum = 1$	$\sum$

Cost model

Accuracy model

$$a = (a_{red} \cdot a_{car}) + (a_{yellow} \cdot a_{bus}) - (a_{red} \cdot a_{car}) \cdot (a_{yellow} \cdot a_{bus})$$

# We devise three approaches

- Greedy (model selection)
- Model optimizer (model selection)
- Order optimizer (model selection + predicate ordering)

Apply Mixed Integer Programming:

- **Model the order of predicates**
- Model the accuracy of the query
- **Model the execution time of the models while taking into account of selectivity**
- Maximize accuracy / Minimize execution time

Predicate ordering

	1	2	3	4
red	1	0	0	0
car	0	1	0	0
yellow	0	0	1	0
bus	0	0	0	1

Model selection

	car	bus	red	yellow
Model 1	0	0	0	0
Model 2	1	0	0	0
Model 3	...	...	...	...
Model 4				
Model 5				
Model 6	...	...	...	...

# Formalizing model repository

- Query
  - $(P_{car} \wedge P_{red}) \vee (P_{bus} \wedge P_{yellow})$
- Model repository
  - $\mathcal{R}(C\{M, P\}, A\{M, P\})$
  - **Selectivity(P)**

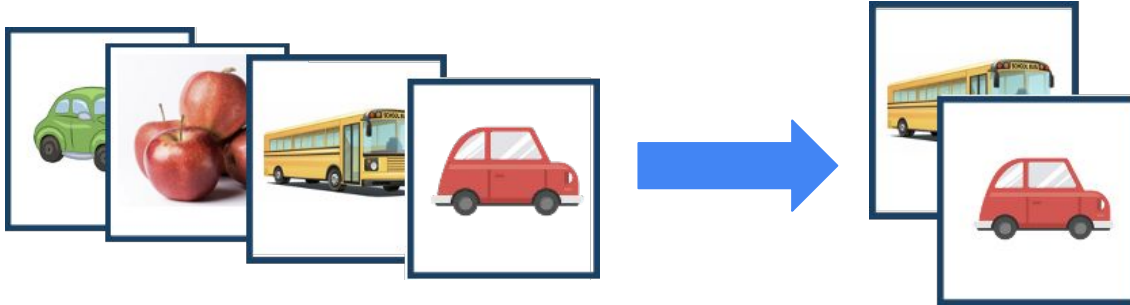


Table 1: Example accuracy  $A$  of models in a repository.

	car	bus	red	yellow
Model 1	0.88	0	0	0
Model 2	0.98	0	0	0
Model 3	0	0.75	0	0
Model 4	0	0.95	0	0
Model 5	0	0	0.96	0.97
Model 6	0	0	0.97	0.98

Table 2: Example execution time  $C$  of models in a repository.

	car	bus	red	yellow
Model 1	15	$\infty$	$\infty$	$\infty$
Model 2	30	$\infty$	$\infty$	$\infty$
Model 3	$\infty$	20	$\infty$	$\infty$
Model 4	$\infty$	35	$\infty$	$\infty$
Model 5	$\infty$	$\infty$	5	5
Model 6	$\infty$	$\infty$	10	10

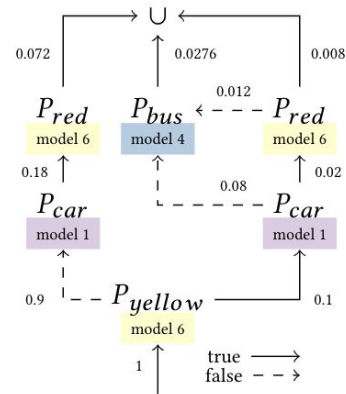
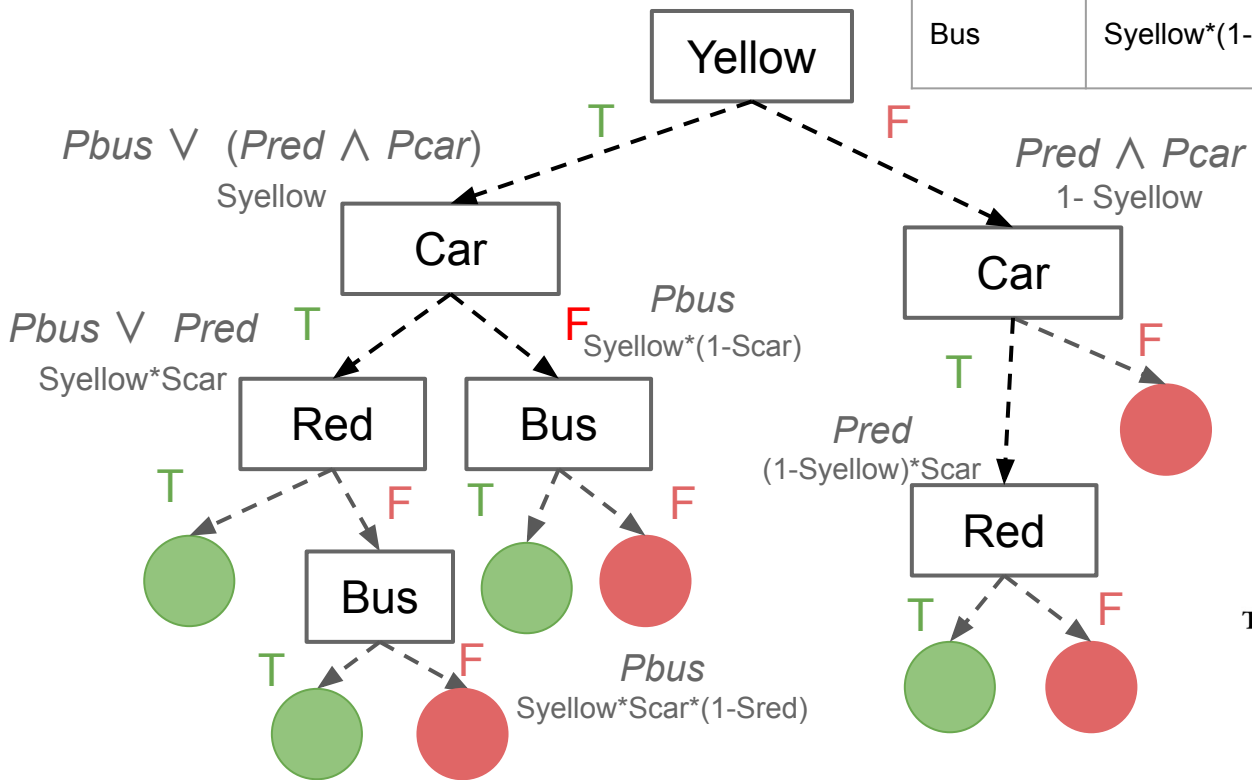
Table 3: Example selectivity of predicates in a dataset.

	car	bus	red	yellow
selectivity	0.2	0.3	0.4	0.1

$$(Pred \wedge P_{car}) \vee (P_{yellow} \wedge P_{bus})$$

Yellow  $\rightarrow$  Car  $\rightarrow$  Red  $\rightarrow$  Bus

Predicate	Data Proportion	Condition
Yellow	1	“
Car	1	“
Red	Scar	$P_{car} = 1$
Bus	$S_{yellow} * (1 - Scar * S_{red})$	$P_{yellow} = 1$ and $(P_{car} = 1$ or $P_{red} = 1)$



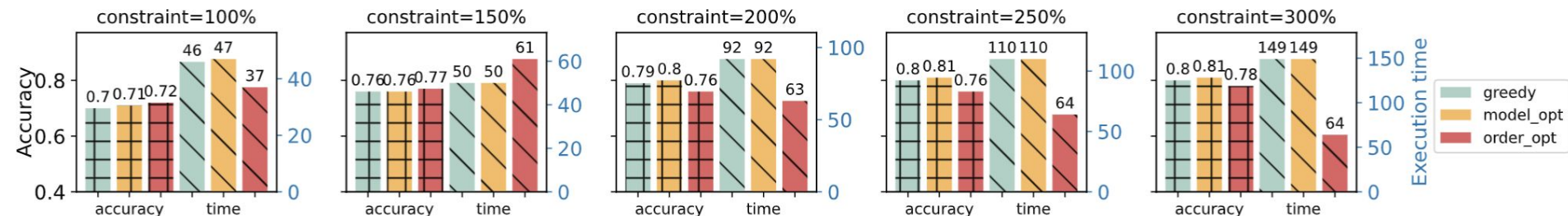
(d) Order optimal query plan

Table 3: Example selectivity of predicates in a dataset.

	car	bus	red	yellow
selectivity	0.2	0.3	0.4	0.1

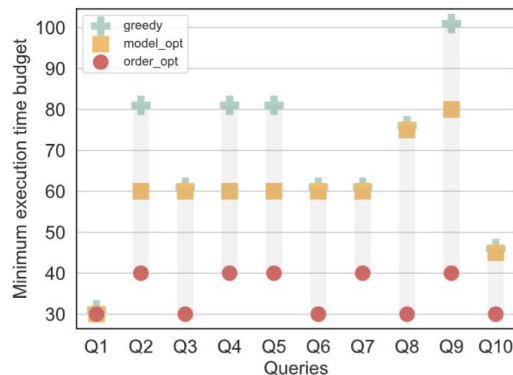
# Preliminary results: accuracy vs execution time

Test on query generated from COCO classes and evaluate on validation set.  
125 model variants generated from YOLOv3 and YOLOv5.

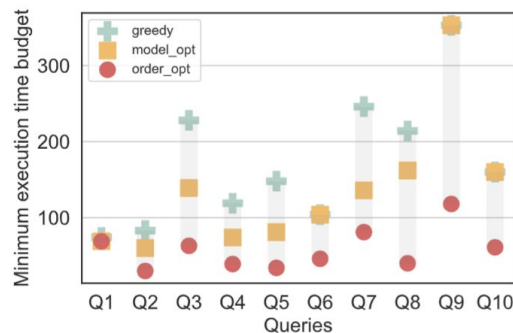


**Figure 6: Performance of Query 4**

# Preliminary results

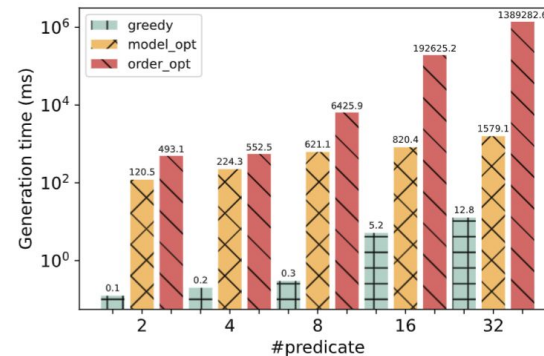


(a) Minimum execution time constraint

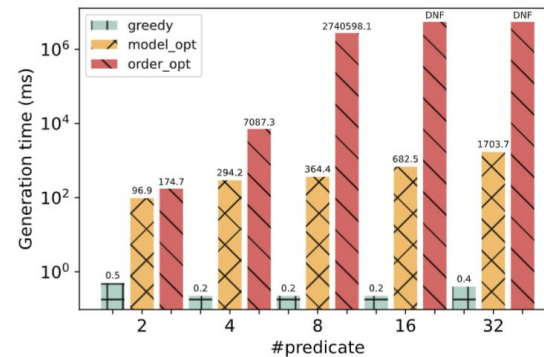


(b) Maximum accuracy constraint

Figure 7: Comparison on the execution time budget with the boundary constraint



(a) Constrained on execution time



(b) Constrained on accuracy

Figure 8: Generation time of a query plan varying number of predicates



# Takeaways

- We motivate the problem by highlighting the emergence of repositories of ML models
  - Available models along with their metadata descriptions.
- We propose three query optimization strategies
  - We evaluate them on a model repository that we construct from real models using queries defined over the COCO dataset
- Our greedy optimizer is the fastest in generating query plans, but our order optimizer produces substantially better query plans when a tight constraint is encountered