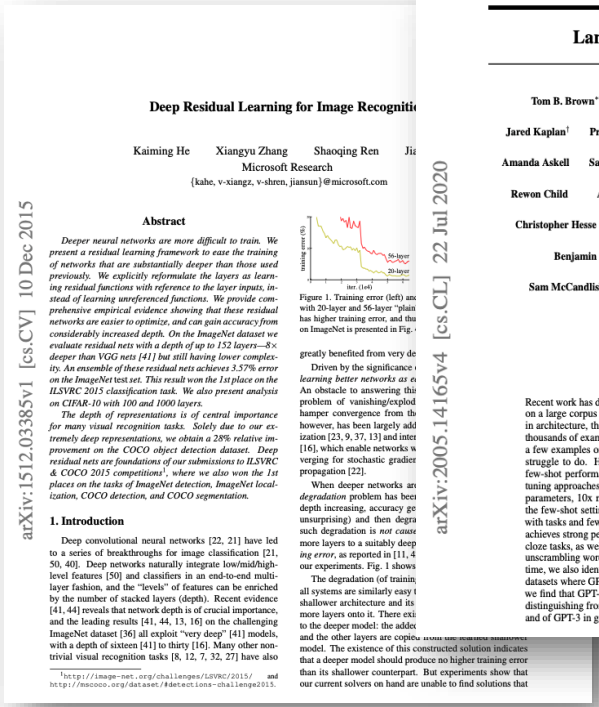
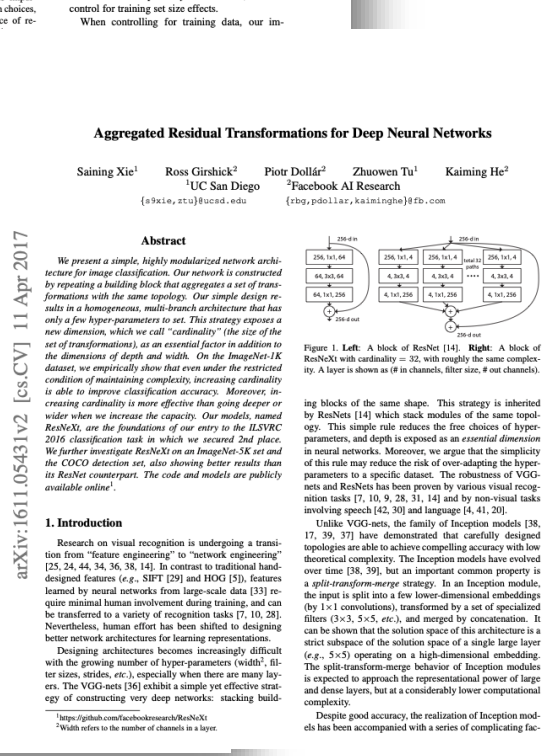
Patrik Okanovic, Andreas Kirsch, Jannes Kasper, Torsten Hoefler, Andreas Krause, Nezihe Merve Gürel

All models are wrong, some are useful: Model Selection with Limited Labels

Motivation

Growing size of pretrained model!





Hugging Face

PyTorch

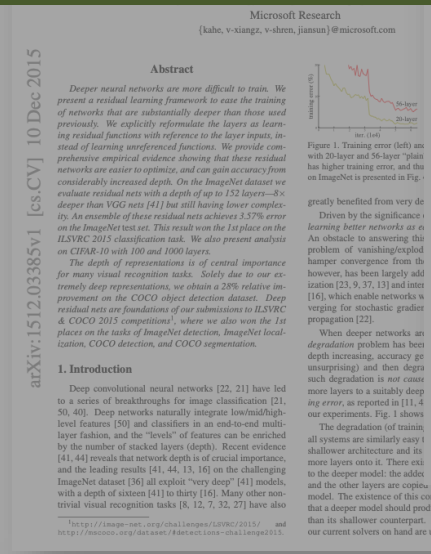
TensorFlow Hub

Motivation

Growing size of pretrained model!

Hugging Face

How to select the best pretrained model?


Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

Abstract

Deeper neural networks are more difficult to train. We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions. We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth. On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [41] but still having lower complexity. An ensemble of these residual nets achieves 3.57% error on the ImageNet test set. This result won the 1st place on the ILSVRC 2015 classification task. We also present analysis on CIFAR-10 with 100 and 1000 layers.

The depth of representations is of central importance for many visual recognition tasks. Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset. Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions[1], where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

1. Introduction

Deep convolutional neural networks [22, 21] have led to a series of breakthroughs for image classification [21, 50, 40]. Deep networks naturally integrate low/mid/high-level features [50] and classifiers in an end-to-end multi-layer fashion, and the "levels" of features can be enriched by the number of stacked layers (depth). Recent evidence [41, 44] reveals that network depth is of crucial importance, and the leading results [41, 44, 13, 16] on the challenging ImageNet dataset [36] all exploit "very deep" [41] models, with a depth of sixteen [41] to thirty [16]. Many other non-trivial visual recognition tasks [8, 12, 7, 32, 27] have also

[1] http://image-net.org/challenges/LSVRC/2015/ and http://mscoco.org/dataset/#detections-challenge2015.

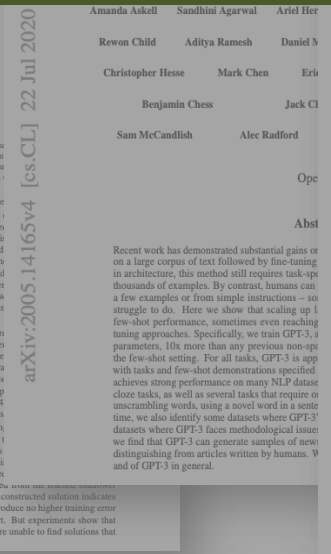
Figure 1. Training error (left) and with 20-layer and 56-layer "plain has higher training error, and tha on ImageNet is presented in Fig.

greatly benefited from very de

Driven by the significance learning better networks as e An obstacle to answering thi problem of vanishing/explod hamper convergence from th however, has been largely add ization [23, 9, 37, 13] and inter [16], which enable networks w verging for stochastic gradien propagation [22].

When deeper networks ar degradation problem has been depth increasing, accuracy ge unsurprising) and then degra such degradation is not caus more layers to a suitably deep ing error, as reported in [11, 4 our experiments. Fig. 1 shows

The degradation (of trainin all systems are similarly easy t shallower architecture and its more layers onto it. There exi to the deeper model: the added and the other layers are copie model. The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart. But experiments show that our current solvers on hand are unable to find solutions that


Amanda Askell Sandhini Agarwal Ariel Her

Rewon Child Aditya Ramesh Daniel M

Christopher Hesse Mark Chen Eric

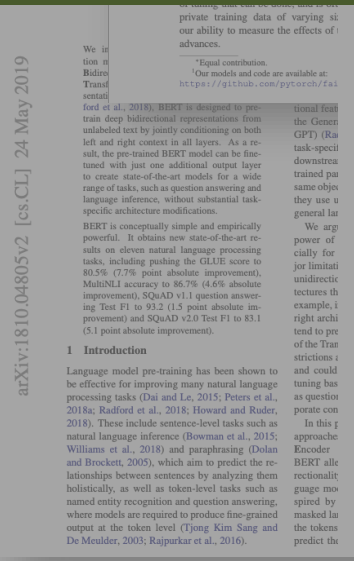
Benjamin Chess Jack Cl

Sam McCandlish Alec Radford

Ope

Abst

Recent work has demonstrated substantial gains o on a large corpus of text followed by fine-tuning in architecture, this method still requires task-spe thousands of examples. By contrast, humans can a few examples or from simple instructions – so struggle to do. Here we show that scaling up l few-shot performance, sometimes even reaching tuning approaches. Specifically, we train GPT-3, a parameters, 10x more than any previous non-spa the few-shot setting. For all tasks, GPT-3 is app with tasks and few-shot demonstrations specified achieves strong performance on many NLP datase cloze tasks, as well as several tasks that require o unscrambling words, using a novel word in a sente time, we also identify some datasets where GPT-3' datasets where GPT-3 faces methodological issue we find that GPT-3 can generate samples of new distinguishing from articles written by humans. W and of GPT-3 in general.


private training data of varying si our ability to measure the effects of advances.

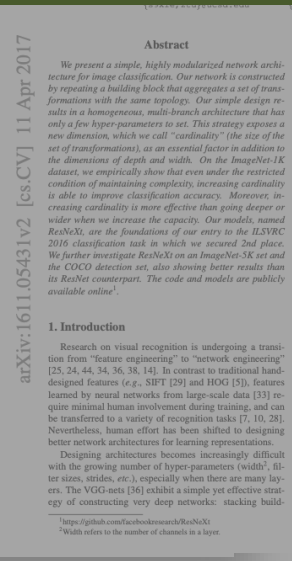
*Equal contribution.
†Our models and code are available at: https://github.com/pytorch/fai

ford et al., 2018), BERT is designed to pre- train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

1 Introduction

Language model pre-training has been shown to be effective for improving many natural language processing tasks (Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018). These include sentence-level tasks such as natural language inference (Bowman et al., 2015; Williams et al., 2018) and paraphrasing (Dolan and Brockett, 2005), which aim to predict the relationships between sentences by analyzing them holistically, as well as token-level tasks such as named entity recognition and question answering, where models are required to produce fine-grained output at the token level (Tjong Kim Sang and De Meulder, 2003; Rajpurkar et al., 2016).


Abstract

We present a simple, highly modularized network architecture for image classification. Our network is constructed by repeating a building block that aggregates a set of transformations with the same topology. Our simple design results in a homogeneous, multi-branch architecture that has only a few hyper-parameters to set. This strategy exposes a new dimension, which we call "cardinality" (the size of the set of transformations), as an essential factor in addition to the dimensions of depth and width. On the ImageNet-1K dataset, we empirically show that even under the restricted condition of maintaining complexity, increasing cardinality is able to improve classification accuracy. Moreover, increasing cardinality is more effective than going deeper or wider when we increase the capacity. Our models, named ResNeXt, are the foundations of our entry to the ILSVRC 2016 classification task in which we secured 2nd place. We further investigate ResNeXt on an ImageNet-5K set and the COCO detection set, also showing better results than its ResNet counterpart. The code and models are publicly available online[1].

1. Introduction

Research on visual recognition is undergoing a transition from "feature engineering" to "network engineering" [25, 24, 44, 34, 36, 38, 14]. In contrast to traditional hand-designed features (*e.g.*, SIFT [29] and HOG [5]), features learned by neural networks from large-scale data [33] require minimal human involvement during training, and can be transferred to a variety of recognition tasks [7, 10, 28]. Nevertheless, human effort has been shifted to designing better network architectures for learning representations.

Designing architectures becomes increasingly difficult with the growing number of hyper-parameters (width[2], filter sizes, strides, *etc.*), especially when there are many layers. The VGG-nets [36] exhibit a simple yet effective strategy of constructing very deep networks: stacking build-

[1] https://github.com/facebookresearch/ResNeXt
[2] Width refers to the number of channels in a layer.

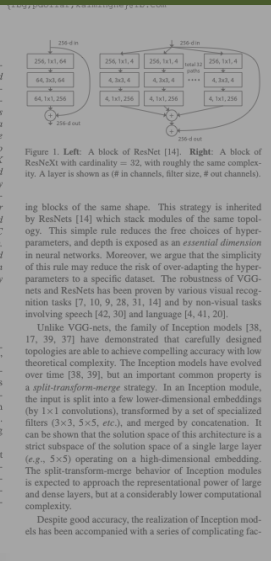


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

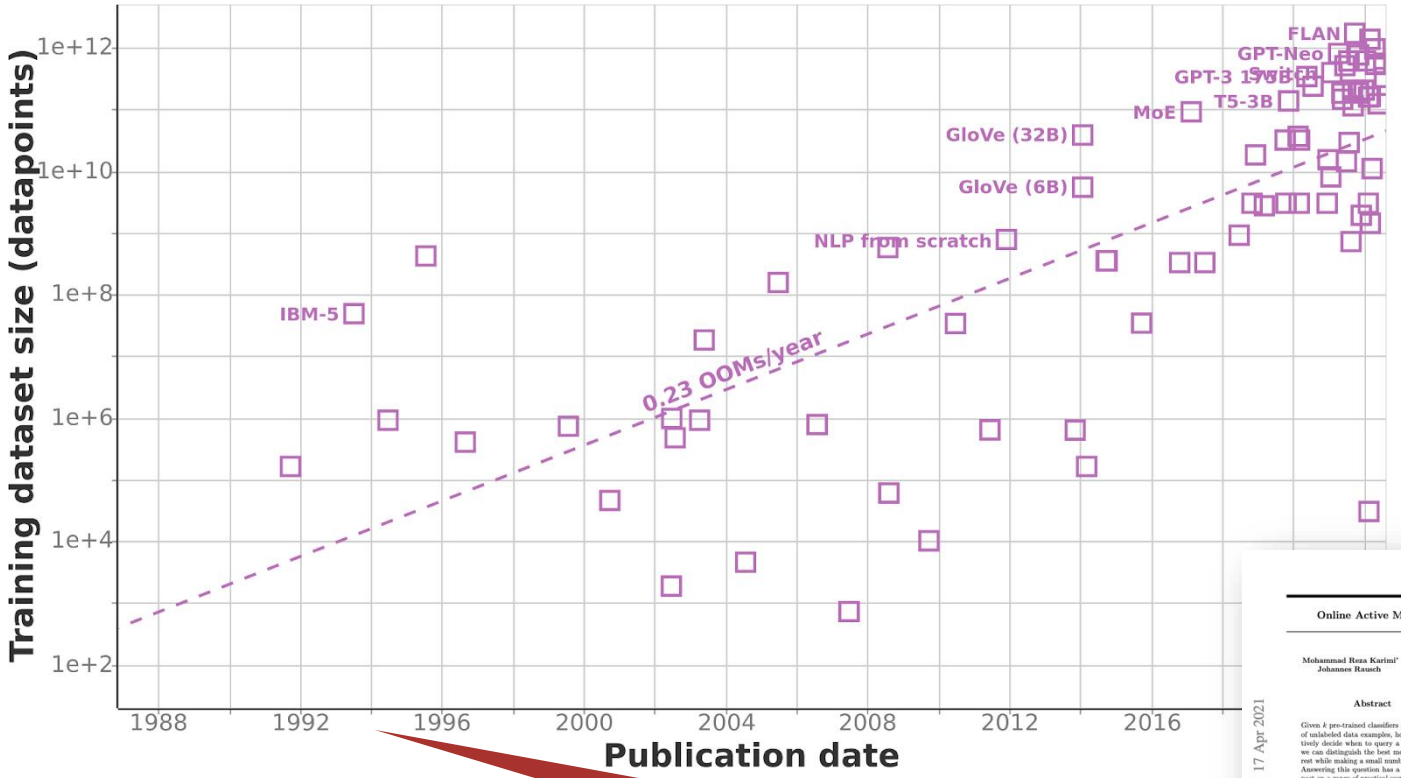
ing blocks of the same shape. This strategy is inherited by ResNets [14] which stack modules of the same topology. This simple rule reduces the free choices of hyper-parameters, and depth is exposed as an *essential dimension* in neural networks. Moreover, we argue that the simplicity of this rule may reduce the risk of over-adapting the hyper-parameters to a specific dataset. The robustness of VGG-nets and ResNets has been proven by various visual recognition tasks [7, 10, 9, 28, 31, 14] and by non-visual tasks involving speech [42, 30] and language [4, 41, 20].

Unlike VGG-nets, the family of Inception models [38, 17, 39, 37] have demonstrated that carefully designed topologies are able to achieve compelling accuracy with low theoretical complexity. The Inception models have evolved over time [38, 39], but an important common property is a *split-transform-merge* strategy. In an Inception module, the input is split into a few lower-dimensional embeddings (by 1×1 convolutions), transformed by a set of specialized filters (3×3, 5×5, *etc.*), and merged by concatenation. It can be shown that the solution space of this architecture is a strict subspace of the solution space of a single large layer (*e.g.*, 5×5) operating on a high-dimensional embedding. The split-transform-merge behavior of Inception modules is expected to approach the representational power of large and dense layers, but at a considerably lower computational complexity.

Despite good accuracy, the realization of Inception models has been accompanied with a series of complicating fac-


TensorFlow Hub

Related Work

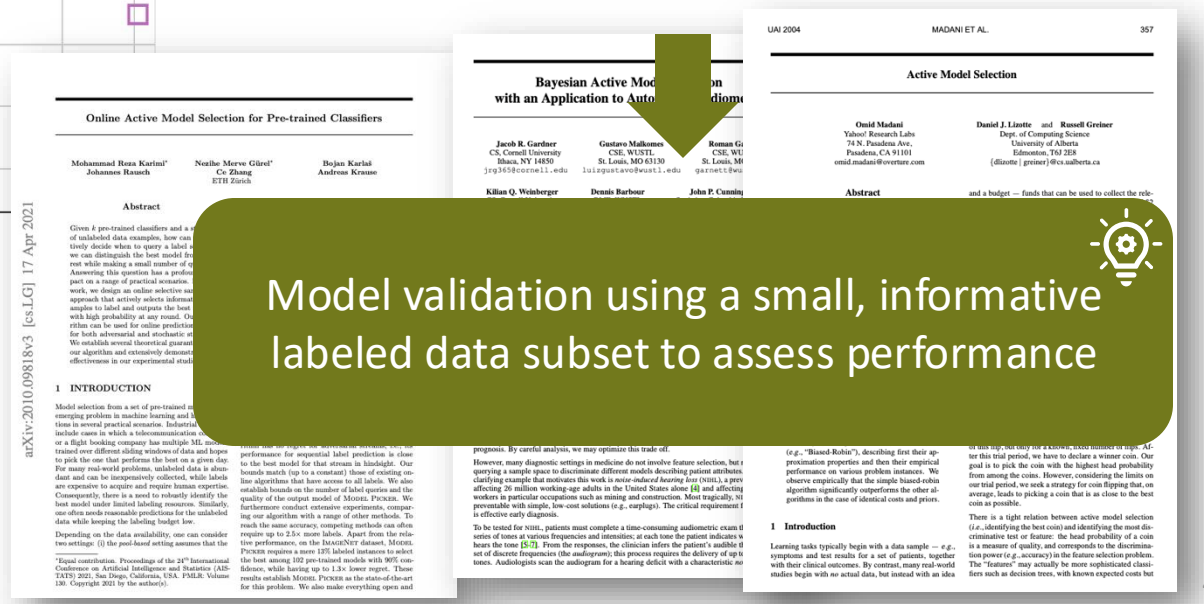


Hard to evaluate on **all** datapoints!

Active Model Selection (AMS)

Model validation using a small, informative labeled data subset to assess performance

Many unlabeled datapoints!



[1] https://epoch.ai/blog/trends-in-training-dataset-sizes

Assumptions in Previous Works

Fixed number of available models ☹

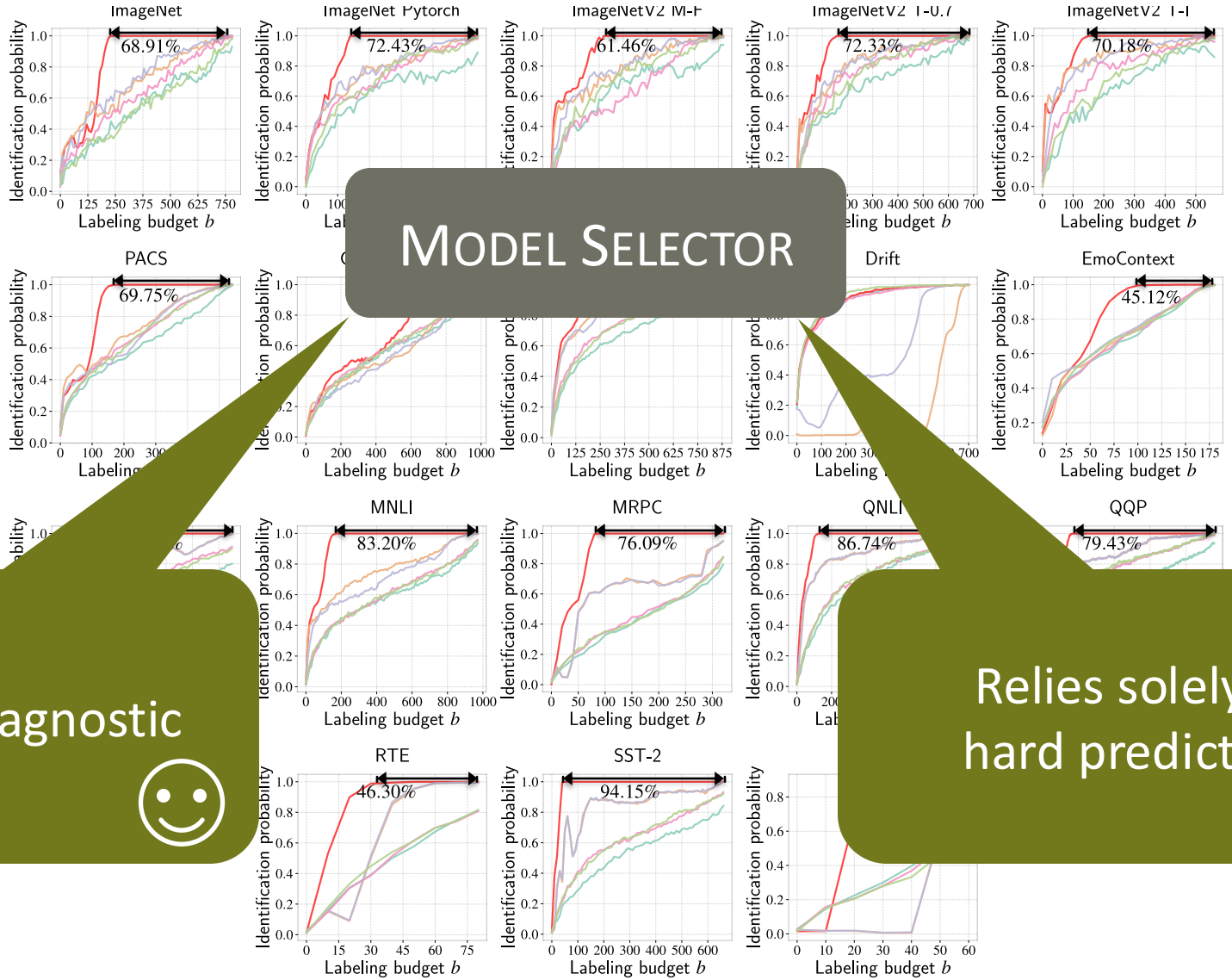
Restricted to specific model families ☹

Restricted to specific model families ☹

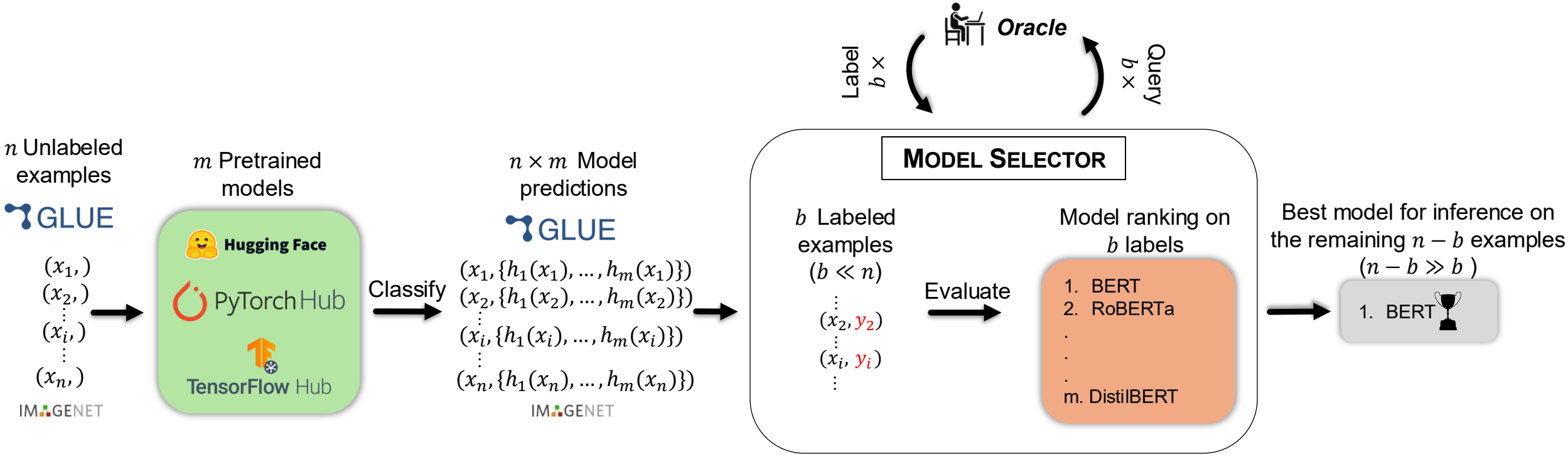
Assumptions in Previous Works

How can we efficiently select the most informative examples for labelling to choose the best classifier in a model-agnostic way?

Contributions



Framework


n Unlabeled examples
GLUE
$(x_1,)$
$(x_2,)$
$(x_i,)$
$(x_n,)$
IMAGENET
m Pretrained models
Hugging Face
PyTorch Hub
TensorFlow Hub
Classify
$n \times m$ Model predictions
GLUE
$(x_1, \{h_1(x_1), \dots, h_m(x_1)\})$
$(x_2, \{h_1(x_2), \dots, h_m(x_2)\})$
$(x_i, \{h_1(x_i), \dots, h_m(x_i)\})$
$(x_n, \{h_1(x_n), \dots, h_m(x_n)\})$
IMAGENET
Oracle
Label $b \times$
Query $b \times$
MODEL SELECTOR
b Labeled examples $(b \ll n)$
(x_2, y_2)
(x_i, y_i)
Evaluate
Model ranking on b labels
1. BERT
2. RoBERTa
m. DistilBERT
Best model for inference on the remaining $n - b$ examples $(n - b \gg b)$
1. BERT

Model Selector

Accuracy is our utility measure

Binary outcome of correct versus incorrect predictions

$$\mathbb{P}(H(x) \neq y | H = h^*) = \epsilon,$$
$$\mathbb{P}(H(x) = y | H = h^*) = 1 - \epsilon$$

Error probabiliy

Best model

Yuxin Chen, S Hamed Hassani, Amin Karbasi, and Andreas Krause. Sequential information maximization: When is greedy near-optimal? In Conference on Learning Theory, pp. 338–363. PMLR, 2015.

MODEL SELECTOR

$$\mathbb{P}(H(x) \neq y | H = h^*) = \epsilon,$$
$$\mathbb{P}(H(x) = y | H = h^*) = 1 - \epsilon$$

Objective: find the set of labeled data examples with max information

$$\mathcal{L}_{\text{OPT}[b]} := \underset{\substack{\mathcal{L} \subset \{(x_i, y_i) \,|\, i \in [n]\} \\ s.t.\ |\mathcal{L}| \leq b}}{\arg\max} \mathbb{I}(H; \mathcal{L})$$

Mutual information

MODEL SELECTOR

$$\mathbb{P}(H(x) \neq y | H = h^*) = \epsilon,$$
$$\mathbb{P}(H(x) = y | H = h^*) = 1 - \epsilon$$

$$\mathcal{L}_{\text{OPT}[b]} := \underset{\substack{\mathcal{L} \subset \{(x_i, y_i) \,|\, i \in [n]\} \\ s.t.\ |\mathcal{L}| \leq b}}{\arg\max} \mathbb{I}(H; \mathcal{L})$$

Strategy of MODEL SELECTOR is to **greedily** pick the unlabeled example that provides the maximal information gain

$$x_t = \underset{x \in \mathcal{U}_t}{\arg\max}\ \mathbb{I}(H; Y | x, \mathcal{L}_t)$$

Unlabeled data

Labeled data

MODEL SELECTOR

$$\mathbb{P}(H(x) \neq y | H = h^*) = \epsilon,$$
$$\mathbb{P}(H(x) = y | H = h^*) = 1 - \epsilon$$

$$\mathcal{L}_{\text{OPT}[b]} := \underset{\substack{\mathcal{L} \subset \{(x_i, y_i) \,|\, i \in [n]\} \\ s.t.\ |\mathcal{L}| \leq b}}{\arg\max} \ \mathbb{I}(H; \mathcal{L})$$

$$x_t = \underset{x \in \mathcal{U}_t}{\arg\max} \ \mathbb{I}(H; Y | x, \mathcal{L}_t)$$

Differential entropies

$$x_t = \underset{x \in \mathcal{U}_t}{\arg\max} \ \mathbb{H}(H|\mathcal{L}_t) - \mathbb{E}_Y[\mathbb{H}(H|\mathcal{L}_t \cup \{(x, Y)\})]$$
$$= \underset{x \in \mathcal{U}_t}{\arg\min} \ \mathbb{E}_Y[\mathbb{H}(H|\mathcal{L}_t \cup \{(x, Y)\})]$$

Equivalent to minimizing the model posterior uncertainty

Quoc Phong Nguyen, Bryan Kian Hsiang Low, and Patrick Jaillet. An information-theoretic framework for unifying active learning problems. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pp. 9126–9134, 2021.

MODEL SELECTOR

Hypothetical model posterior

Hypothetical label

$$\mathbb{P}(H = h_j | \mathcal{L}_t \cup \{(x, Y = c)\}) \propto \mathbb{P}(\mathcal{L}_t \cup \{(x, Y = c)\}) | H = h_j)\ \mathbb{P}(H = h_j)$$

$$\mathbb{P}(H(x) \neq y | H = h^*) = \epsilon,$$
$$\mathbb{P}(H(x) = y | H = h^*) = 1 - \epsilon$$

$$\mathcal{L}_{\text{OPT}[b]} := \underset{\substack{\mathcal{L} \subset \{(x_i, y_i) | i \in [n]\} \\ s.t.\ |\mathcal{L}| \leq b}}{\arg\max} \mathbb{I}(H; \mathcal{L})$$

$$x_t = \underset{x \in \mathcal{U}_t}{\arg\max}\ \mathbb{I}(H; Y | x, \mathcal{L}_t)$$

$$x_t = \underset{x \in \mathcal{U}_t}{\arg\max}\ \mathbb{H}(H | \mathcal{L}_t) - \mathbb{E}_Y[\mathbb{H}(H | \mathcal{L}_t \cup \{(x, Y)\})]$$
$$= \underset{x \in \mathcal{U}_t}{\arg\min}\ \mathbb{E}_Y[\mathbb{H}(H | \mathcal{L}_t \cup \{(x, Y)\})]$$

Model Selector

$$\mathbb{P}(H(x) \neq y | H = h^*) = \epsilon,$$
$$\mathbb{P}(H(x) = y | H = h^*) = 1 - \epsilon$$

If we assume $P(H = h_j) = 1/m$

$$\mathcal{L}_{\text{OPT}[b]} := \underset{\substack{\mathcal{L} \subset \{(x_i, y_i) | i \in [n]\} \\ s.t.\ |\mathcal{L}| \leq b}}{\arg\max} \mathbb{I}(H; \mathcal{L})$$

$$x_t = \underset{x \in \mathcal{U}_t}{\arg\max}\ \mathbb{I}(H; Y | x, \mathcal{L}_t)$$

$$x_t = \underset{x \in \mathcal{U}_t}{\arg\max}\ \mathbb{H}(H | \mathcal{L}_t) - \mathbb{E}_Y[\mathbb{H}(H | \mathcal{L}_t \cup \{(x, Y)\})]$$
$$= \underset{x \in \mathcal{U}_t}{\arg\min}\ \mathbb{E}_Y[\mathbb{H}(H | \mathcal{L}_t \cup \{(x, Y)\})]$$

$$\mathbb{P}(H = h_j | \mathcal{L}_t \cup \{(x, Y = c)\}) \propto$$
$$\mathbb{P}(\mathcal{L}_t \cup \{(x, Y = c)\}) | H = h_j)\ \mathbb{P}(H = h_j)$$

$$\mathbb{P}(H = h_j | \mathcal{L}_t \cup \{(x, Y = c)\}) \propto$$
$$(1 - \epsilon)^{h_{j,(t,x)}} \epsilon^{t - h_{j,(t,x)}}$$

Number of correct predictions of classifier

Model Selector

At iteration t Model Selector selects the most informative x_t and queries *oracle* for y_t

Up to budget b

Updates

$$\mathcal{U}_{t+1} = \mathcal{U}_t \backslash \{x_t\}$$
$$\mathcal{L}_{t+1} = \mathcal{L}_t \cup \{x_t, y_t\}$$

$$\mathbb{P}(H = h_j | \mathcal{L}_{t+1}) \propto \mathbb{P}(H = h_j | \mathcal{L}_t)(1-\epsilon)^{\mathbb{1}[h_j(x_t)=y_t]}\epsilon^{\mathbb{1}[h_j(x_t)}$$

$$\mathbb{P}(H(x) \neq y | H = h^*) = \epsilon,$$
$$\mathbb{P}(H(x) = y | H = h^*) = 1 - \epsilon$$

$$\mathcal{L}_{\text{OPT}[b]} := \underset{\substack{\mathcal{L} \subset \{(x_i, y_i) | i \in [n]\} \\ s.t.\ |\mathcal{L}| \leq b}}{\arg\max} \mathbb{I}(H; \mathcal{L})$$

$$x_t = \underset{x \in \mathcal{U}_t}{\arg\max}\ \mathbb{I}(H; Y | x, \mathcal{L}_t)$$

$$x_t = \underset{x \in \mathcal{U}_t}{\arg\max}\ \mathbb{H}(H|\mathcal{L}_t) - \mathbb{E}_Y[\mathbb{H}(H|\mathcal{L}_t \cup \{(x, Y)\})]$$
$$= \underset{x \in \mathcal{U}_t}{\arg\min}\ \mathbb{E}_Y[\mathbb{H}(H|\mathcal{L}_t \cup \{(x, Y)\})]$$

$$\mathbb{P}(H = h_j | \mathcal{L}_t \cup \{(x, Y = c)\}) \propto \mathbb{P}(\mathcal{L}_t \cup \{(x, Y = c)\}) | H = h_j)\ \mathbb{P}(H = h_j)$$

$$\mathbb{P}(H = h_j | \mathcal{L}_t \cup \{(x, Y = c)\}) \propto (1-\epsilon)^{h_{j,(t,x)}} \epsilon^{t - h_{j,(t,x)}}$$

Selecting the Parameter ε

Assign noisy labels to datapoints

$$(x_1, \{h_1(x_1), \dots, h_m(x_1)\})$$
$$(x_2, \{h_1(x_2), \dots, h_m(x_2)\})$$
$$\vdots$$
$$(x_i, \{h_1(x_i), \dots, h_m(x_i)\})$$
$$\vdots$$
$$(x_n, \{h_1(x_n), \dots, h_m(x_n)\})$$

Efficient: reuse label predictions

A coarse-grained search with larger steps to identify promising intervals

$$\epsilon \in \{0.25, 0.3, 0.35, 0.4, 0.45, 0.5\}$$

A fine-grained search within these intervals

$$\epsilon \in \{0.4, 0.41, 0.42, 0.43, 0.44, 0.45\}$$

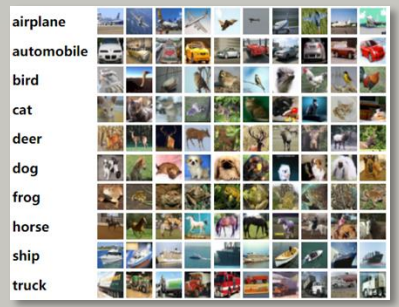
Datasets and Model Collections

Vision tasks

ResNets, EfficientNet, MobileNet,…

6 – 1,000 classes

Domain adaptation setting



NLP tasks

BERT, DistilBERT, RoBERTa, GPT,…

From just 71 examples to 40,000

GLUE

2 – 3 classes

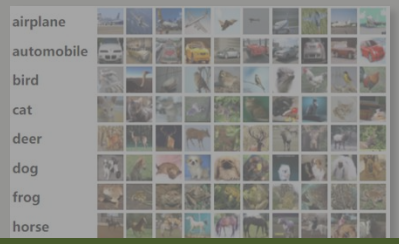
Datasets and Model Collections

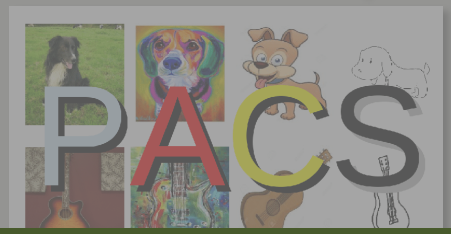
ResNets, EfficientNet, MobileNet,…

Domain adaptation setting

6 – 1,000 classes

Vision tasks





In total 18 model collections on 16 different datasets comprising over 1,500 pretrained models

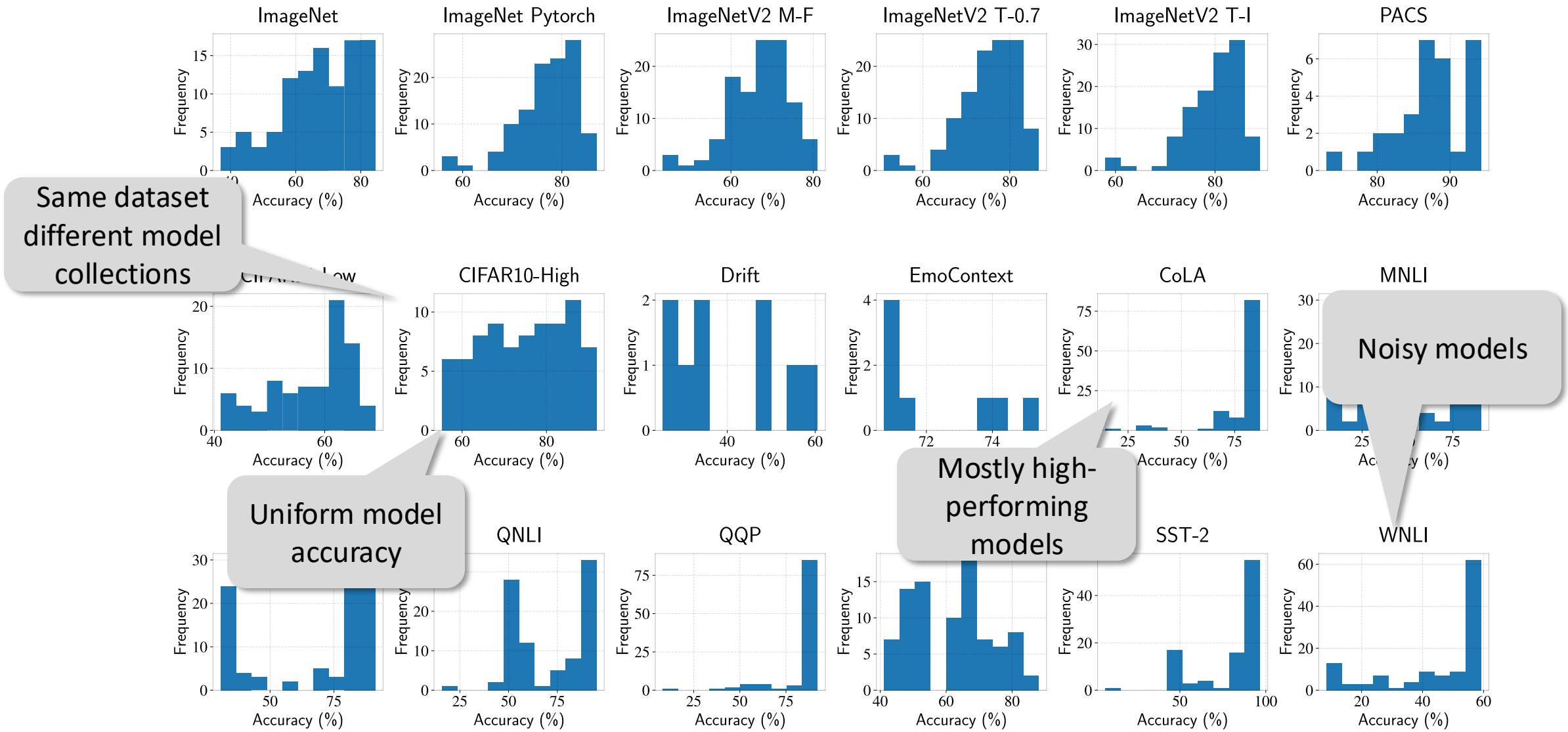
RoBERTa, GPT,…

From just 71 examples to 40,000

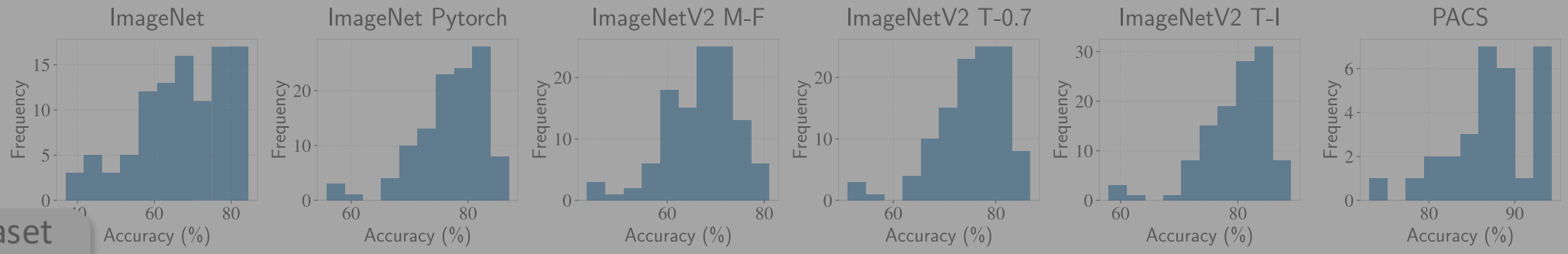
GLUE

2 – 3 classes

Datasets and Model Collections



Datasets and Model Collections

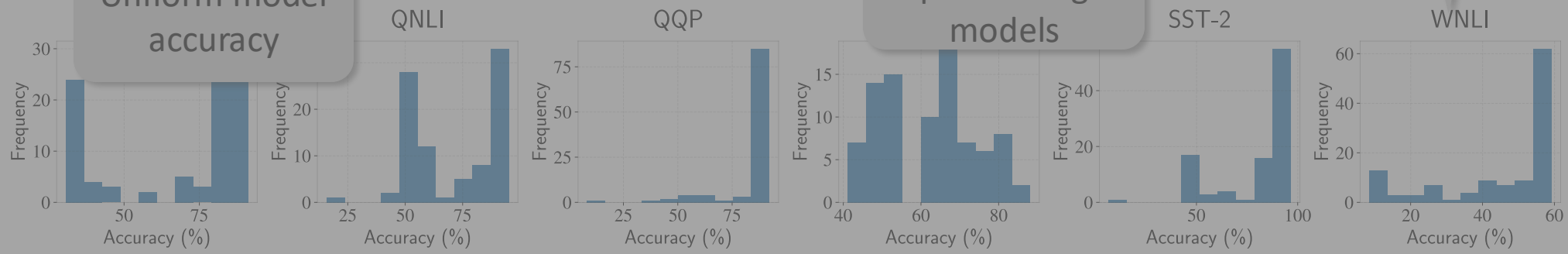


Same dataset different model

Wide range of possible model accuracies without making any assumptions about their distribution

Uniform model accuracy

Mostly high-performing models



Baselines

Random

Uncertainty Sampling

Margin Sampling

Originally comparing 2 models

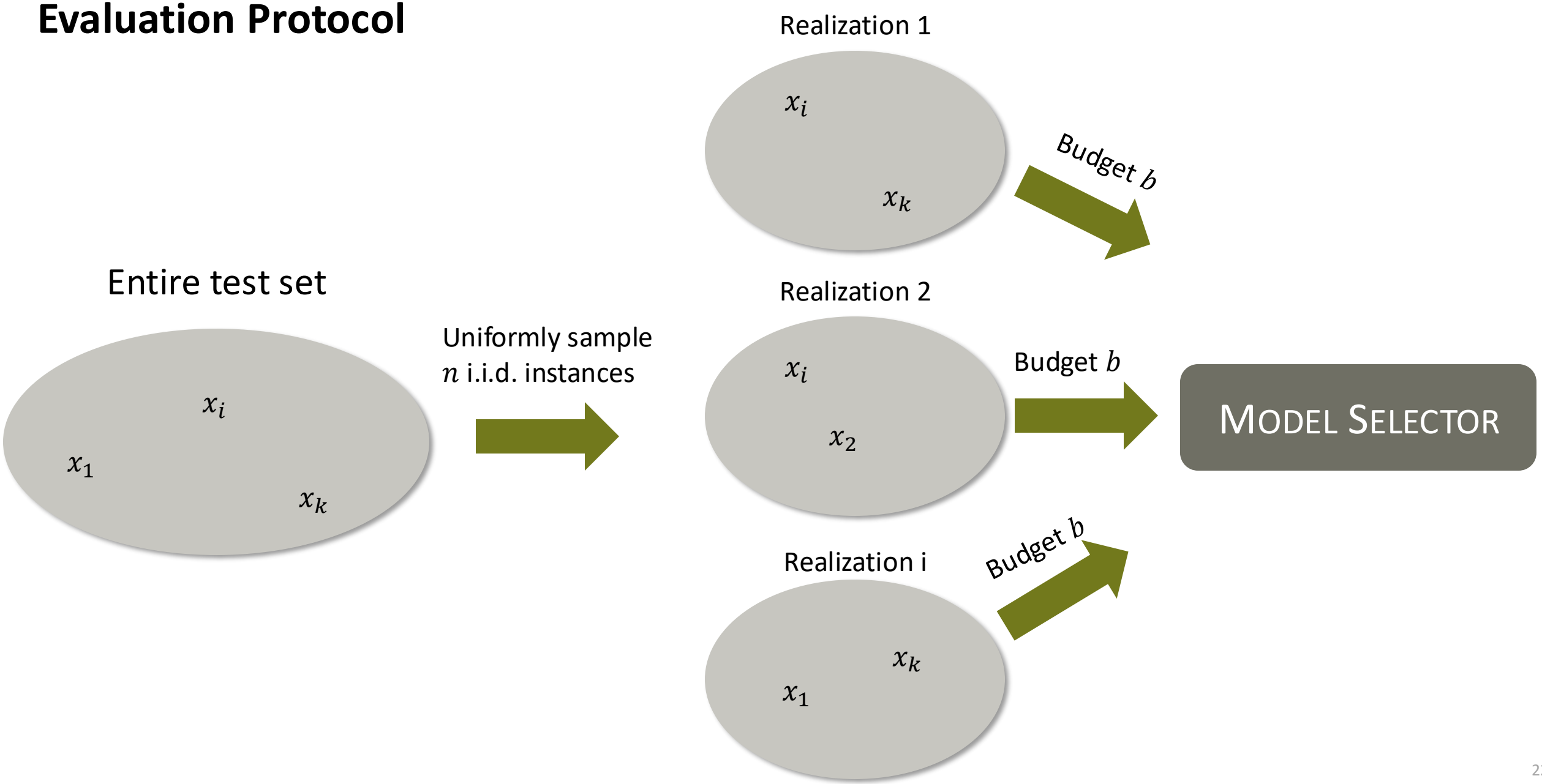
Active Model Comparison (AMC)

Variance Minimization Approach (VMA)

Requires soft predictions

[2] Ido Dagan and Sean P Engelson. Committee-based sampling for training probabilistic classifiers. In Machine Learning Proceedings 1995, pp. 150–157. Elsevier, 1995.
[3] H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In Proceedings of the fifth annual workshop on Computational learning theory, pp. 287–294, 1992.
[4] Christoph Sawade, Niels Landwehr, and Tobias Scheffer. Active comparison of prediction models. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), Advances in Neural Information Processing Systems, volume 25. Curran Associates, Inc., 2012.
[5] Mitsuru Matsuura and Satoshi Hara. Active model selection: A variance minimization approach. In NeurIPS 2023 Workshop on Adaptive Experimental Design and Active Learning in the Real World, 2023.

Evaluation Protocol



Performance Metrics

Identification Probability

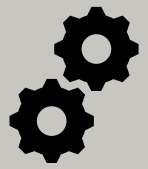
- Fraction of realizations where a method successfully identifies the true best model of that realization

Label Efficiency

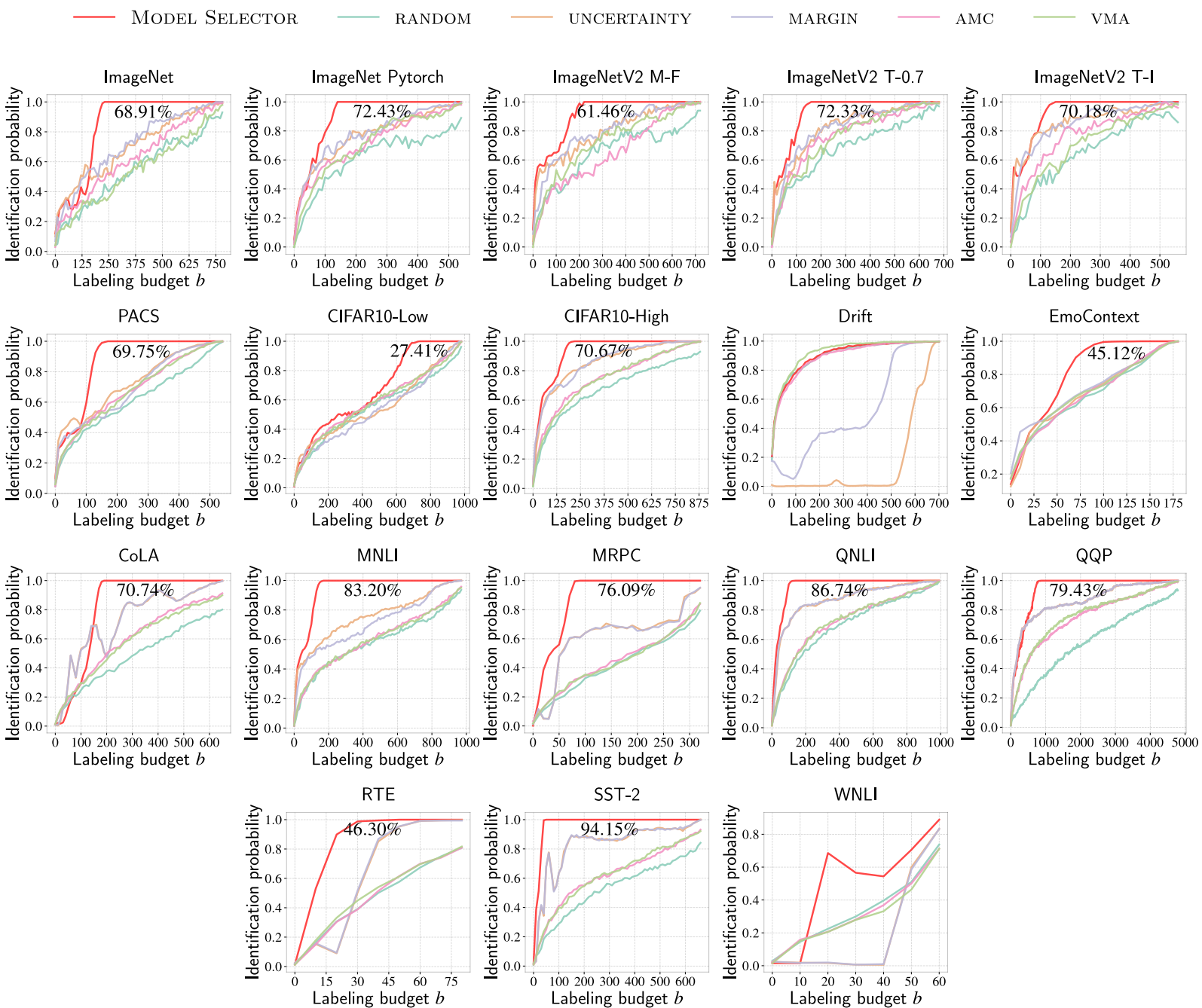
- The reduction in number of labels (%) required to select the best or a near-best (δ) model over all realizations

95thPercentile Accuracy Gap

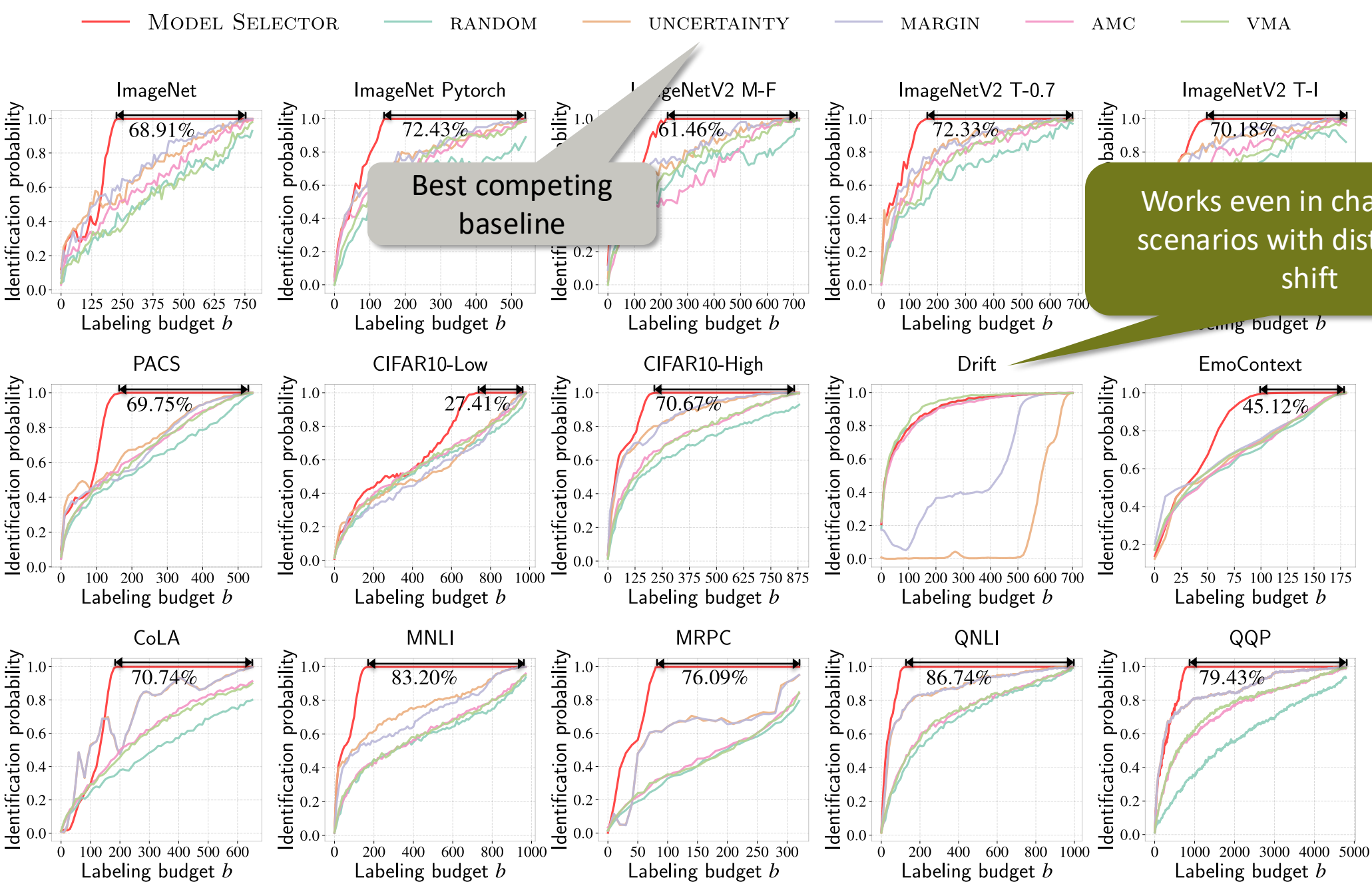
- Represents the 95th percentile of the accuracy gap across all realizations



Best Model Identification Probability



Best Model Identification Probability



Best Model Identification Probability

Reduces labeling cost by up to 94.15%

Label Efficiency for Near-Best Models

Dataset	$\delta = 1\%$	$\delta = 0.5\%$	$\delta = 0.1\%$
CIFAR10-High	↓ **48.04%**	↓ **58.40%**	↓ **72.23%**
CIFAR10-Low	↓ **21.07%**	↓ **21.82%**	↓ **25.67%**
EmoContext	↓ **20.56%**	↓ **34.19%**	↓ **39.89%**
PACS	↓ **62.73%**	↓ **66.81%**	↓ **68.62%**
Drift	↑ 23.79%	↑ 7.96%	↑ 11.18%
ImageNet	↓ **53.62%**	↓ **63.80%**	↓ **69.81%**
ImageNet Pytorch	↓ **40.94%**	↓ **64.07%**	↓ **73.36%**
ImageNetV2 T-I	↑ 6.12%	↓ **49.12%**	↓ **70.61%**
ImageNetV2 T-0.7	↓ **57.58%**	↓ **57.79%**	↓ **73.39%**
ImageNetV2 M-F	↓ **48.18%**	↓ **61.39%**	↓ **56.72%**
MRPC	↓ **72.41%**	↓ **73.62%**	↓ **74.54%**
CoLA	↓ **45.89%**	↓ **53.75%**	↓ **71.01%**
QNLI	↓ **46.88%**	↓ **78.39%**	↓ **85.75%**
QQP	↑ 11.90%	↓ **26.55%**	↓ **73.36%**
SST-2	↓ **7.89%**	↓ **39.66%**	↓ **93.33%**
WNLI	0.00%	0.00%	0.00%
MNLI	↓ **69.42%**	↓ **79.83%**	↑ 3.95%
RTE	↓ **40.96%**	↓ **40.96%**	↓ **40.96%**

Label Efficiency for Near-Best Models

Consistently reduces labeling cost to reach the δ vicinity of the true best model

$\delta = 1\%$	$\delta = 0.5\%$	$\delta = 0.1\%$
↓ **48.04%**	↓ **58.40%**	↓ **72.23%**
↓ **21.07%**	↓ **21.82%**	↓ **25.67%**
↓ **20.56%**	↓ **34.19%**	↓ **39.89%**
↓ **62.73%**	↓ **66.81%**	↓ **68.62%**
↑ 23.79%	↑ 7.96%	↑ 11.18%
↓ **53.62%**	↓ **63.80%**	↓ **69.81%**
↓ **40.94%**	↓ **64.07%**	↓ **73.36%**
↑ 6.12%	↓ **49.12%**	↓ **70.61%**
↓ **57.58%**	↓ **57.79%**	↓ **73.39%**
↓ **48.18%**	↓ **61.39%**	↓ **56.72%**
↓ **72.41%**	↓ **73.62%**	↓ **74.54%**
↓ **45.89%**	↓ **53.75%**	↓ **71.01%**
↓ **46.88%**	↓ **78.39%**	↓ **85.75%**
↑ 11.90%	↓ **26.55%**	↓ **73.36%**
↓ **7.89%**	↓ **39.66%**	↓ **93.33%**
0.00%	0.00%	0.00%
↓ **69.42%**	↓ **79.83%**	↑ 3.95%
↓ **40.96%**	↓ **40.96%**	↓ **40.96%**

Label Efficiency for Near-Best Models

Consistently reduces labeling cost to reach the δ vicinity of

$\delta = 1\%$	$\delta = 0.5\%$	$\delta = 0.1\%$
↓ **48.04%**	↓ **58.40%**	↓ **72.23%**
↓ **21.07%**	↓ **21.82%**	↓ **25.67%**
↓ **20.56%**	↓ **34.19%**	↓ **39.89%**
↓ **62.73%**	↓ **66.81%**	↓ **68.62%**
↑ 23.79%	↑ 7.96%	↑ 11.18%
↓ **53.62%**	↓ **63.80%**	↓ **69.81%**
↓ **40.94%**	↓ **64.07%**	↓ **73.36%**
↓ **72.41%**	↓ **73.62%**	↓ **74.54%**
↓ **45.89%**	↓ **53.75%**	↓ **71.01%**
↓ **46.88%**	↓ **78.39%**	↓ **85.75%**
↑ 11.90%	↓ **26.55%**	↓ **73.36%**
↓ **7.89%**	↓ **39.66%**	↓ **93.33%**
0.00%	0.00%	0.00%
↓ **69.42%**	↓ **79.83%**	↑ 3.95%
↓ **40.96%**	↓ **40.96%**	↓ **40.96%**

MODEL SELECTOR is consistently more label-efficient for near-best models

Robustness Analysis

Dataset Identification probability	MODEL SELECTOR (70%/80%/90%/100%)	RANDOM (70%/80%/90%/100%)	MARGIN (70%/80%/90%/100%)	UNCERTAINTY (70%/80%/90%/100%)	AMC (70%/80%/90%/100%)	VMA (70%/80%/90%/100%)
CIFAR10-High	**1.90/0.80/0.40/0.00**	5.00/3.90/3.50/3.00	2.00/1.30/1.30/1.10	2.50/1.50/1.00/0.70	3.80/2.60/2.10/1.80	4.00/3.00/2.60/1.90
CIFAR10-Low	**1.40/0.90/0.50/0.00**	2.00/1.80/1.40/1.30	2.10/1.80/1.60/1.40	2.10/1.80/1.50/1.30	1.70/1.40/1.20/1.10	2.00/1.60/1.50/1.30
EmoContext	1.30/**0.60/0.30/0.00**	**1.10**/1.00/0.90/0.70	2.00/1.00/0.80/0.50	1.50/1.10/0.70/0.50	1.40/1.00/0.80/0.50	1.20/1.00/0.90/0.50
PACS	**1.40/1.10/0.40/0.00**	1.90/1.70/1.70/1.70	1.80/1.80/1.80/1.80	1.70/1.60/1.50/1.50	1.70/1.60/1.50/1.40	1.80/1.60/1.70/1.50
Drift	**11.33**/8.27/5.87/**0.00**	11.47/**7.87**/6.27/**0.00**	16.67/16.67/13.87/7.60	18.00/17.33/10.00/10.00	11.87/8.13/6.53/**0.00**	11.60/9.47/**3.60**/**0.00**
ImageNet	**0.90/0.90/0.80/0.00**	2.30/2.20/2.10/2.10	1.20/1.20/1.20/1.10	1.10/1.50/1.30/1.30	1.70/1.70/1.30/1.40	1.70/1.70/1.70/1.70
ImageNet Pytorch	**0.80/0.50/0.20/0.00**	3.70/3.30/3.00/2.60	1.30/0.90/0.80/0.70	1.00/1.00/1.00/0.80	2.20/1.90/1.30/1.20	3.60/2.40/1.90/1.20
ImageNetV2 T-I	**1.20/0.70/0.10/0.00**	4.30/4.50/3.00/2.20	1.30/1.30/1.10/0.50	1.70/1.40/0.60/0.60	3.50/2.80/1.90/1.80	3.10/2.40/2.30/1.60
ImageNetV2 T-0.7	**1.00/0.50/0.20/0.00**	4.20/3.70/3.50/2.50	1.50/1.30/1.10/1.10	1.50/1.50/1.30/1.00	2.60/2.40/1.80/1.30	2.80/2.70/2.30/1.80
ImageNetV2 M-F	**0.90/0.40/0.30/0.00**	4.10/2.60/2.60/2.60	1.10/1.00/0.90/0.60	1.10/1.10/0.90/0.60	3.10/1.10/1.10/0.90	3.70/1.70/1.60/1.60
MRPC	**1.14/1.14/0.29/0.00**	5.71/5.43/5.14/4.86	2.00/1.43/1.14/0.86	1.71/**1.14**/0.86/1.14	5.43/5.14/5.14/4.86	5.43/5.14/4.86/4.29
CoLA	**0.88/0.62/0.25/0.00**	3.38/3.37/3.37/3.12	1.12/0.88/1.12/1.37	1.00/0.88/1.12/1.37	2.62/2.50/2.50/2.37	2.50/2.50/2.38/2.25
QNLI	**1.00/0.60/0.30/0.00**	4.60/4.20/3.90/3.80	2.10/1.40/1.00/0.80	2.40/1.40/1.00/0.80	4.60/4.00/3.80/3.60	4.40/4.20/3.90/3.60
QQP	0.46/**0.24/0.12/0.00**	1.50/1.44/1.36/1.30	0.40/0.30/0.26/0.22	**0.38**/0.30/0.26/0.24	1.08/0.96/0.80/0.72	1.10/0.90/0.80/0.72
SST-2	**0.40/0.27/0.13/0.00**	6.80/6.40/6.27/6.40	0.40/0.40/0.40/0.40	0.40/0.40/0.40/0.40	5.87/5.60/5.60/5.47	5.73/5.47/5.33/5.33
WNLI	**3.08/3.08/1.54/0.00**	12.31/4.62/**1.54**/1.54	6.15/**3.08**/**1.54**/1.54	6.15/**3.08**/**1.54**/1.54	9.23/**3.08**/3.08/1.54	9.23/**3.08**/3.08/1.54
MNLI	**1.00/0.80/0.40/0.00**	4.70/4.30/3.90/2.90	1.20/1.20/1.10/1.00	1.10/1.10/1.10/1.00	4.40/4.00/3.20/2.70	4.00/4.30/3.70/2.90
RTE	**10.40/10.00/4.40/0.00**	22.40/21.20/16.40/11.20	21.20/16.80/17.20/6.80	20.80/16.80/19.20/7.20	22.80/20.80/16.80/10.80	22.00/19.20/14.80/10.80

Robustness Analysis

Model Selector (70%/80%/90%/100%)
1.90/0.80/0.40/0.00
1.40/0.90/0.50/0.00
1.30/**0.60/0.30/0.00**
1.40/1.10/0.40/0.00
11.33/8.27/5.87/**0.00**
0.90/0.90/0.80/0.00
0.80/0.50/0.20/0.00
1.20/0.70/0.10/0.00
1.00/0.50/0.20/0.00
0.90/0.40/0.30/0.00
1.14/1.14/0.29/0.00
0.88/0.62/0.25/0.00
1.00/0.60/0.30/0.00
0.46/**0.24/0.12/0.00**
0.40/0.27/0.13/0.00
3.08/3.08/1.54/0.00
1.00/0.80/0.40/0.00
10.40/10.00/4.40/0.00

Significantly smaller accuracy gaps compared to baseline methods (up to 9.8x times)

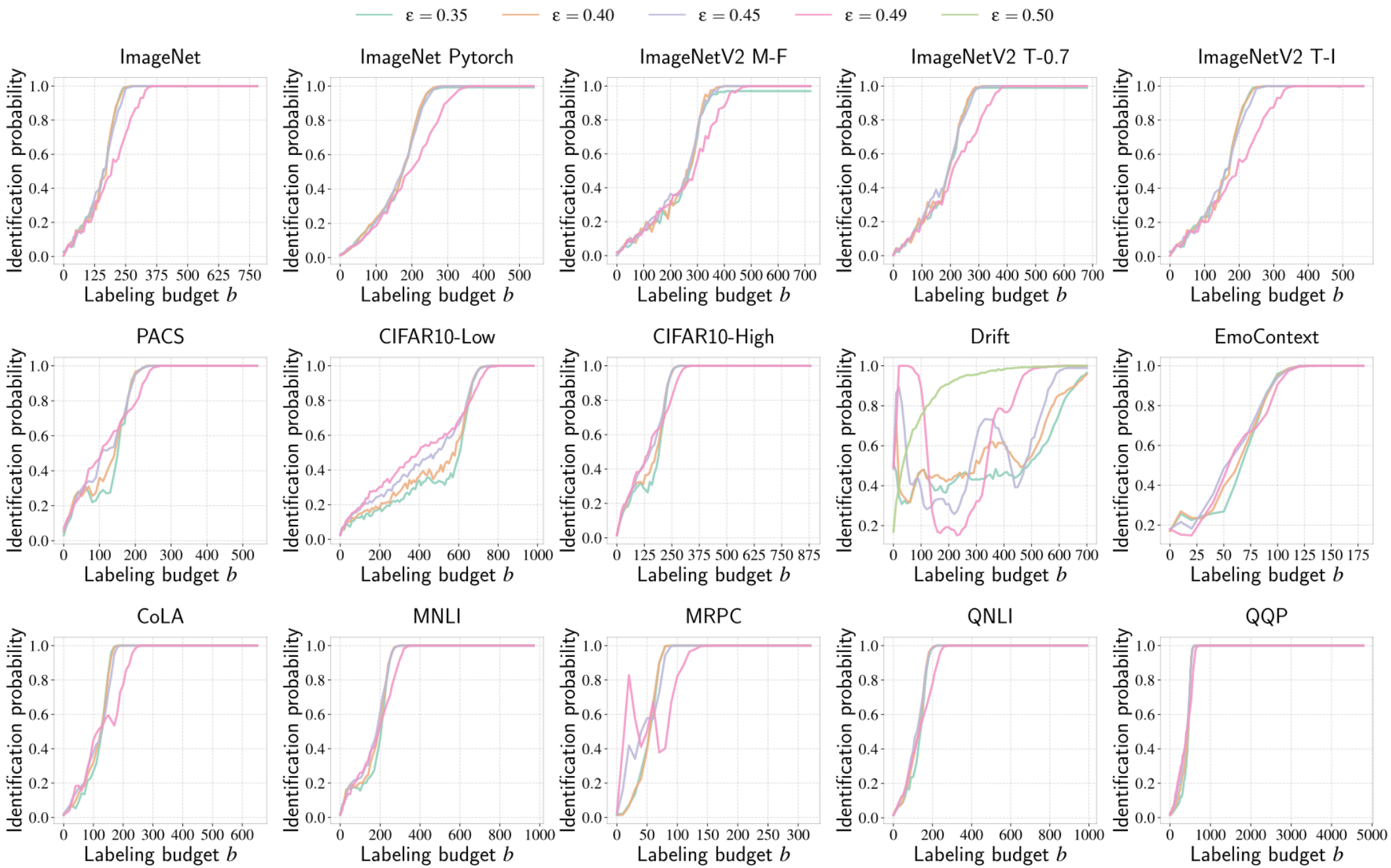
Robustness Analysis

Model Selector (70%/80%/90%/100%)
1.90/0.80/0.40/0.00
1.40/0.90/0.50/0.00
1.30/0.60/0.30/0.00
1.40/1.10/0.40/0.00
11.33/8.27/5.87/0.00
0.90/0.40/0.30/0.00
1.14/1.14/0.29/0.00
0.88/0.62/0.25/0.00
1.00/0.60/0.30/0.00
0.46/0.24/0.12/0.00
0.40/0.27/0.13/0.00
3.08/3.08/1.54/0.00
1.00/0.80/0.40/0.00
10.40/10.00/4.40/0.00

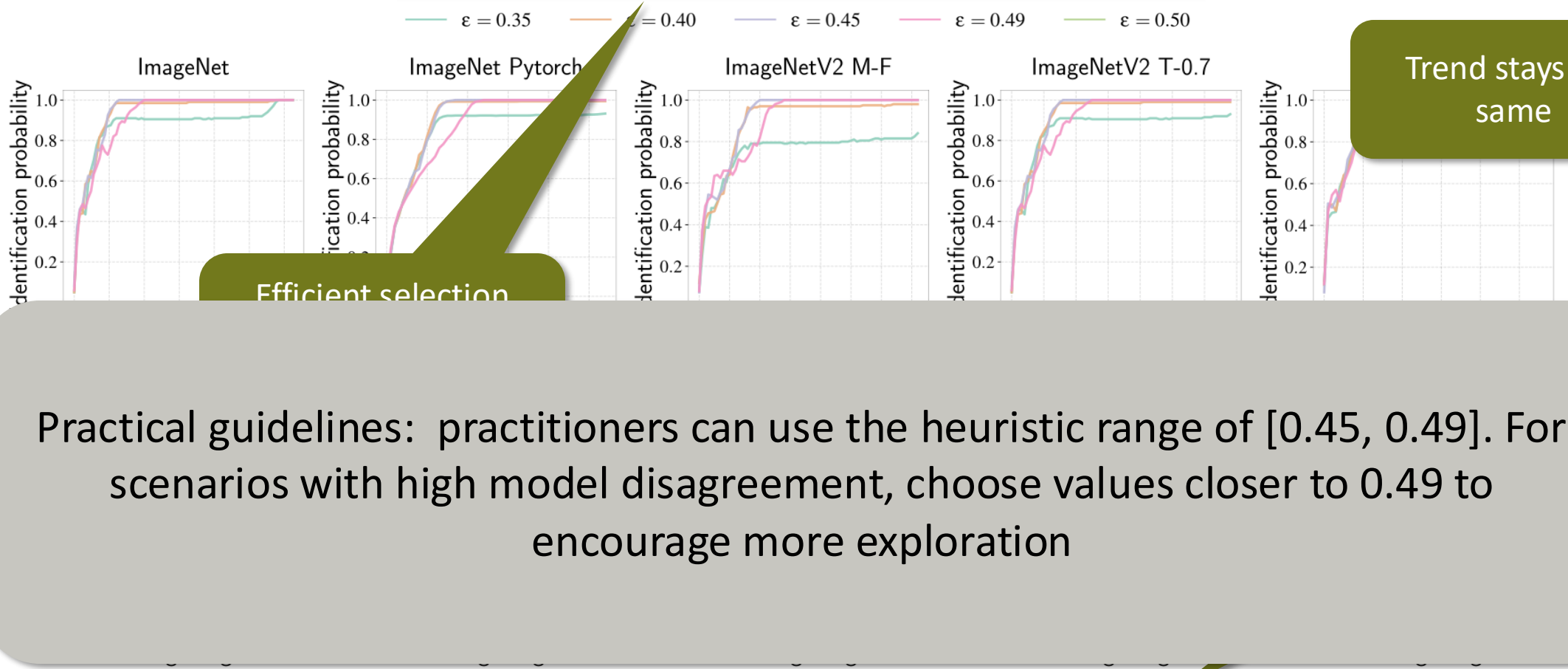
Significantly smaller accuracy gaps compared to baseline methods (up

Model Selector is more robust compared to baseline methods

Choosing ε with Noisy Labels



Choosing ε with Oracle Labels



Can We Use Noisy Labels to Identify the Best Model?

acc. on true labels – acc. on noisy labels

Dataset	Best model accuracy gap
CIFAR10-High	4.28%
CIFAR10-Low	2.64%
EmoContext	0.96%
PACS	1.56%
Drift	13.78%
ImageNet	3.49%
ImageNet Pytorch	4.47%
ImageNetV2 T-I	4.53%
ImageNetV2 T-0.7	5.72%
ImageNetV2 M-F	7.27%
MRPC	1.29%
CoLA	5.22%
QNLI	3.25%
QQP	1.08%
SST-2	3.93%
WNLI	3.49%
MNLI	3.48%
RTE	15.93%

Best model accuracy gap > 0

Can We Use Noisy Labels to Identify the Best Model?

Dataset	Best model accuracy gap
CIFAR10-High	4.28%
CIFAR10-Low	2.64%
EmoContext	0.96%
PACS	1.56%
Drift	13.78%
ImageNet	3.49%
ImageNetV2 M-F	7.27%
MRPC	1.29%
CoLA	5.22%
QNLI	3.25%
QQP	1.08%
SST-2	3.93%
WNLI	3.49%
MNLI	3.48%
RTE	15.93%

We cannot use noisy labels to identify the best model!

Conclusions

More of SPCL's research:

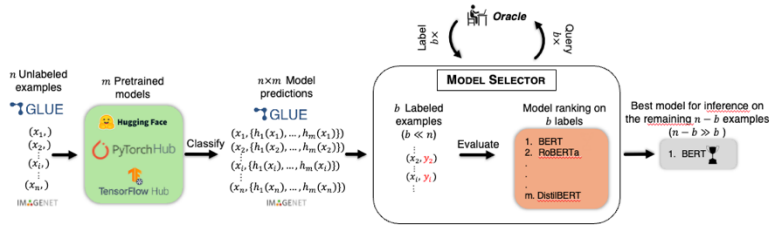
youtube.com/@spcl — 210+ Talks

twitter.com/spcl_eth — 1.6K+ Followers

github.com/spcl — 5.6K+ Stars

… or spcl.ethz.ch

Framework



Performance Metrics

Identification Probability

- Fraction of realizations where a method successfully identifies the true best model of that realization

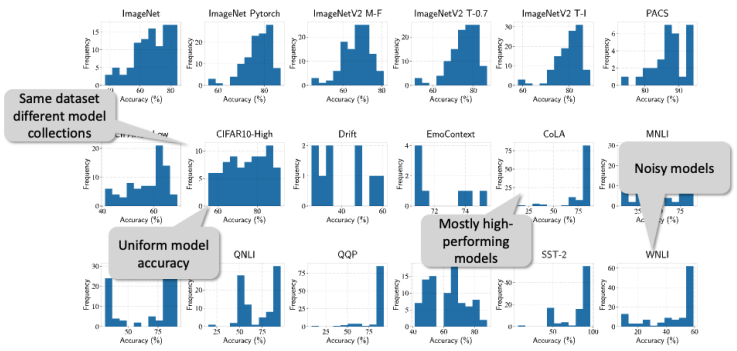
Label Efficiency

- The reduction in number of labels (%) required to select the best or a near-best (δ) model over all realizations

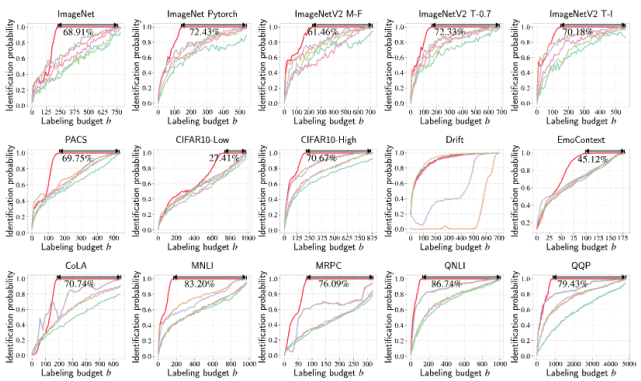
95thPercentile Accuracy Gap

- Represents the 95th percentile of the accuracy gap across all realizations

Datasets and Model Collections



Best Model Identification Probability



Paper:

Code: