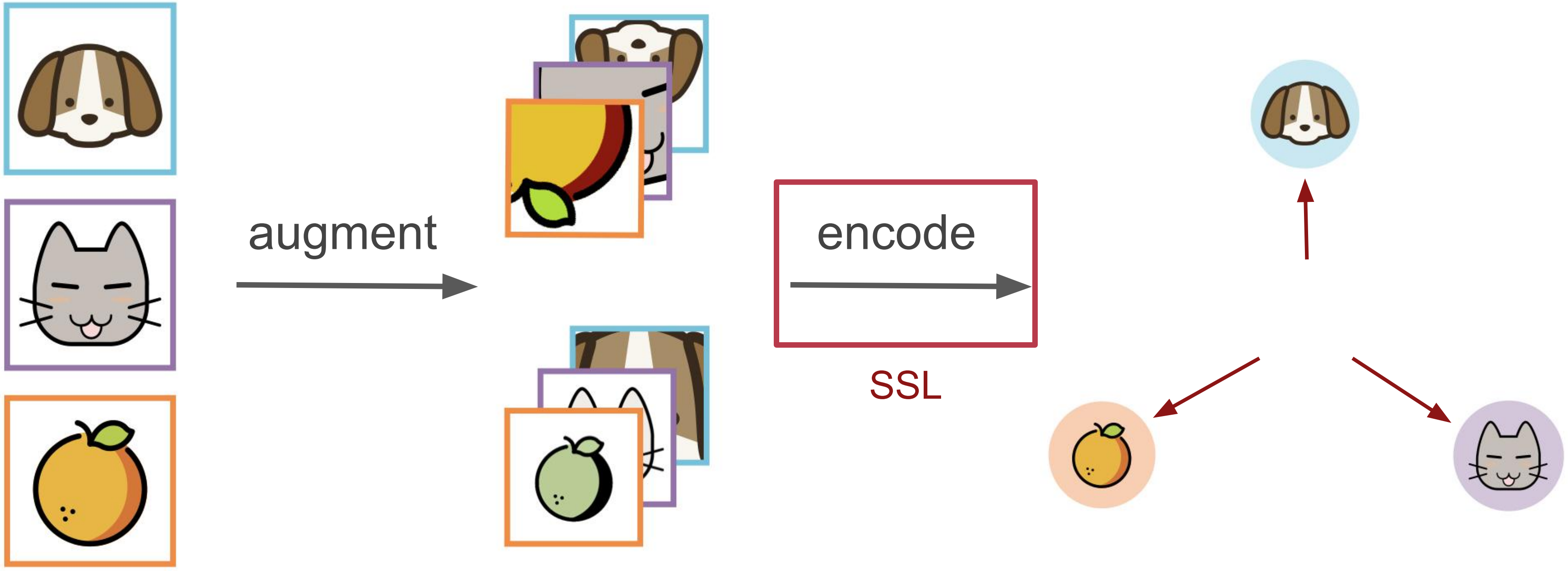
Evaluating Self-Supervised Learning via Risk Decomposition

Yann Dubois, Tatsunori Hashimoto, Percy Liang

ICML 2023 Oral

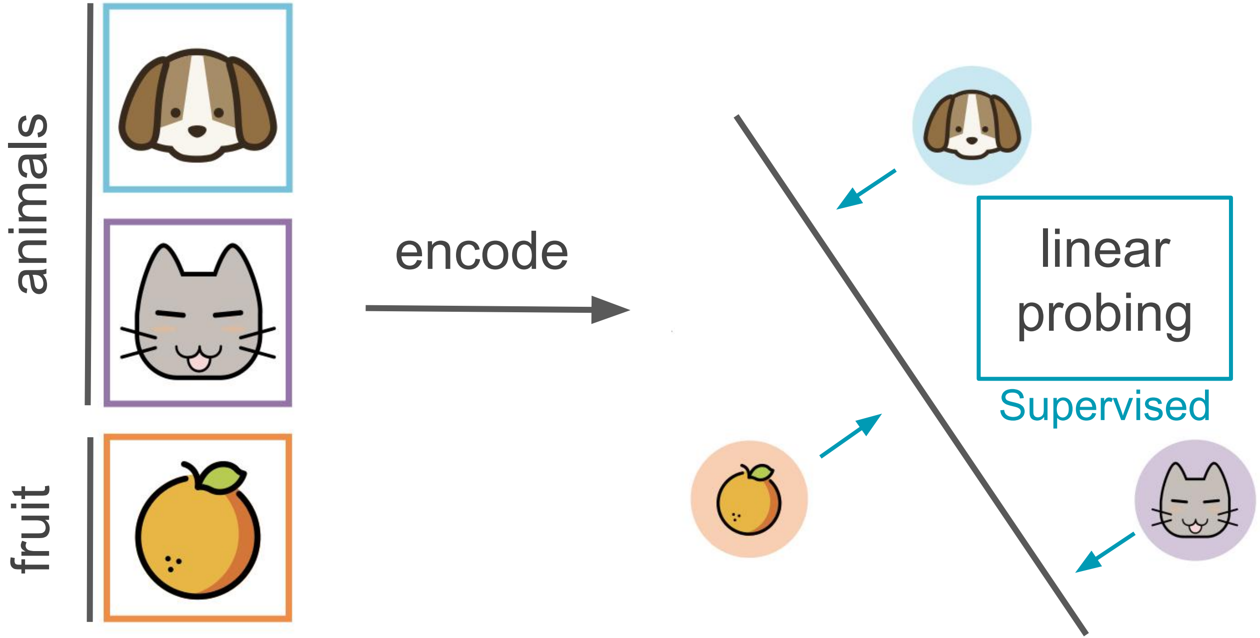
Background: pretraining in SSL

- E.g. train on unlabeled ImageNet using SSL



Background: linear probing in SSL

- E.g. train on labeled ImageNet using supervised learning



Motivation: evaluating SSL

- There are many SSL pipelines that get proposed, with different:

  SSL objectives   architectures   optimizers   pretraining data

- Evaluated on a single metric: linear probing on ImageNet.

Motivation: evaluating SSL

- There are many SSL pipelines that get proposed, with different:

SSL objectives architectures optimizers pretraining data

- Evaluated on a single metric: linear probing on ImageNet.

✗ why is an SSL pipeline better?

✗ when is an SSL pipeline better?

✗ how to improve the SSL pipeline?

Motivation: supervised risk decomposition

- **Supervised learning** monitor training/validation loss
    - underfitting ⇒ increase capacity
    - overfitting ⇒ regularize
    - small training loss: model would be better with large datasets
    - …

Motivation: supervised risk decomposition

- **Supervised learning** monitor training/validation loss
  - underfitting ⇒ increase capacity
  - overfitting ⇒ regularize
  - small training loss: model would be better with large datasets
  - …

$$\boxed{\text{Risk} = \underbrace{\text{approximation error}}_{\sim\text{training error}} + \underbrace{\text{generalization error}}_{\sim\text{training - validation error}}}$$

Motivation: supervised risk decomposition

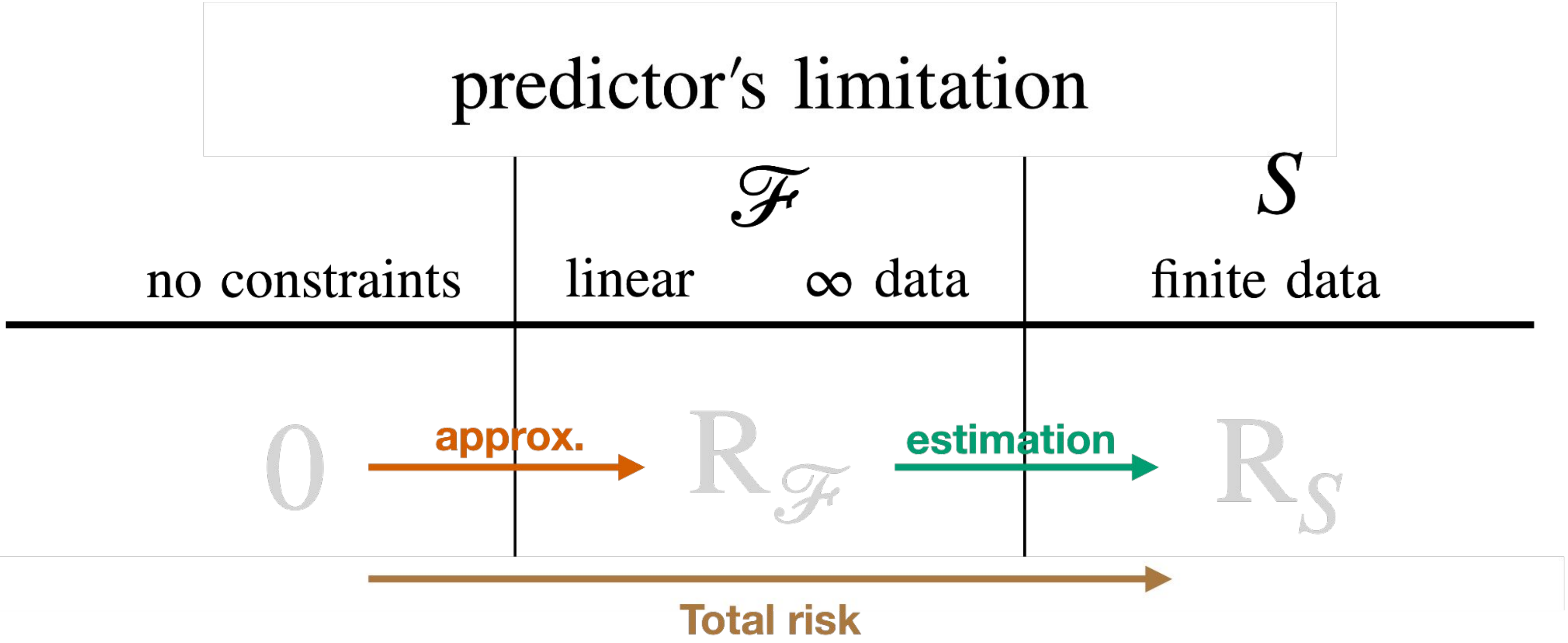
- **Supervised learning** monitor training/validation loss
  - underfitting ⇒ increase capacity
  - overfitting ⇒ regularize
  - small training loss: model would be better with large datasets
  - …
- **Self-supervised learning**
  - ?

Motivation: supervised risk decomposition

- **Supervised learning** monitor training/validation loss
  - underfitting ⇒ increase capacity
  - overfitting ⇒ regularize
  - small training loss: model would be better with large datasets
  - …
- **Self-supervised learning**
  - ?

Idea: generalize risk decomposition to SSL and estimate it

Motivation: supervised risk decomposition



Proposed: SSL risk decomposition

probe

encoder

0	$\mathscr{F}$ linear ∞ data	S finite data
Φ supervised ∞ data		
A_Φ self – supervised ∞ data		
U self – supervised finite data		

Proposed: SSL risk decomposition

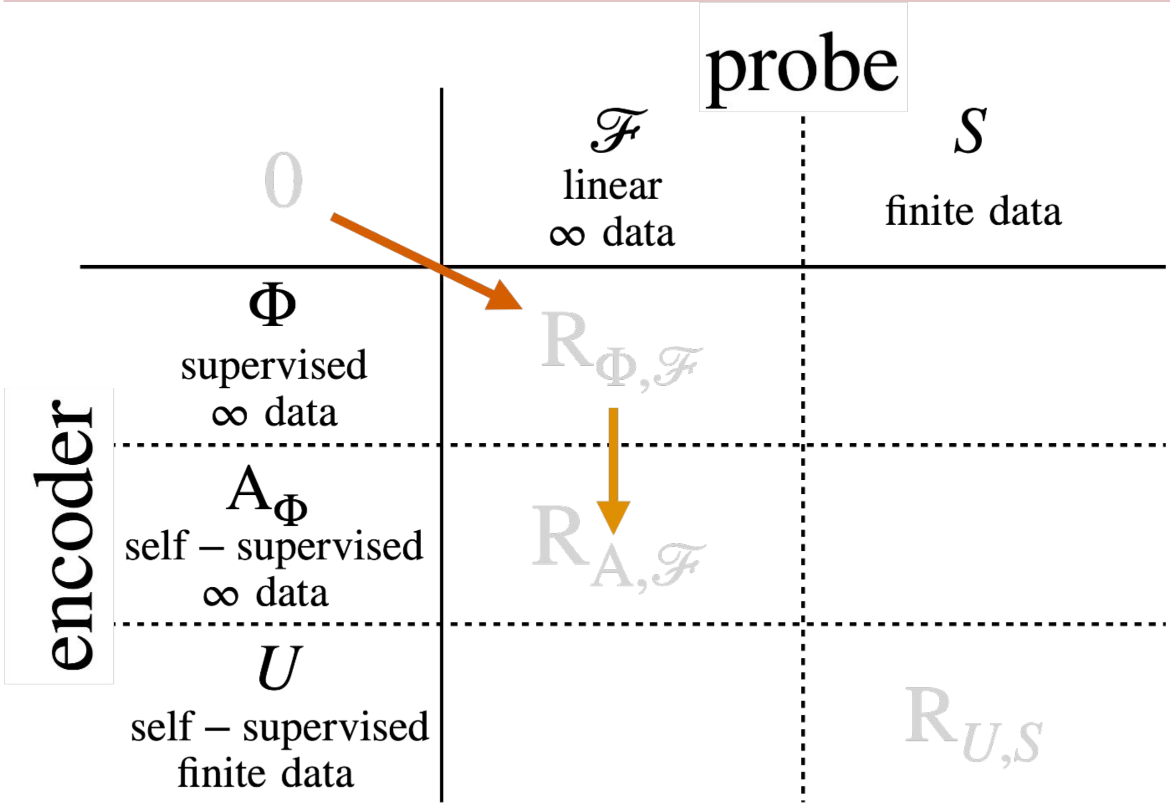
probe

encoder	$\mathcal{F}$ linear ∞ data	S finite data
	0	
Φ supervised ∞ data	$\mathrm{R}_{\Phi,\mathcal{F}}$	
A_Φ self – supervised ∞ data		
U self – supervised finite data		$\mathrm{R}_{U,S}$

approximation error

architectural constraints

Proposed: SSL risk decomposition

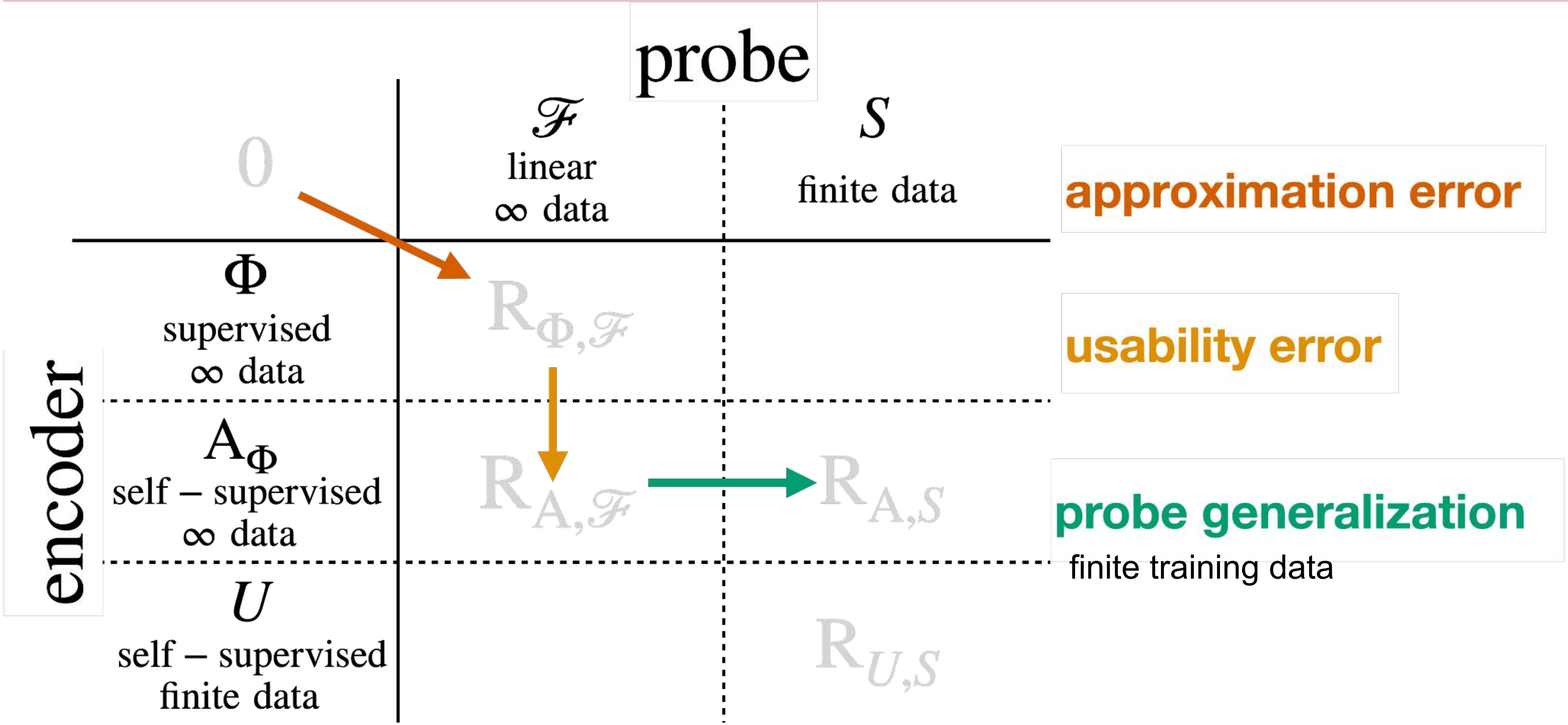


approximation error

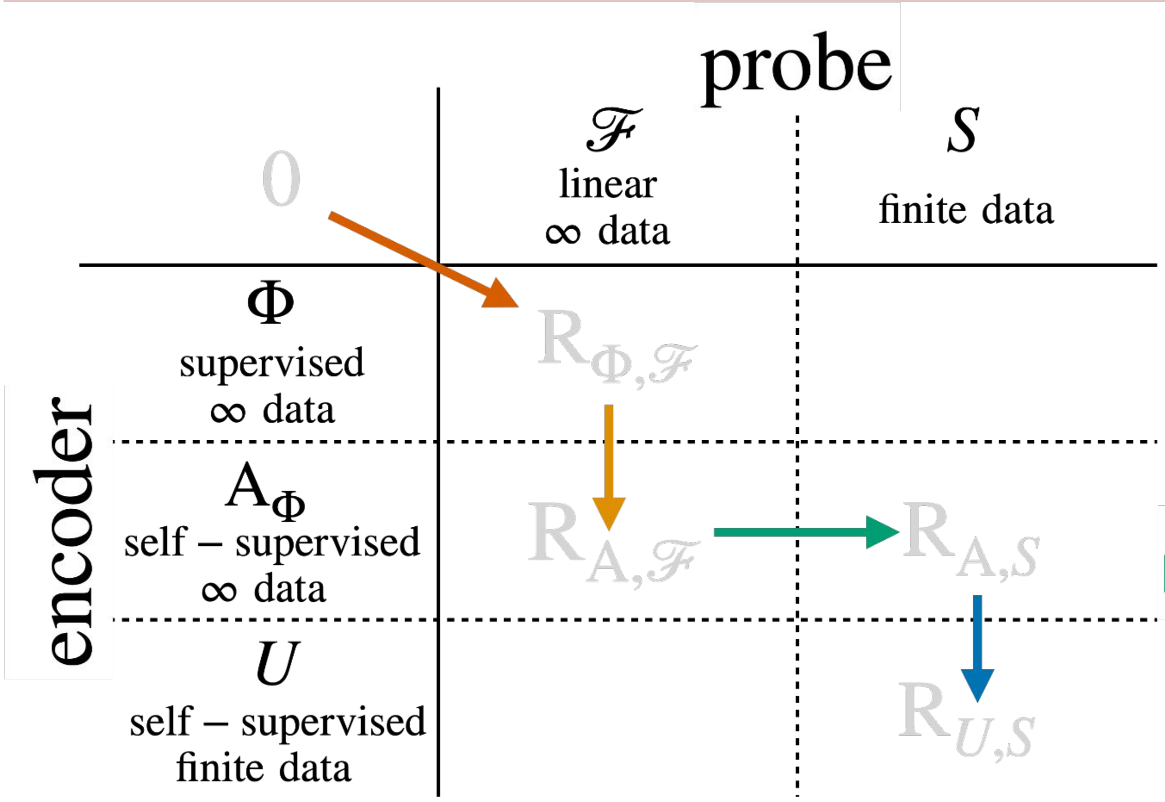
usability error

use of SSL instead of supervised learning => representations might not be linearly separable

Proposed: SSL risk decomposition



Proposed: SSL risk decomposition



approximation error

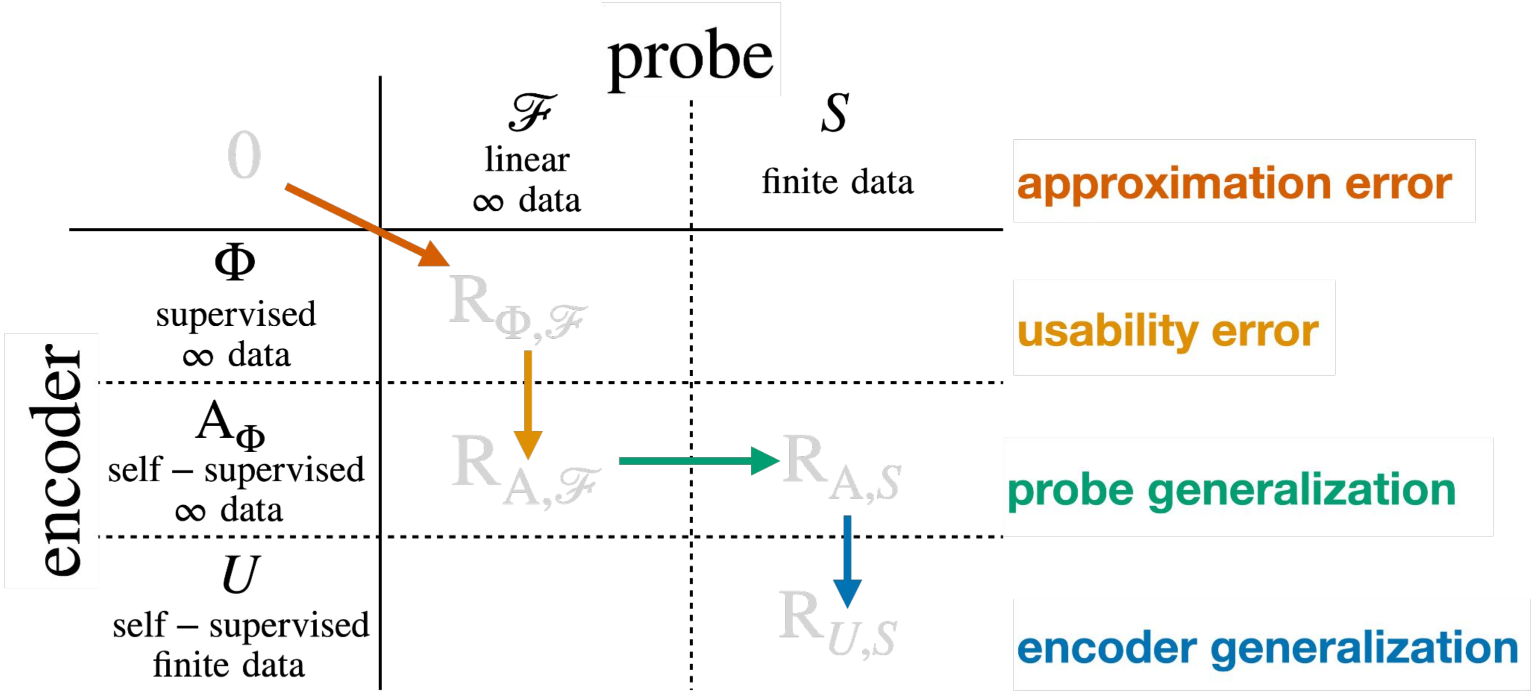
usability error

probe generalization

encoder generalization

finite pre-training data

Proposed: SSL risk decomposition



We provide efficient estimators for each component!

Experiments: supervised risk decomposition

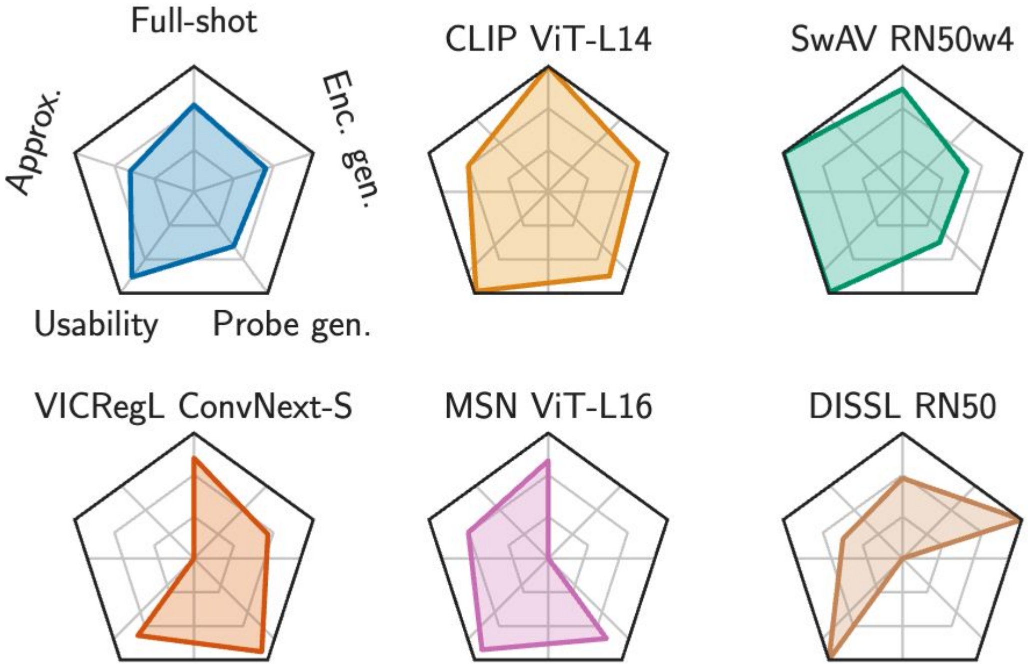
Broad evaluation of SSL methods:

169 pretrained encoders, 28 objectives, 20 arch., 7 years

Benchmark:

- linear probes on ImageNet (100%, 30-shot, 1%, 5-shot, 3-shot)
- estimate each error component

Results: no model is uniformly better



no model is uniformly better!

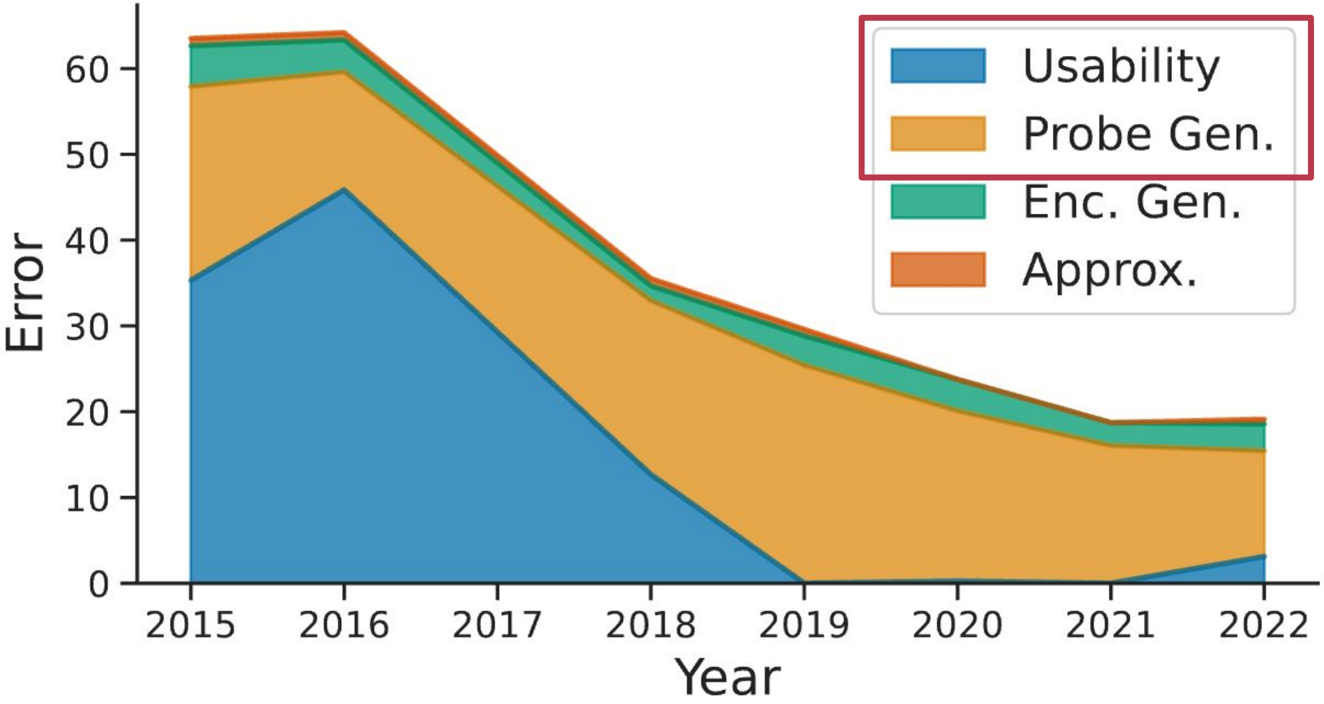
Results: Full- vs Few-shot Tradeoff

			ImageNet probe acc.		
Obj.	Arch.	Param.	100%	1%	3-shot
MoCo-v3	RN50	24M	73.7	55.5	40.4
DINO	RN50	24M	74.2	52.9	35.9
SwAV	RN50w4	375M	76.2	56.2	36.9
VICRegL	CnvNxt-B	85M	74.8	64.3	56.3
MUGS	ViT-S16	22M	77.3	62.9	49.6
MSN	ViT-S16	22M	76.1	67.5	60.4
MSN	ViT-B4	86M	80.1	75.1	69.3
MUGS	ViT-L16	303M	80.9	74.0	68.5
MSN	ViT-L7	303M	79.9	74.9	**69.8**
CLIP	ViT-L14	304M	**85.0**	75.2	62.9
OpenCLIP	ViT-H14	632M	84.4	**75.8**	63.7

the best model in full-shot is always different than in few-shot

Results: risk components over time

Usability → probe gen.



Results: implication for SSL method design

	# dim. ↓	# views ↑	ViT	# param.↑	MLP proj.	generative SSL	# epoch ↑	Adam
Usability error	↑	↓		↓	↓	↑	↓	
Probe gen. error	↓	↓	↓		↓	↓	↓	↓
Full-shot error	↑	↓	↓	↓	↓	↑	↓	↓
3-shot error	↓	↓	↓	↓	↓	↓	↓	↓

Results: dimensionality

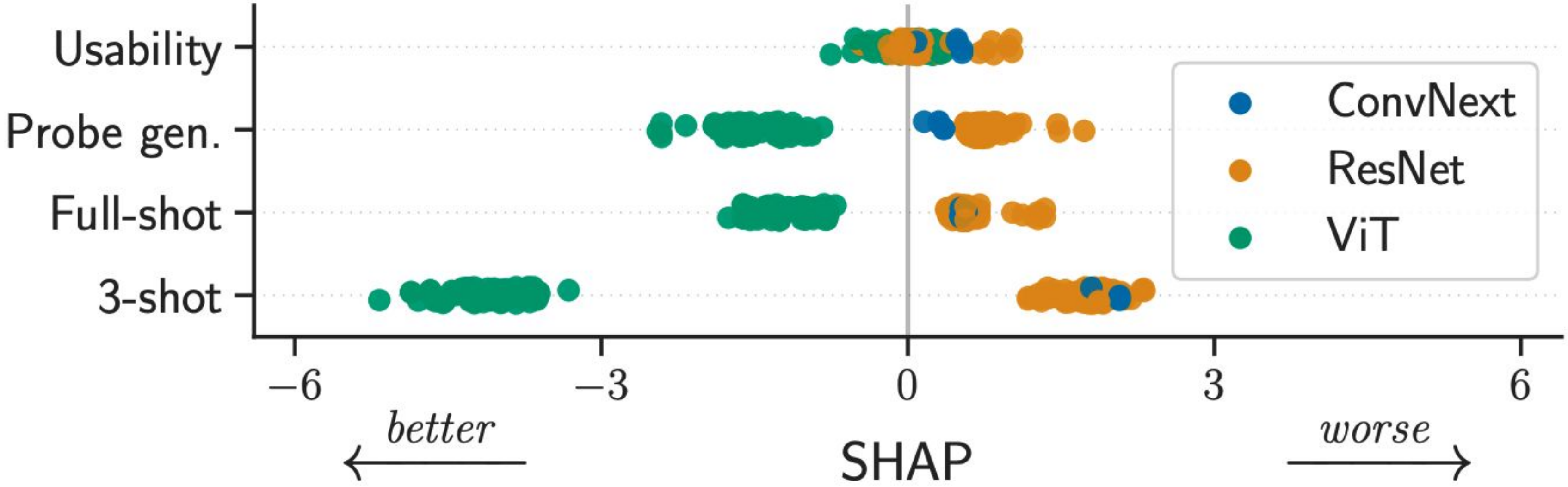
Some design choices (e.g. dimensionality) can control U-P tradeoff => full- vs few-shot

Ours	Obj.	ViT	Dim.	100%	1%	3-shot
✗	MUGS	S16	1536	**77.3**	62.9	49.6
✓	MUGS	S16	384	77.0	**66.6**	**57.9**
✗	OpenCLIP	H14	1280	**84.4**	75.8	63.7
✓	OpenCLIP	H14	1024	84.3	**76.5**	**65.5**

by decreasing dimensionality we can greatly improve few shot performance without any retraining!

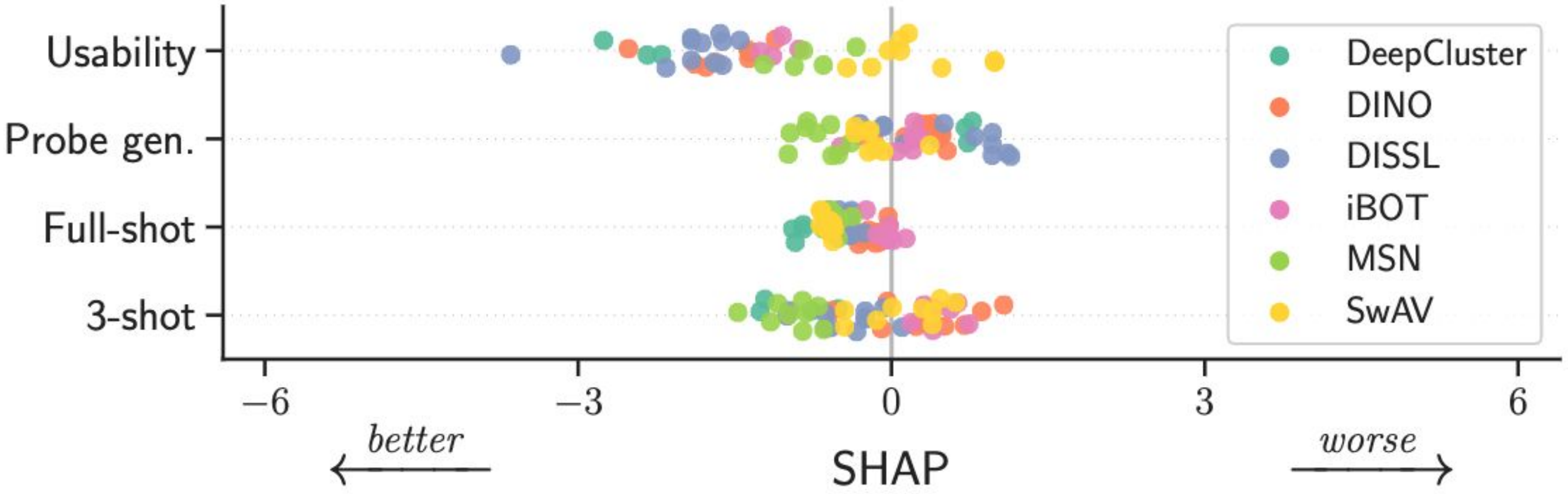
Results: architecture

Other design choices (e.g. architecture) overcome the tradeoff => uniform improvement



Results: exact objective

Other design choices (e.g. the exact objective in generative or contrastive) don't matter when controlling for confounders!



Summary

- New risk decomposition for SSL with efficient estimators

$$\underbrace{\mathrm{R}_{U,S} - \mathrm{R}_*}_{\text{excess risk}} = \underbrace{\mathrm{R}_{U,S} - \mathrm{R}_{A,S}}_{\text{encoder generalization}} + \underbrace{\mathrm{R}_{A,S} - \mathrm{R}_{A,\mathcal{F}}}_{\text{probe generalization}} + \underbrace{\mathrm{R}_{A,\mathcal{F}} - \mathrm{R}_{\Phi,\mathcal{F}}}_{\text{representation usability}} + \underbrace{\mathrm{R}_{\Phi,\mathcal{F}} - \mathrm{R}_*}_{\text{approximation}}$$

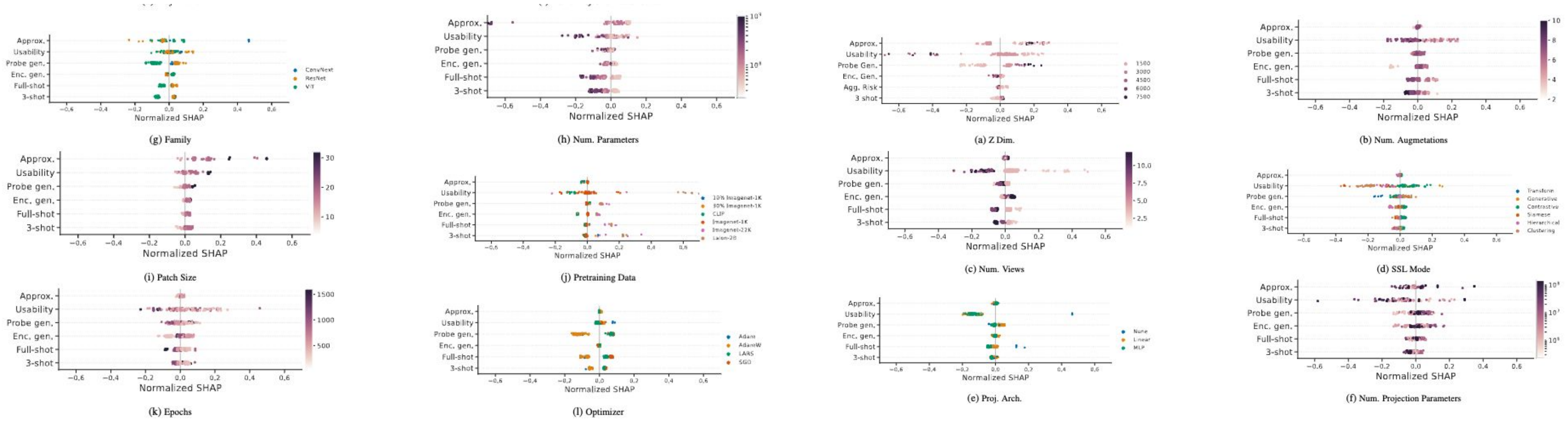
Algorithm 1 Estimating risk components in the standard SSL setting

Require: Encoder family Φ, probe family $\mathcal{F}$, training S_{tr} and testing S_{te} sets, SSL algorithm A_Φ, evaluation loss ℓ.

function $\text{RISK}(\mathcal{F}, \mathcal{D}_{tr}, \mathcal{D}_{te})$
    $\hat{f} \leftarrow \inf_{f \in \mathcal{F}} \sum_{(x,y)\in\mathcal{D}_{tr}} \ell(y, f(x))$ ▷ Risk minimization
    return $\frac{1}{|\mathcal{D}_{te}|} \sum_{(x,y)\in\mathcal{D}_{te}} \ell(y, \hat{f}(x))$ ▷ Test risk
$\hat{\mathrm{R}}_{\Phi,\mathcal{F}} \leftarrow \text{RISK}(\Phi \circ \mathcal{F}, S_{\text{tr}}, S_{\text{tr}})$ ▷ Supervised train performance
$\phi \leftarrow \mathrm{A}_\Phi(\Phi, S_{\text{tr}})$ ▷ Pretrain SSL encoder
$S^\phi_{\text{tr}} \leftarrow [(\phi(\text{x}),\text{y})$ **for** x,y **in** $S_{\text{tr}}]$ ▷ Featurize data
$S^\phi_{\text{te}} \leftarrow [(\phi(\text{x}),\text{y})$ **for** x,y **in** $S_{\text{te}}]$
$S^\phi_{\text{sub}} \leftarrow \text{subset}(S^\phi_{\text{tr}}, n = \text{len}(S_{\text{te}}))$
$\hat{\mathrm{R}}_{A,\mathcal{F}} \leftarrow \text{RISK}(\mathcal{F}, S^\phi_{\text{tr}}, S^\phi_{\text{tr}})$ ▷ Risk without generalization
$\hat{\mathrm{R}}_{A,S} \leftarrow \text{RISK}(\mathcal{F}, S^\phi_{\text{tr}} \setminus S^\phi_{\text{sub}}, S^\phi_{\text{sub}})$ ▷ Risk with only probe gen.
$\hat{\mathrm{R}}_{U,S} \leftarrow \text{RISK}(\mathcal{F}, S^\phi_{\text{tr}}, S^\phi_{\text{te}})$ ▷ Risk with enc. and probe gen.
approx_error $\leftarrow \hat{\mathrm{R}}_{\Phi,\mathcal{F}}$
usability_error $\leftarrow \hat{\mathrm{R}}_{A,\mathcal{F}} - \hat{\mathrm{R}}_{\Phi,\mathcal{F}}$
probe_gen $\leftarrow \hat{\mathrm{R}}_{A,S} - \hat{\mathrm{R}}_{A,\mathcal{F}}$
encoder_gen $\leftarrow \hat{\mathrm{R}}_{U,S} - \hat{\mathrm{R}}_{A,S}$
return approx_error, usability_error, probe_gen, encoder_gen

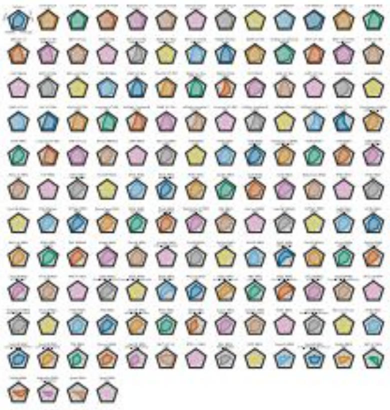
Summary

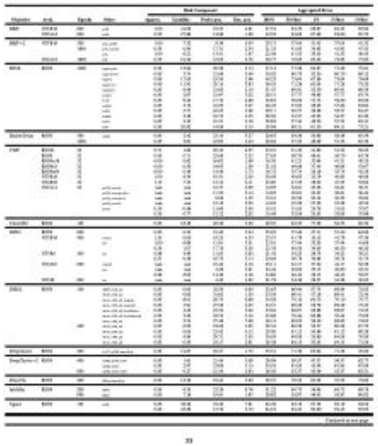
- New risk decomposition for SSL with efficient estimators
- Meta-analysis of 169 models and 30 design choices

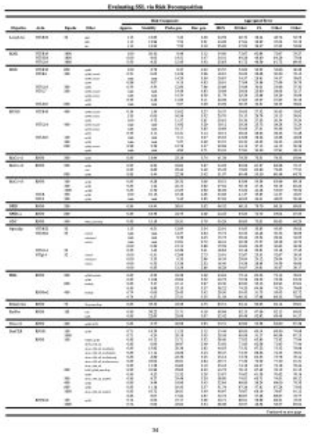


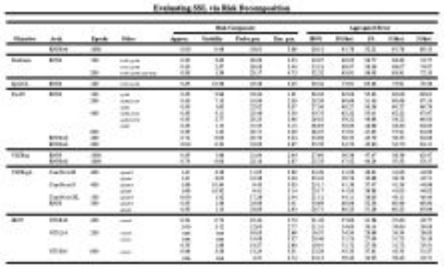
Summary

- New risk decomposition for SSL with efficient estimators
- Meta-analysis of 169 models and 30 design choices
- Many more results in the paper!
  - Thorough analysis of each design choice
  - Large scale evaluation of SSL with different metrics









Summary

- New risk decomposition for SSL with efficient estimators
- Meta-analysis of 169 models and 30 design choices
- Many more results in the paper!
- Torch Hub API & code to access any models or metadata in one line

SSL-Risk-Decomposition

```
import torch

# loads the desired pretrained model and preprocessing pipeline
name = "dino_rn50" # example
model, preprocessor = torch.hub.load('YannDubs/SSL-Risk-Decomposition:main', name, trust_repo=True)

# gets all available models
available_names = torch.hub.list('YannDubs/SSL-Risk-Decomposition:main')

# gets all results and hyperparameters as a dataframe
results_df = torch.hub.load('YannDubs/SSL-Risk-Decomposition:main', "results_df")
```