SURI Presentation

Robustness of a Tunable Loss Function Family on Corrupted Datasets

Kyle Otstot | Sankar Lab | August 9th, 2021

Agenda

- Brief introduction
- Background
  - Image classification
  - Robust loss functions
  - Data augmentation
- Empirical investigation
  - Motivation
  - Setting (control & variable)
  - Hyperparameter tuning
  - Results summary
- Discussion
  - Estimated distribution metrics
  - Backpropagation
- Takeaways
- Next step
- Q&A

Agenda

- Brief introduction
- Background
  - Image classification
  - Robust loss functions
  - Data augmentation
- Empirical investigation
  - Motivation
  - Setting (control & variable)
  - Hyperparameter tuning
  - Results summary
- Discussion
  - Estimated distribution metrics
  - Backpropagation
- Takeaways
- Next step
- Q&A

My SURI Experience

Brief Introduction

- Hello, I’m Kyle Otstot!
- Incoming 3rd year CS & Math major at ASU
- Interned under Dr. Lalitha Sankar this summer
    - Worked with John Cava
    - Supervised by Lalitha Sankar, Chaowei Xiao, and Tyler Sypherd

Project Overview
Brief Introduction

- In this project, we set out to
  1. Introduce a classification setting applicable to real-world problems
     - Argue that training and evaluating on corrupted datasets are inevitable obstacles
  2. Propose a novel task that formulates a robust solution to the classification setting
  3. Evaluate the performance of α-loss relative to some selected SoTA robust loss function family in the context of this novel task

Agenda

- Brief introduction
- Background
    - Image classification
    - Robust loss functions
    - Data augmentation
- Empirical investigation
    - Motivation
    - Setting (control & variable)
    - Hyperparameter tuning
    - Results summary
- Discussion
    - Estimated distribution metrics
    - Backpropagation
- Takeaways
- Next step
- Q&A

Image Classification

Background

- Each image classification problem is defined by the following:
  - Feature space $\mathscr{X}$, Label space $\mathscr{Y}$
  - "Ground-truth" mapping $f : \mathscr{X} \rightarrow \mathscr{Y}$
    - $\forall_{x \in \mathscr{X}, y \in \mathscr{Y}} \quad y = f(x)$ iff x is an image of y
  - Observable dataset $\mathscr{D} \subset \{(x, y) \in \mathscr{X} \times \mathscr{Y} : y = f(x)\}$
  - Classifier $\hat{f} : \mathscr{X} \rightarrow \mathscr{P}$, an estimate of f , given $\mathscr{D}$

Image Classification

Background

x

( , Red panda) → Classifier ($\hat{f}$) →

y

$\in \mathscr{D}$

$\hat{P}$		P
0.45	Fox	0
0.02	Cow	0
0.23	Red panda	1
0.25	Bear	0
0.05	Dolphin	0

$\mathscr{Y}$

Image Classification

Background

- Dataset Corruptions
  - Labels

( , Red panda) → ( , Dolphin)

  - Features

( , Red panda) → ( , Red panda)

Image Classification

Background

- Proposed remedies to corrupted datasets
  - Labels
    - Robust loss functions (NEXT)
  - Features
    - Data augmentation (AFTER)

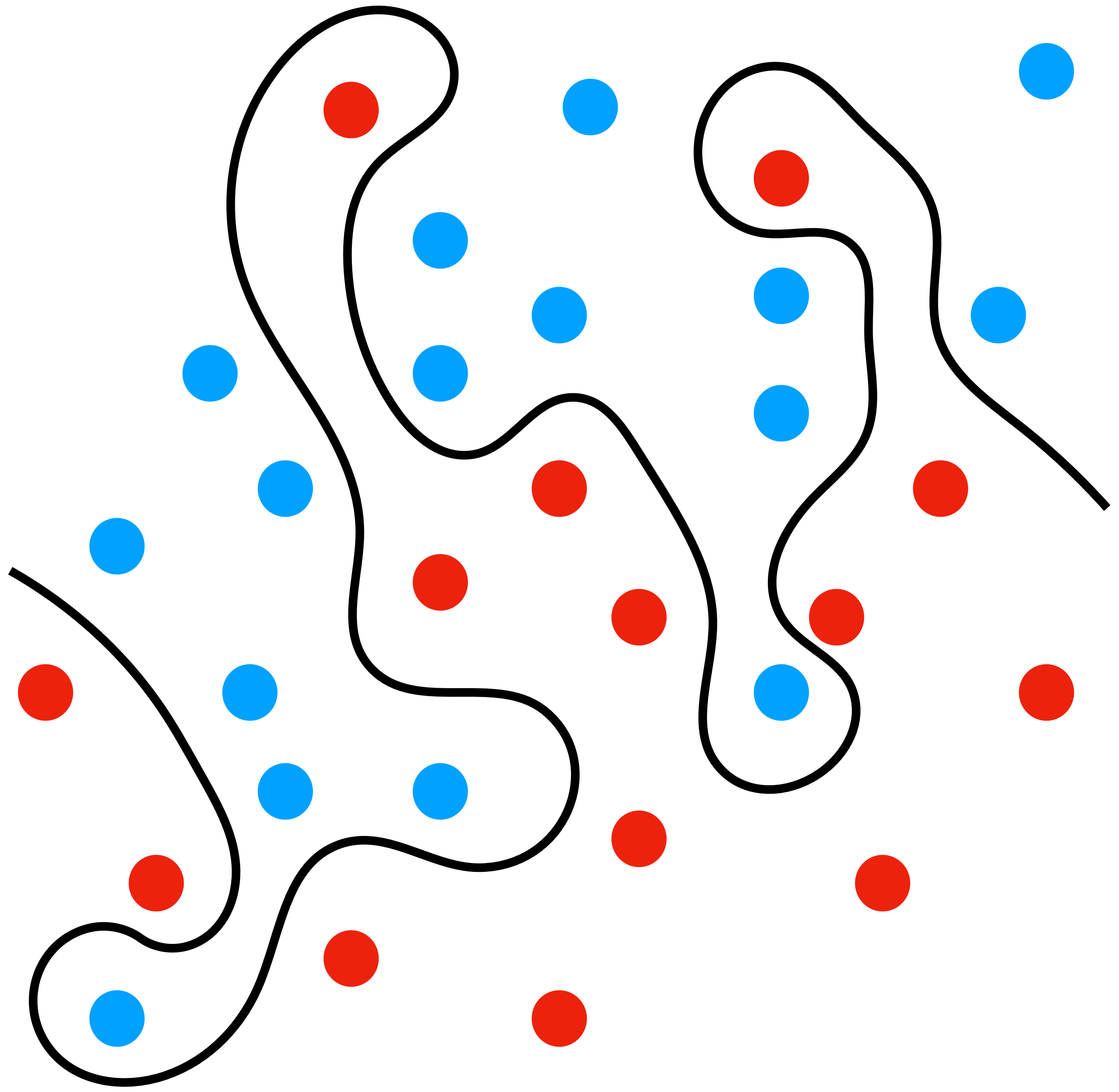
Agenda

- Brief introduction
- Background
  - Image classification
  - Robust loss functions
  - Data augmentation
- Empirical investigation
  - Motivation
  - Setting (control & variable)
  - Hyperparameter tuning
  - Results summary
- Discussion
  - Estimated distribution metrics
  - Backpropagation
- Takeaways
- Next step
- Q&A

Robust Loss Functions

Background

- General loss function $\ell : \mathscr{P} \times \mathscr{Y} \to \mathbb{R}$
  - Maps an estimated probability distribution and true class label to a real number
- Standard loss function: *Cross Entropy*
  - $\ell_{CE}(\hat{P}, y) = -\log \hat{P}(y)$
  - Fails to be robust to label noise

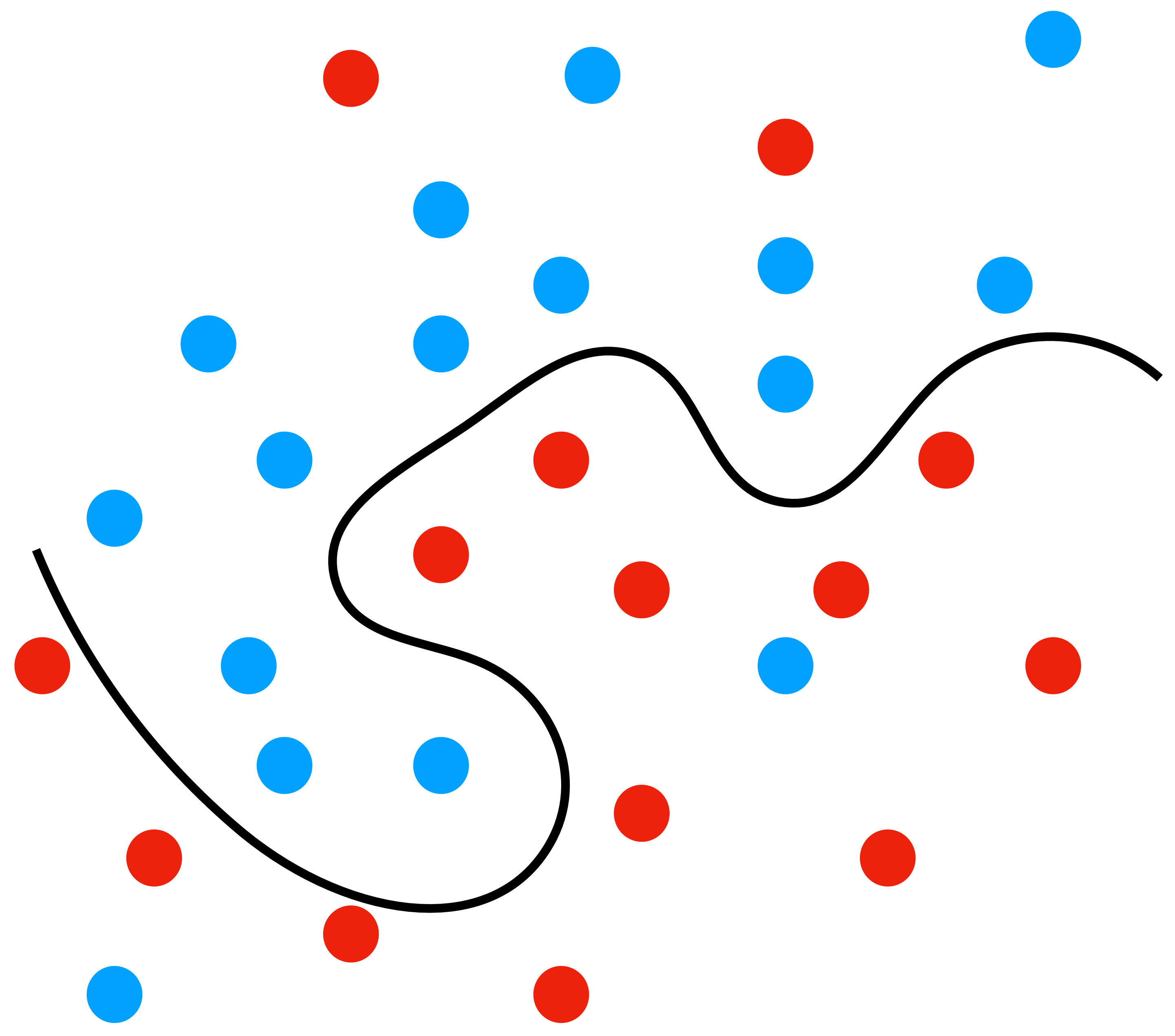


Overfitting to noisy label set

Robust Loss Functions

Background

- SoTA loss function family: *NCE+RCE*
  - $\ell_{+}(\hat{P}, y; \alpha, \beta) = \alpha \cdot \ell_{NCE}(\hat{P}, y) + \beta \cdot \ell_{RCE}(\hat{P}, y)$
  - Will define ℓ_{NCE} and ℓ_{RCE} later
  - Shown to be robust to label noise *(Ma, et al.)*

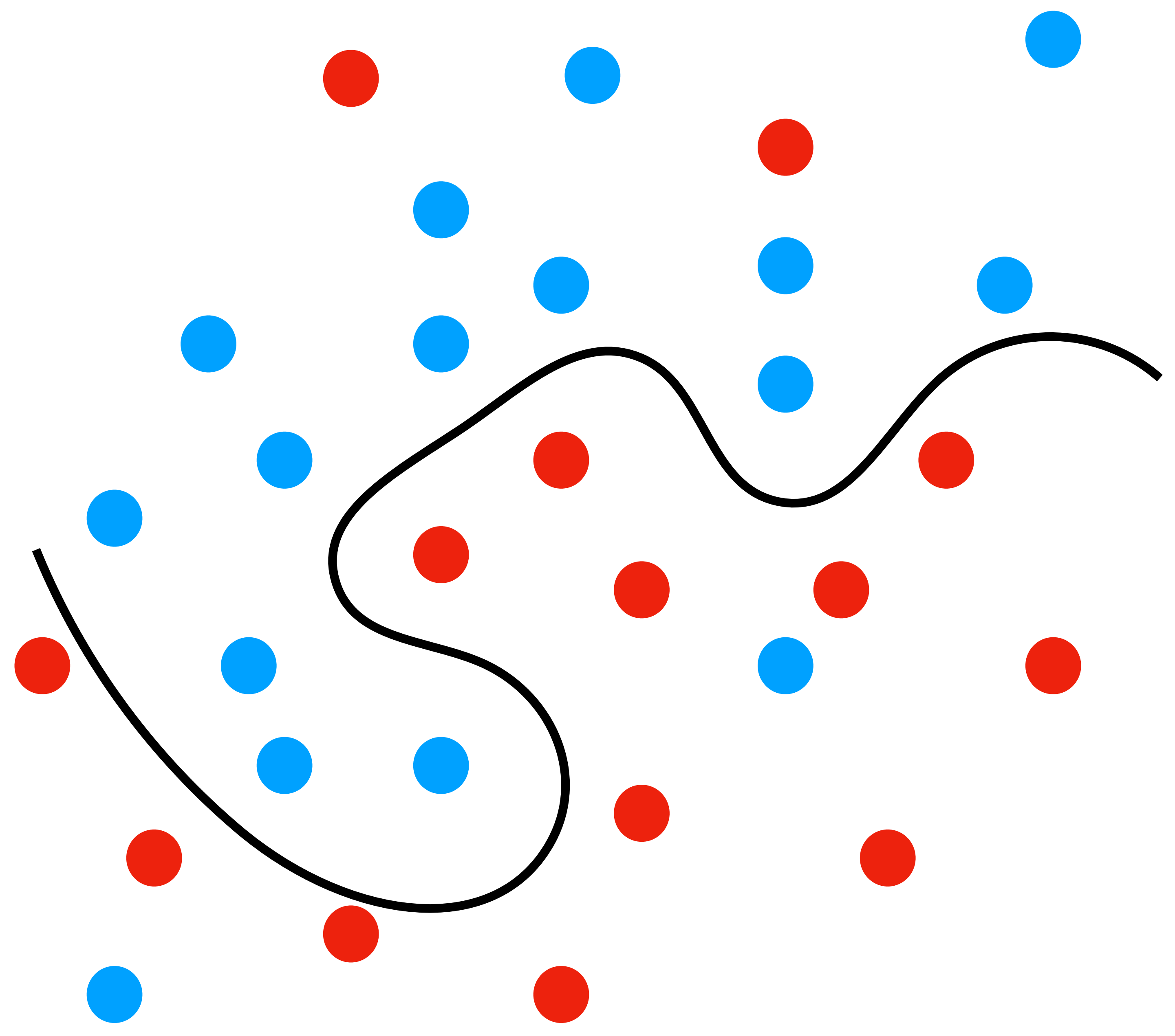


Generalizing to true label set

Robust Loss Functions

Background

- Featured loss function family: *α-loss*
  - $\ell_\alpha(\hat{P}, y; \alpha) = \frac{\alpha}{\alpha - 1}\left(1 - \hat{P}(y)^{1-\frac{1}{\alpha}}\right)$
  - Shown to be robust to label noise when $\alpha > 1$
  - $\ell_\alpha = \ell_{CE}$ when $\alpha = 1$



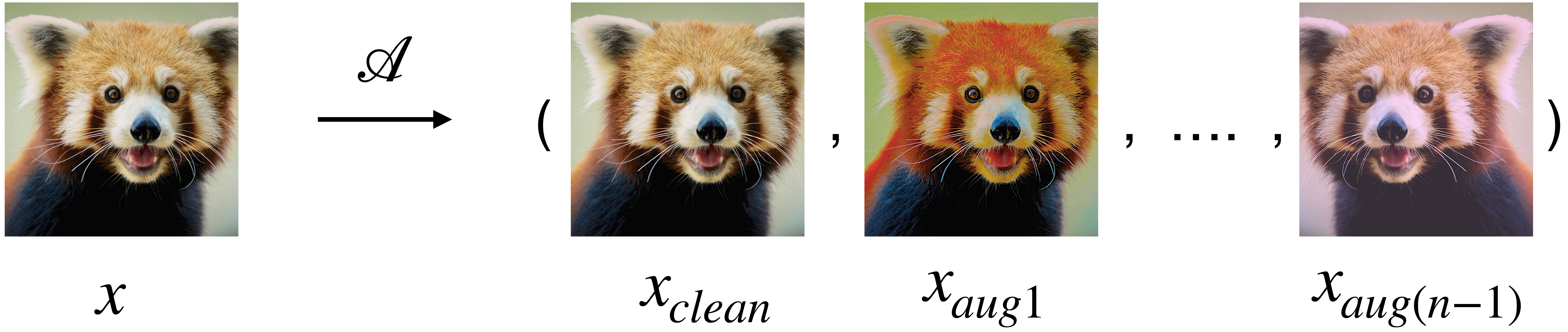
Generalizing to true label set

Agenda

- Brief introduction
- Background
    - Image classification
    - Robust loss functions
    - Data augmentation
- Empirical investigation
    - Motivation
    - Setting (control & variable)
    - Hyperparameter tuning
    - Results summary
- Discussion
    - Estimated distribution metrics
    - Backpropagation
- Takeaways
- Next step
- Q&A

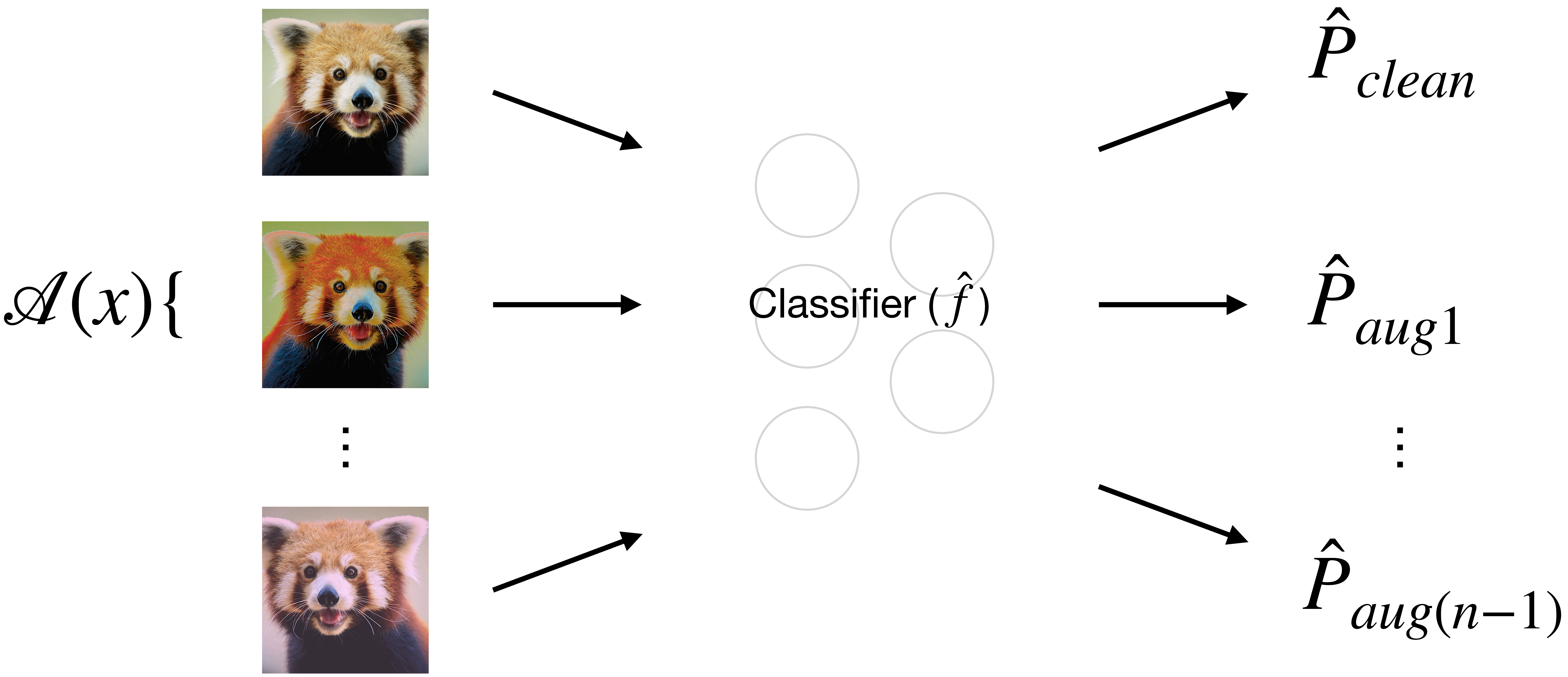
Data Augmentation
Background

- The general augmentor $\mathscr{A} : \mathscr{X} \rightarrow \mathscr{X}^n$ returns an n-tuple $(x_{clean}, x_{aug1}, x_{aug2}, \ldots, x_{aug(n-1)})$ given $x \in \mathscr{X}$, where $x = x_{clean}$ and each $x_{aug(i)}$ is a unique corruption of x.



Data Augmentation

Background



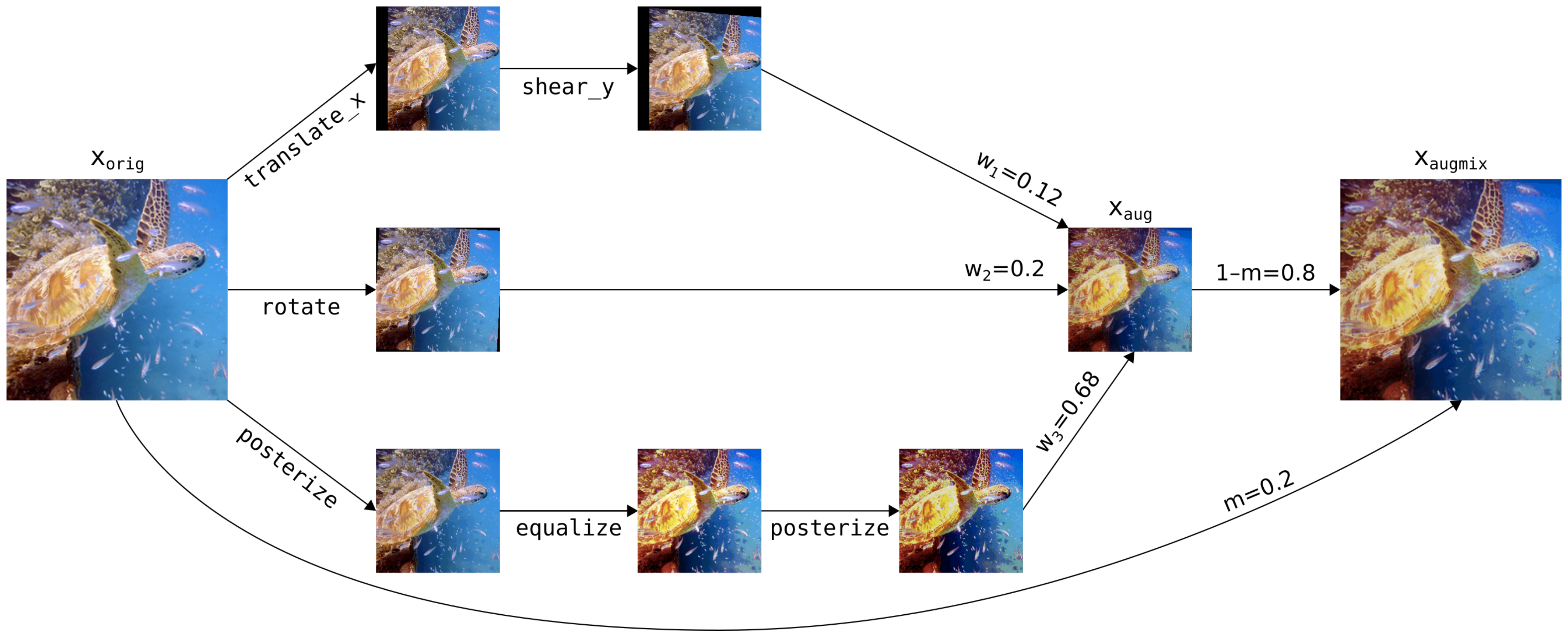
Data Augmentation
Background

- The use of augmentation warrants a loss supplement $\ell_{\mathscr{A}} : \mathscr{P}^n \rightarrow \mathbb{R}$ that regulates the output of each feature in $\mathscr{A}(x)$
- An effective regularizer will ensure that $\hat{P}_{clean} \approx \hat{P}_{aug(i)}$, which aims to improve the classifier $\hat{f}$'s robustness to corrupted features
- Training with augmentation makes use of the general loss function $\mathscr{L}$:
    - $\mathscr{L}\left(\hat{P}_{tuple}, y; \lambda\right) = \ell_{base}(\hat{P}_{clean}, y) + \lambda \cdot \ell_{\mathscr{A}}(\hat{P}_{clean}, \hat{P}_{aug1}, \ldots, \hat{P}_{aug(n-1)})$
    - $\ell_{base} \in \{\ell_{CE}\ , \ \ell_{+}\ , \ \ell_{\alpha}\ , \ \ \ldots\}$

Data Augmentation

Background

- Examples of augmentation
  - *STANDARD*: the case where $n = 1$. $\mathscr{A}_{STD}(x)$ simply returns x
  - *AUGMIX*: a SoTA example of $n = 3$. (Hendrycks, et al.)



Data Augmentation
Background

- Examples of augmentation regularizers
  - Jensen-Shannon Divergence Consistency Loss (ℓ_{JS})
    - $\hat{P}_{mix} = \frac{1}{n}\left(\hat{P}_{clean} + \hat{P}_{aug1} + \ldots + \hat{P}_{aug(n-1)}\right)$
    - $\ell_{JS}(\hat{P}_{tuple}) = \frac{1}{n}\left(KL(\hat{P}_{clean}\|\hat{P}_{mix}) + KL(\hat{P}_{aug1}\|\hat{P}_{mix}) + \ldots + KL(\hat{P}_{aug(n-1)}\|\hat{P}_{mix})\right)$
    - Note that $\ell_{JS} = 0$ with standard augmentation

Agenda

- Brief introduction
- Background
  - Image classification
  - Robust loss functions
  - Data augmentation
- Empirical investigation
  - Motivation
  - Setting (control & variable)
  - Hyperparameter tuning
  - Results summary
- Discussion
  - Estimated distribution metrics
  - Backpropagation
- Takeaways
- Next step
- Q&A

Motivation

Empirical Investigation

- We place ourselves in the following real-world classification setting:

  1. Our classifier $\hat{f}$'s architecture is state-of-the-art but fixed
  2. Our train & test sets are generated from some sufficiently large dataset $\mathscr{D}$
  3. The train set labels are corrupted at some *unknown* rate $0 < r < 0.5$
  4. The test set features undergo a series of common corruptions

Motivation

Empirical Investigation

- We propose the following set of tasks that formulate a robust solution:

1. Train with state-of-the-art data augmentation $\mathscr{A}$
2. Train with loss function $\mathscr{L} = \ell_{base} + \lambda \cdot \ell_{\mathscr{A}}$ for some base loss ℓ_{base}, positive scalar λ, and augmentation regularizer $\ell_{\mathscr{A}}$
3. Choose λ and $\ell_{\mathscr{A}}$ to optimize performance of $\mathscr{A}$
4. Tune a robust loss function family to optimize performance at some reasonable (& fixed) label noise rate $r_0 > 0$, and assign the result to ℓ_{base}

Motivation

Empirical Investigation

- We construct an investigation that
  - Assumes the classification setting outlined previously
  - Establishes a baseline metric
    - Train with $\mathscr{A} := \mathscr{A}_{STD}$ and $\ell_{base} := \ell_{CE}$
  - Sets $(\mathscr{A}, \lambda, \ell_{\mathscr{A}}) := (\mathscr{A}_{AUGMIX},\ 12, \ell_{JS})$ for state-of-the-art data augmentation
  - Reduces our task to the selection of ℓ_{base}

Motivation
Empirical Investigation

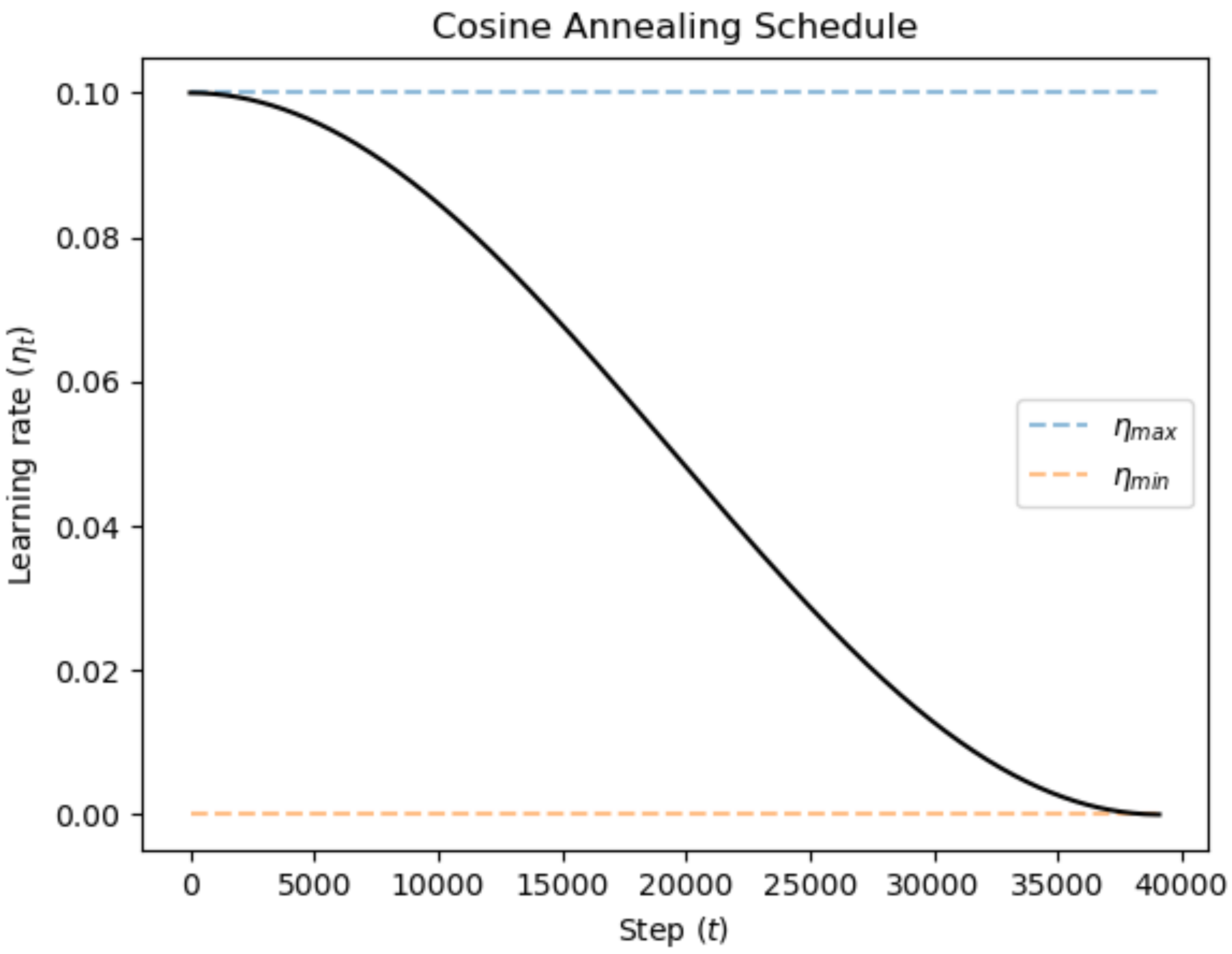
- Our investigation seeks to answer the following questions
  - Does the optimality of ℓ_{base} (w.r.t. test performance) depend on the choice of $\mathscr{A}$?
  - In our proposed classification setting, how does α-loss compare to NCE+RCE w.r.t. performance in
    - Hyperparameter tuning efficiency?
    - Evaluation on common corruptions?

Agenda

- Brief introduction
- Background
  - Image classification
  - Robust loss functions
  - Data augmentation
- Empirical investigation
  - Motivation
  - Setting (control & variable)
  - Hyperparameter tuning
  - Results summary
- Discussion
  - Estimated distribution metrics
  - Backpropagation
- Takeaways
- Next step
- Q&A

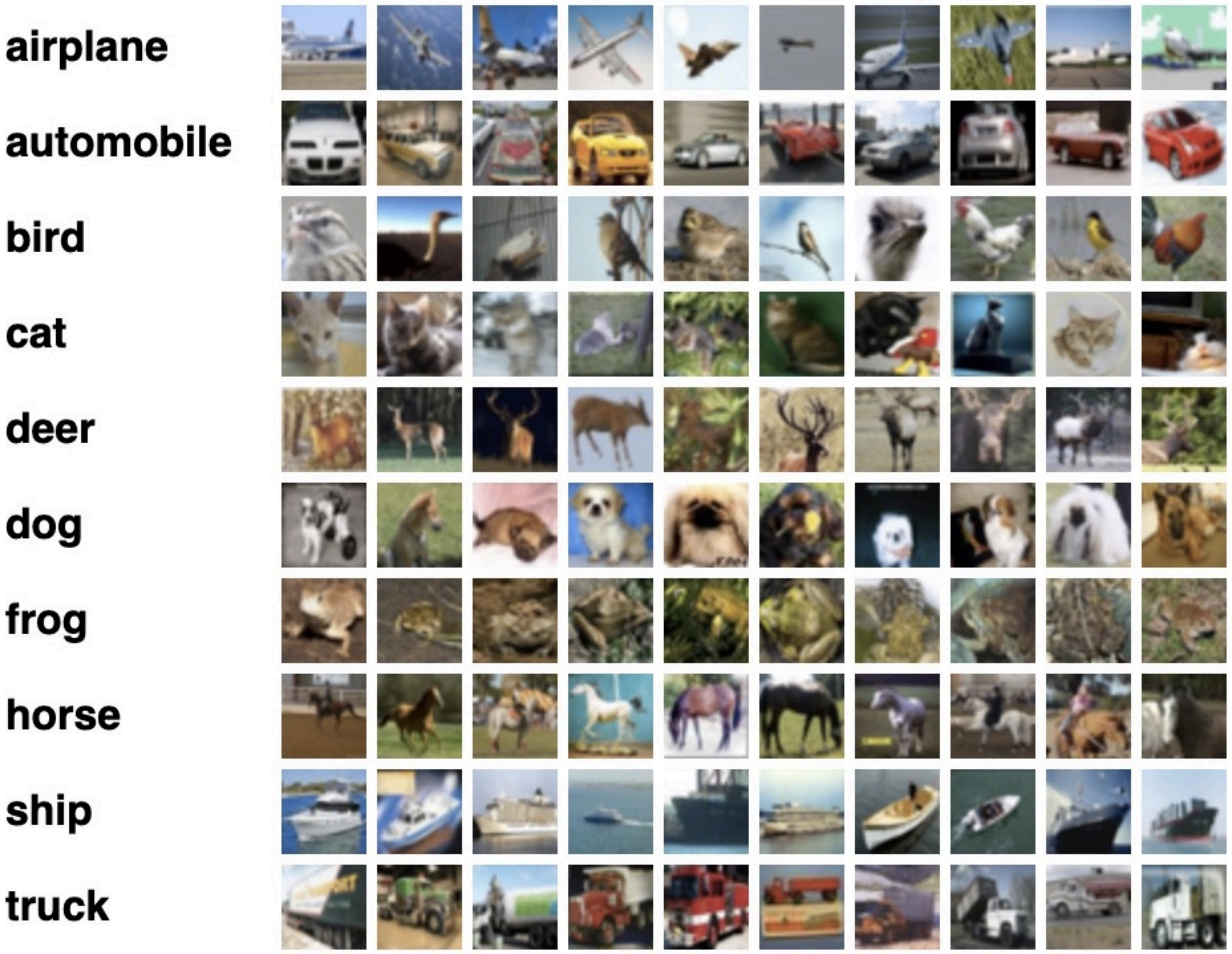
Setting (control)
Empirical Investigation

- Classifier ($\hat{f}$) architecture
  - SoTA Model: WideResNet-40-2
  - Optimizer: SGD
    - Nesterov momentum (γ) : 0.9
    - Weight decay (λ_w) : 5×10^{-4}
  - Learning rate scheduler: Cosine Annealing
    - Initial learning rate (η_{max}) : 0.1
    - Final learning rate (η_{min}) : 10^{-6}
  - Number of epochs: 100



Setting (variable)
Empirical Investigation

- Dataset ($\mathscr{D}$)
  - CIFAR-10
    - Classes: 10
    - Train-test: $50{,}000$ / $10{,}000$
    - Batch-size: 128



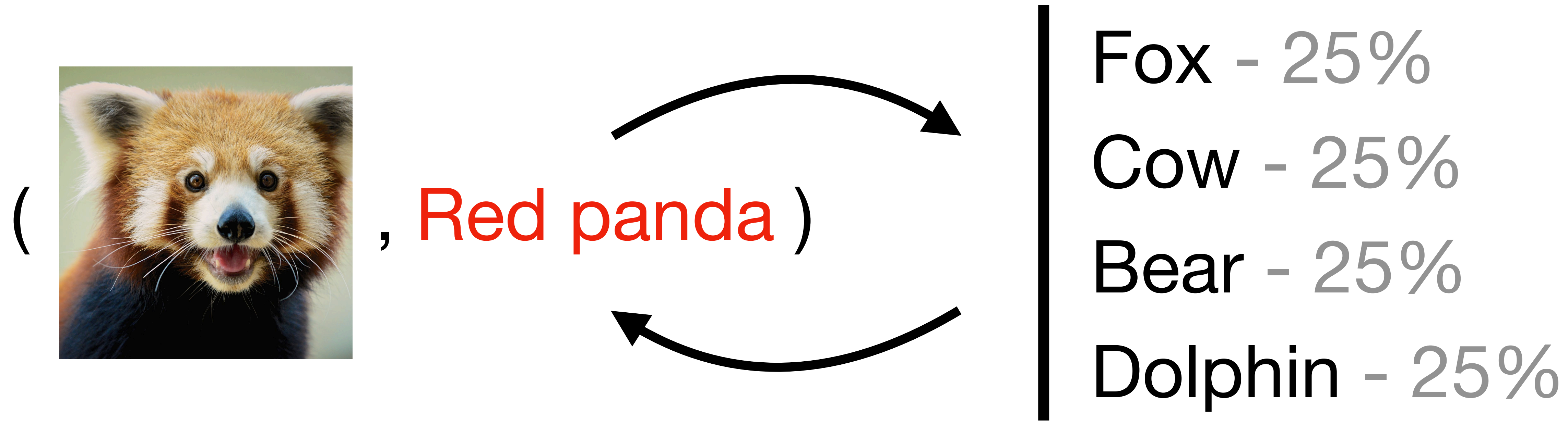
Setting (variable)
Empirical Investigation

- Dataset ($\mathscr{D}$)
  - CIFAR-100
    - Classes: 100
    - Superclasses: 20
    - Train-test: $50{,}000$ / $10{,}000$
    - Batch-size: 128

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Setting (variable)

Empirical Investigation



Symmetric noise labeling

- Label noise generation (train set)
  - Noise rate $r \in \{0.0^*,\ 0.1, 0.2, 0.3, 0.4\}$
  - Methods
    - *Symmetric:* Each label with probability r is flipped; the other labels are equally likely to be chosen as the new one
    - *Asymmetric:* Each label with probability r is flipped; labels with similar classes are more likely to be chosen as the new one

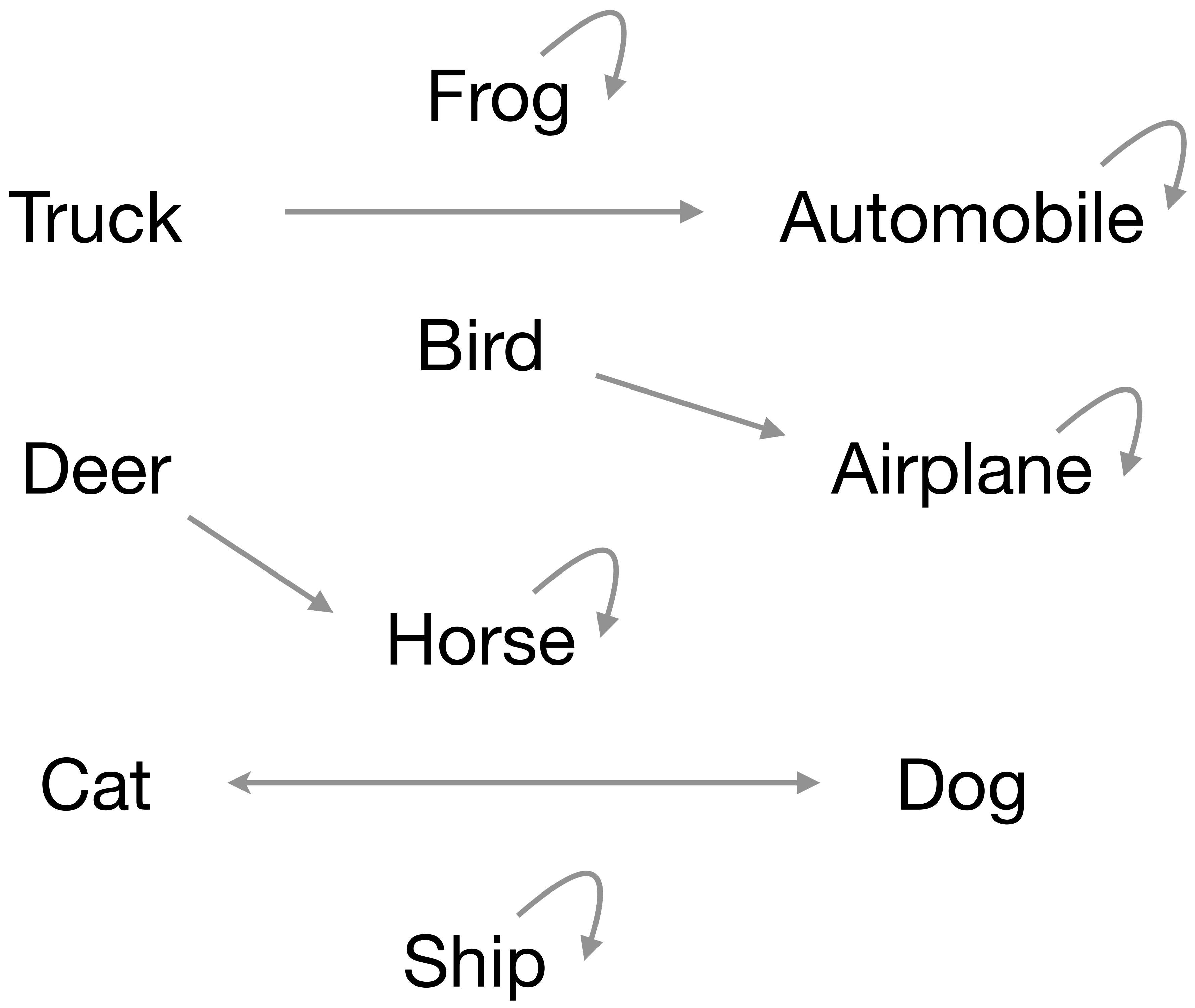
* Baseline metric

Setting (variable)

Empirical Investigation

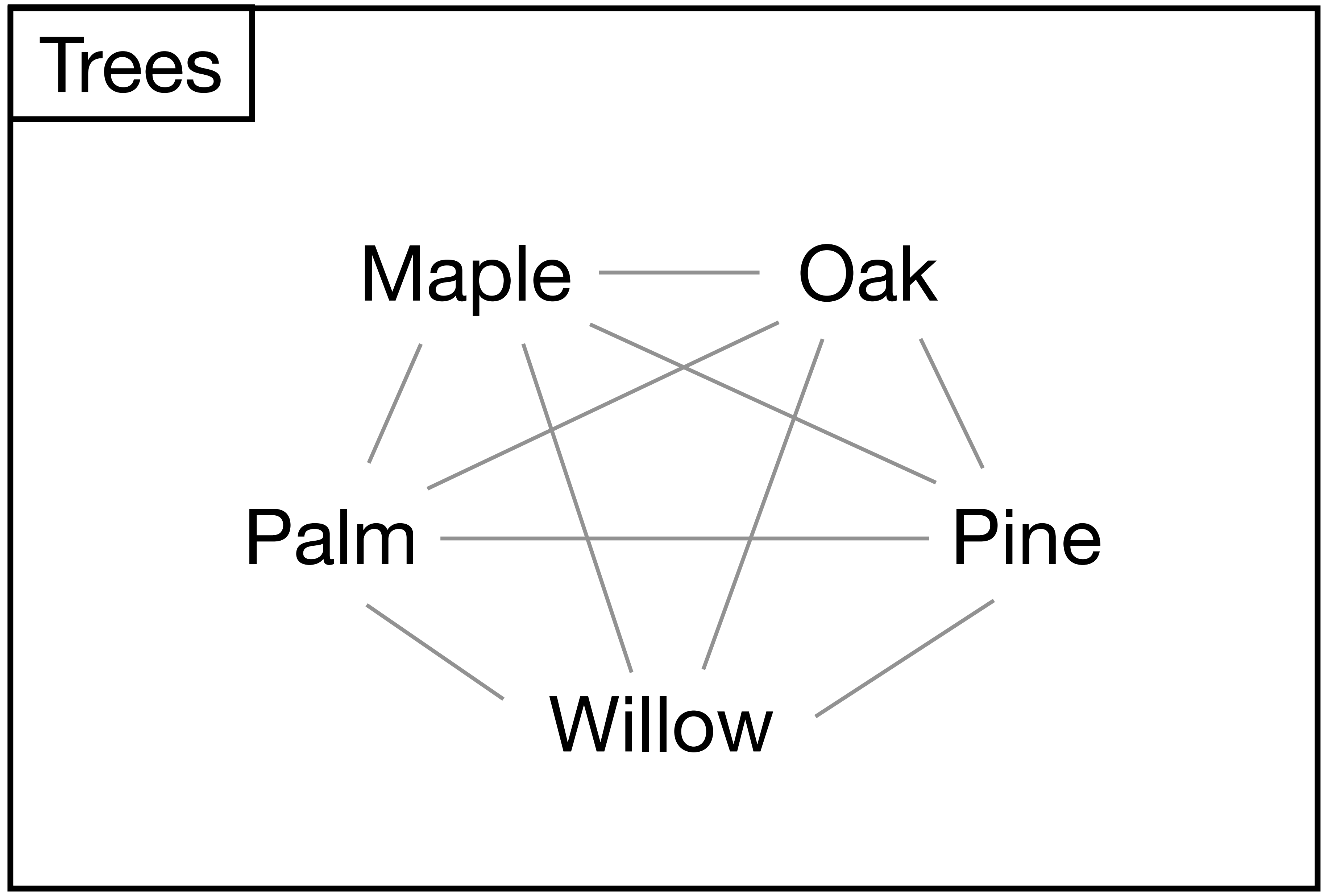
- Label noise generation (train set)
    - Asymmetric mappings

CIFAR-10



CIFAR-100

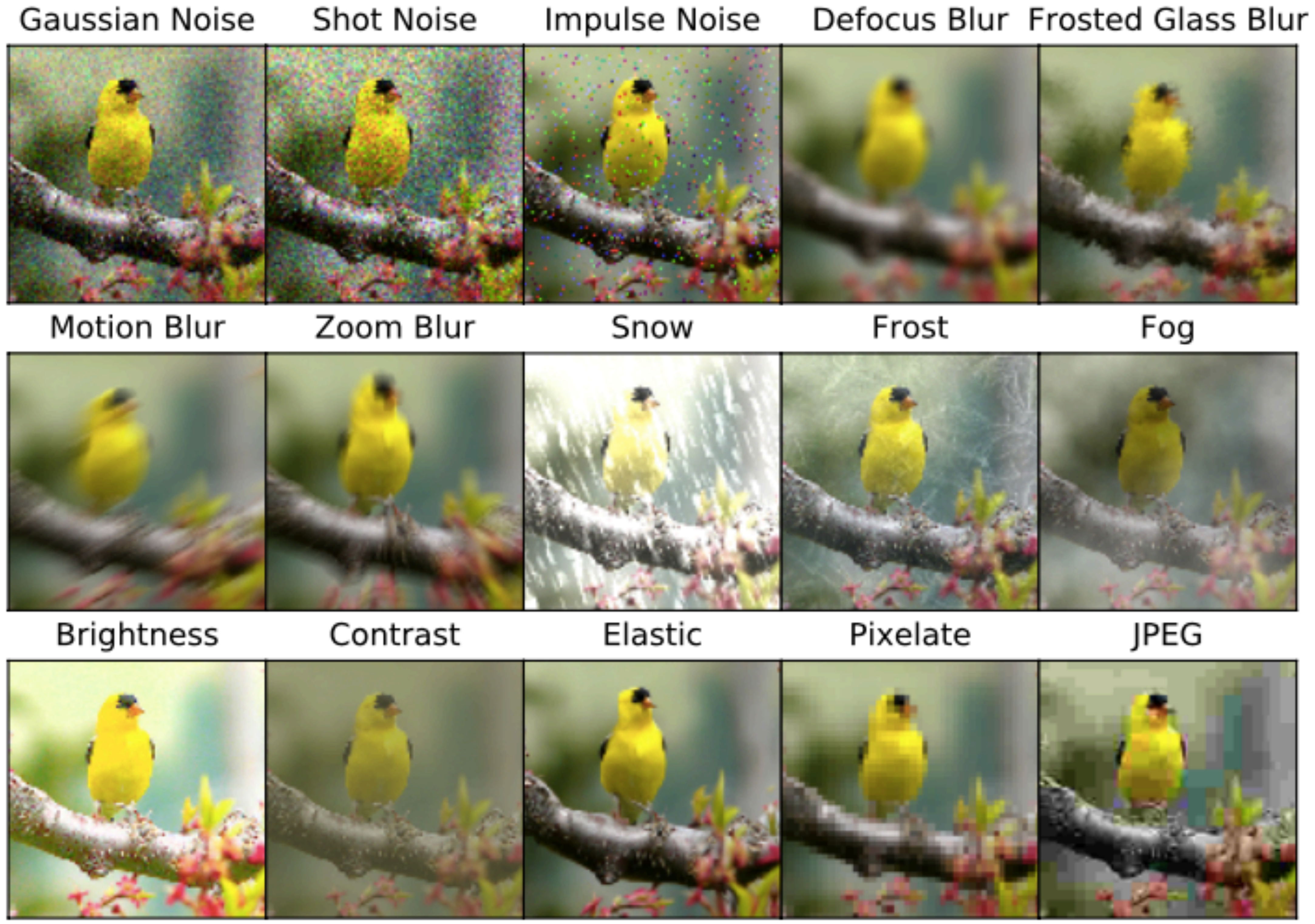
- Symmetric mapping within each superclass



Setting (variable)
Empirical Investigation

- Feature noise generation (test set)
  - *Clean*:* simply test on original set
  - *Corruption:* test on 15 sets, each generated by a different common corruption; report the mean error

```
errors = [ ]
for corruption ∈ corruptions do:
    corrupt_set = corruption(test_set)
    test_error = test(corrupt_set)
    errors.add(test_error)
return mean(errors)
```



* Baseline metric

Setting (variable)

Empirical Investigation

- Data augmentation
  - $\mathscr{A} \in \{\mathscr{A}_{STD}{}^*, \mathscr{A}_{AUGMIX}\}$
  - $\mathscr{A}_{STD}$: Identity augmentor; $x \mapsto x$
  - $\mathscr{A}_{AUGMIX}$: Augmix; $x \mapsto (x_{clean}, x_{aug1}, x_{aug2})$
- Base loss function
  - $\ell_{base} \in \{\ell_{CE}{}^*, \ell_+, \ell_\alpha\}$
  - General loss function $\mathscr{L} = \ell_{base} + \lambda \cdot \ell_{JS}$, where $\lambda = 12$

* Baseline metric

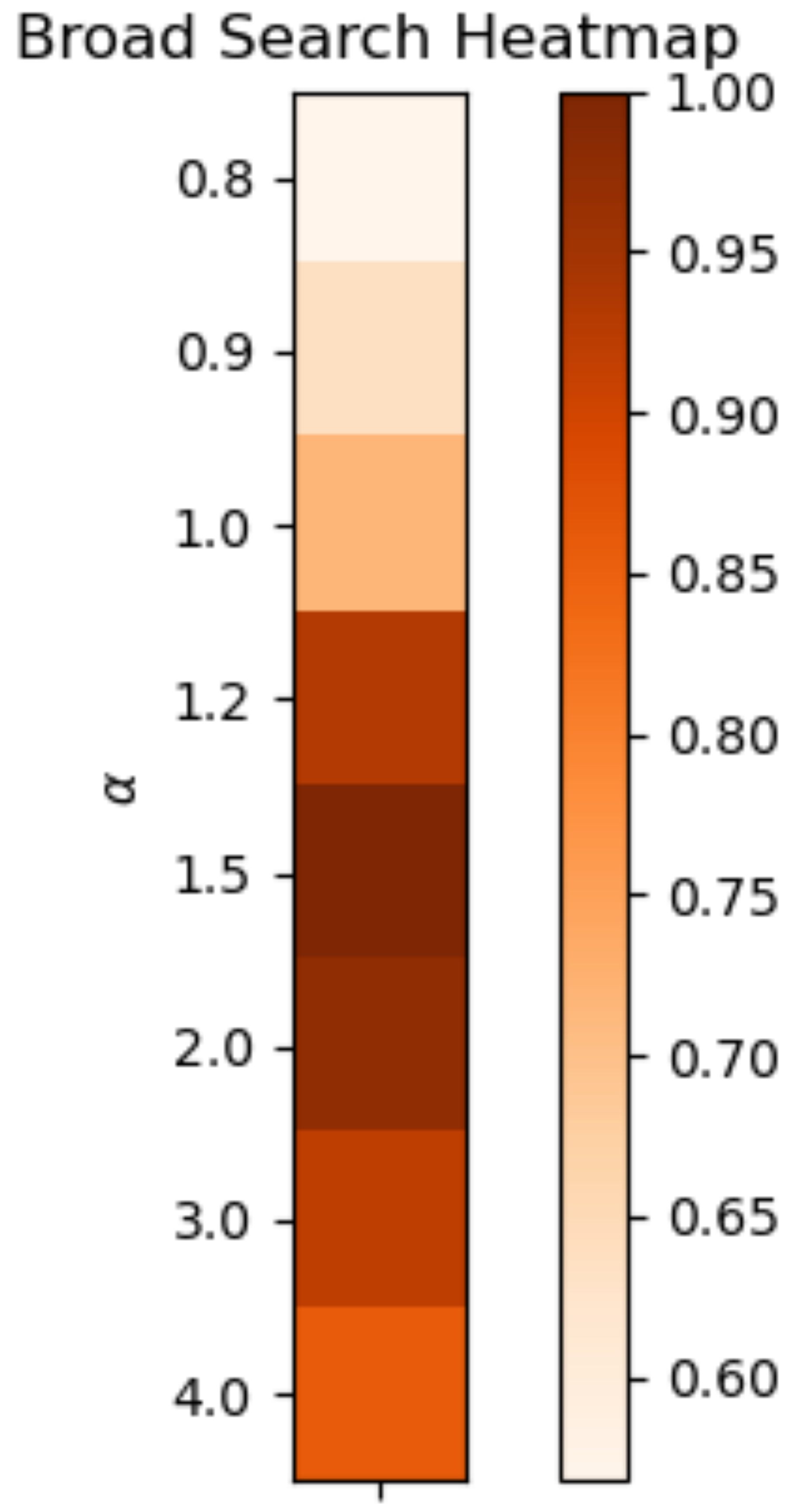
Agenda

- Brief introduction
- Background
    - Image classification
    - Robust loss functions
    - Data augmentation
- Empirical investigation
    - Motivation
    - Setting (control & variable)
    - Hyperparameter tuning
    - Results summary
- Discussion
    - Estimated distribution metrics
    - Backpropagation
- Takeaways
- Next step
- Q&A

Hyperparameter Tuning

Empirical Investigation

- The α-*loss* family is parameterized by $\alpha \in \mathbb{R}^+$
  - Tuning algorithm for each setting combination with $r_0 = 0.2$:
    1. Run broad search
       - $\Pi_B = \{0.8, 0.9, 1.0, 1.2, 1.5, 2, 3, 4, 6\}$
    2. Set & run narrowed search
       - $accuracy(\Pi_B)$ consistently unimodal; center Π_N around peak
    3. Select $\alpha^* = \arg\max_{\alpha \in \Pi_N} \left\{ accuracy\left(\ell_a(\alpha)\right) \right\}$



Hyperparameter Tuning

Empirical Investigation

- The *NCE+RCE* family is parameterized by $(\alpha, \beta) \in \mathbb{R}^+ \times \mathbb{R}^+$
  - If we let $k := \alpha + \beta$ and $c := \dfrac{\alpha}{\alpha + \beta}$, then we can rewrite ℓ_+ to be
    - $\ell_+ = k\left(c \cdot \ell_{NCE} + (1 - c) \cdot \ell_{RCE}\right)$
  - Now we have two parameters k, c that intuitively denote the scale of ℓ_+ and ratio between ℓ_{NCE} and ℓ_{RCE} , respectively
  - Then we search for $(k*, c*)$ and solve $(\alpha*, \beta*) = \left(k*c*, k*(1 - c*)\right)$

Hyperparameter Tuning

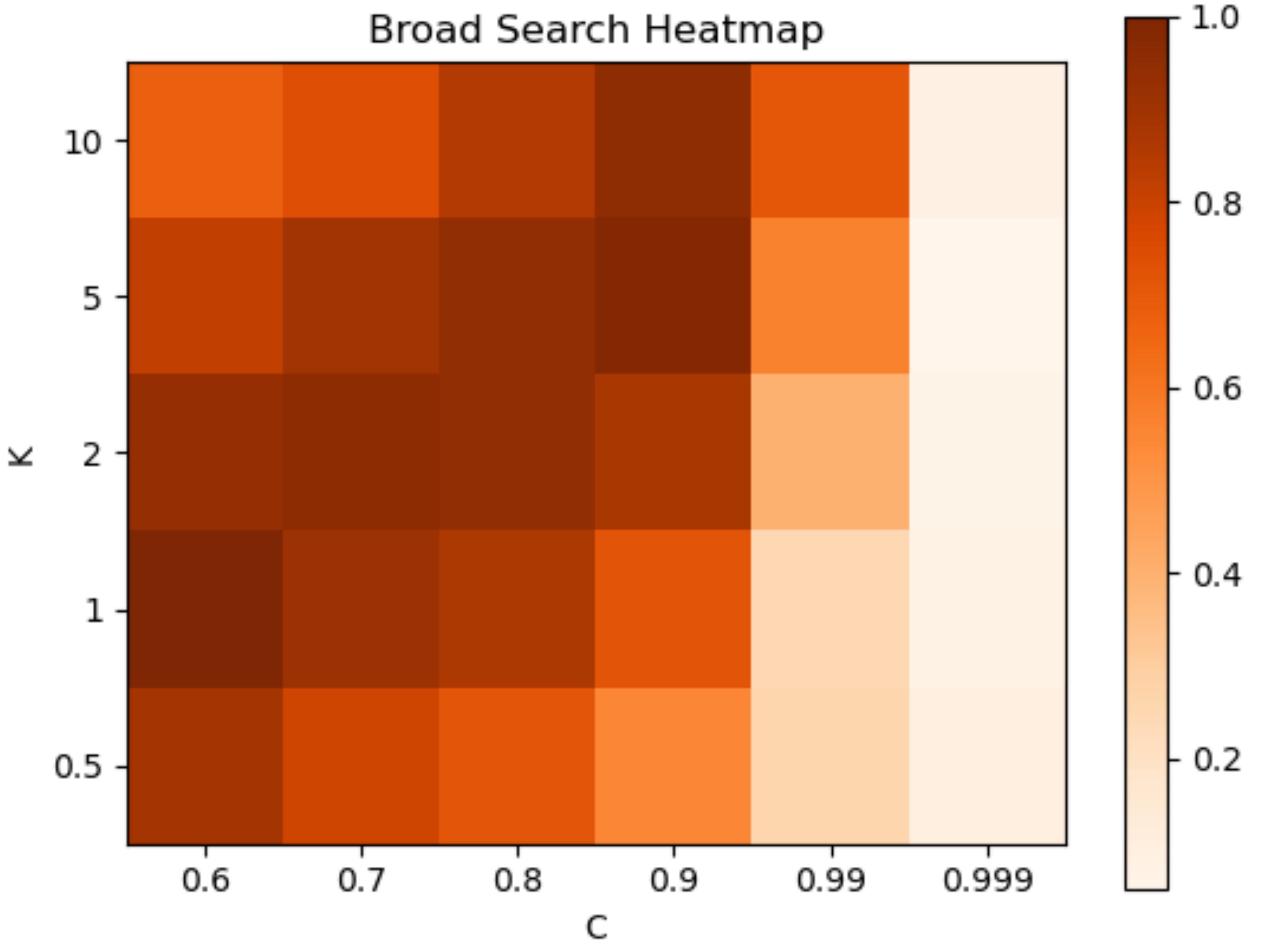
Empirical Investigation

- The *NCE+RCE* family is parameterized by $(\alpha, \beta) \in \mathbb{R}^+ \times \mathbb{R}^+$
    - Tuning algorithm for each setting combination with $r_0 = 0.2$:
        1. Run broad search
- $$\Pi_B = \begin{cases} \{0.5,1,2,5,10\} \times \{0.8,0.9,0.99,0.999\} & \mathscr{D}_{CIFAR-10} \wedge \mathscr{A}_{STD} \\ \{20,40,60,80,100,120\} \times \{0.8,0.9,0.99,0.999,0.9999\} & \mathscr{D}_{CIFAR-100} \wedge \mathscr{A}_{STD} \\ \{0.5,1,2,5,10\} \times \{0.6,0.7,0.8,0.9,0.99,0.999\} & \mathscr{D}_{CIFAR-10} \wedge \mathscr{A}_{AUGMIX} \\ \{20,40,60,80,100,120\} \times \{0.5,0.7,0.8,0.9,0.99,0.999,0.9999\} & \mathscr{D}_{CIFAR-100} \wedge \mathscr{A}_{AUGMIX} \end{cases}$$

Hyperparameter Tuning

Empirical Investigation

- The *NCE+RCE* family is parameterized by $(\alpha, \beta) \in \mathbb{R}^+ \times \mathbb{R}^+$
  - Tuning algorithm for each setting combination with $r_0 = 0.2$:
    - 2. Set & run narrowed search
      - $accuracy(\Pi_B)$ can be unimodal, bimodal, or multimodal
      - Construct a space $\Pi_N^{(i)}$ centered around each peak π_i
      - Then $\Pi_N = \bigcup_i \Pi_N^{(i)}$
    - 3. Select $(k*, c*) = \arg \max_{(k,c)\in\Pi_N} \left\{ accuracy\left(\ell_+(k,c)\right) \right\}$



Agenda

- Brief introduction
- Background
  - Image classification
  - Robust loss functions
  - Data augmentation
- Empirical investigation
  - Motivation
  - Setting (control & variable)
  - Hyperparameter tuning
  - Results summary
- Discussion
  - Estimated distribution metrics
  - Backpropagation
- Takeaways
- Next step
- Q&A

Results Summary

Empirical Investigation

Task	Dataset	Loss	Parameter	Noise 0	0.1	0.2	0.3	0.4
Standard Symmetric	CIFAR10	CE	—-	**27.06 ± 0.27**	35.71 ± 0.54	39.94 ± 0.97	44.94 ± 0.92	51.65 ± 0.32
		α-loss	3	28.42 ± 0.35	**29.33 ± 0.73**	30.98 ± 0.15	**32.54 ± 0.64**	**34.87 ± 1.57**
		NCE+RCE	(0.6,0.99)	30.1 ± 0.22	30.1 ± 0.22	**30.61 ± 0.36**	32.63 ± 0.28	34.9 ± 0.52
	CIFAR100	CE	—-	**53.6 ± 0.2**	59.69 ± 0.31	64.49 ± 0.22	68.5 ± 0.27	72.6 ± 0.17
		α-loss	2	54.29 ± 0.1	**55.44 ± 0.23**	56.72 ± 0.34	57.85 ± 0.33	60.38 ± 0.26
		NCE+RCE	(80,0.99)	55.66 ± 0.42	56.59 ± 0.17	**55.65 ± 1.83**	**57.65 ± 1.9**	**57.37 ± 1.62**
Standard Asymmetric	CIFAR10	CE	—-	**26.71 ± 0.41**	29.87 ± 0.49	32.61 ± 0.21	34.71 ± 0.87	38.47 ± 0.24
		α-loss	2.5	27.76 ± 0.56	**28.42 ± 0.57**	30.65 ± 0.22	33.27 ± 0.72	39.28 ± 0.46
		NCE+RCE	(5,0.995)	28.58 ± 0.92	28.98 ± 0.33	**30.11 ± 1.28**	**32.61 ± 0.41**	**37.8 ± 0.89**
	CIFAR100	CE	—-	**53.74 ± 0.04**	58.9 ± 0.25	62.89 ± 0.27	66.03 ± 0.50	69.96 ± 0.28
		α-loss	3	54.39 ± 0.02	**55.55 ± 0.30**	57.67 ± 0.26	59.78 ± 0.62	62.92 ± 0.36
		NCE+RCE	(110,0.999)	54.44 ± 0.18	55.85 ± 0.13	**56.93 ± 0.16**	**59.21 ± 0.61**	**61.65 ± 0.17**

TABLE 1. MEAN CORRUPTION ERROR GIVEN VARYING TASKS WHEN MODEL TUNED TO MINIMIZE MCE.

Results Summary

Empirical Investigation

Task	Dataset	Loss	Parameter	Noise 0	0.1	0.2	0.3	0.4
Augmix Symmetric	CIFAR10	CE	—-	**11.2 ± 0.07**	12.86 ± 0.07	14.81 ± 0.23	17.47 ± 0.25	21.29 ± 0.25
		α-loss	2	11.29 ± 0.33	**11.79 ± 0.21**	**12.36 ± 0.07**	**12.95 ± 0.17**	**14.09 ± 0.26**
		NCE+RCE	(1,0.8)	12.21 ± 0.09	12.36 ± 0.16	12.58 ± 0.06	13.14 ± 0.12	**14.09 ± 0.25**
	CIFAR100	CE	—-	35.83 ± 0.15	38.76 ± 0.18	41.26 ± 0.12	43.95 ± 0.11	47.69 ± 0.21
		α-loss	1.3	**35.74 ± 0.15**	**36.58 ± 0.08**	**37.56 ± 0.13**	**39.73 ± 0.13**	**41.94 ± 0.08**
		NCE+RCE	(50,0.99)	38.08 ± 0.09	38.15 ± 0.08	39.09 ± 0.12	40.7 ± 0.05	42.75 ± 0.29
Augmix Asymmetric	CIFAR10	CE	—-	**11.24 ± 0.16**	11.9 ± 0.04	12.77 ± 0.06	14.12 ± 0.02	16.77 ± 0.29
		α-loss	1.7	11.45 ± 0.1	**11.75 ± 0.14**	**12.34 ± 0.26**	**13.65 ± 0.32**	**16.1 ± 0.05**
		NCE+RCE	(1,0.7)	12.53 ± 0.45	12.61 ± 0.22	12.78 ± 0.43	13.74 ± 0.27	25.93 ± 1.45
	CIFAR100	CE	—-	**35.72 ± 0.14**	38.3 ± 0.34	40.29 ± 0.11	42.33 ± 0.33	44.68 ± 0.21
		α-loss	1.5	36.09 ± 0.12	**37.19 ± 0.16**	**38.43 ± 0.1**	**40.19 ± 0.34**	**41.37 ± 0.11**
		NCE+RCE	(50,0.99)	38.04 ± 0.19	38.95 ± 0.24	39.97 ± 0.27	41.42 ± 0.31	43.61 ± 0.02

TABLE 1. MEAN CORRUPTION ERROR GIVEN VARYING TASKS WHEN MODEL TUNED TO MINIMIZE MCE.

Results Summary
Empirical Investigation

BASELINE: $\mathscr{A}_{STD}$ + ℓ_{CE}

	Loss	CIFAR-10	CIFAR-100
Symmetric	CE	43.06	66.32
Asymmetric	CE	33.92	64.45

Results Summary

Empirical Investigation

- Both α-loss and NCE+RCE significantly outperform CE
- α-loss is competitive with NCE+RCE
- NCE+RCE shows to be SoTA

(1)

BASELINE: $\mathscr{A}_{STD} + \ell_{CE}$

	Loss	CIFAR-10	CIFAR-100
Symmetric	CE	43.06	66.32
Asymmetric	CE	33.92	64.45

$+ \ell_\alpha, \ell_+$ (1)

	Loss	CIFAR-10	CIFAR-100
Symmetric	α-loss	31.88	57.60
	NCE+RCE	32.06	56.82
Asymmetric	α-loss	33.16	58.98
	NCE+RCE	32.38	58.41

Results Summary

Empirical Investigation

- SoTA data augmentation *drastically* improves performance on corrupted test features even when the base loss is not robust to label noise (2)

BASELINE: $\mathscr{A}_{STD} + \ell_{CE}$

	Loss	CIFAR-10	CIFAR-100
Symmetric	CE	43.06	66.32
Asymmetric	CE	33.92	64.45

$+ \mathscr{A}_{AUGMIX}$ → (2)

	Loss	CIFAR-10	CIFAR-100
Symmetric	CE	16.61	42.92
Asymmetric	CE	13.89	41.4

$+ \ell_{\alpha}, \ell_{+}$ ↓ (1)

	Loss	CIFAR-10	CIFAR-100
Symmetric	α-loss	31.88	57.60
	NCE+RCE	32.06	56.82
Asymmetric	α-loss	33.16	58.98
	NCE+RCE	32.38	58.41

Results Summary

Empirical Investigation

- In our designed task, α-loss slightly but consistently outperforms NCE+RCE
- This task produces the best overall results for our designed setting

(3)

BASELINE: $\mathscr{A}_{STD} + \ell_{CE}$

	Loss	CIFAR-10	CIFAR-100
Symmetric	CE	43.06	66.32
Asymmetric	CE	33.92	64.45

(2) $+ \mathscr{A}_{AUGMIX}$

	Loss	CIFAR-10	CIFAR-100
Symmetric	CE	16.61	42.92
Asymmetric	CE	13.89	41.4

(1) $+ \ell_\alpha, \ell_+$

	Loss	CIFAR-10	CIFAR-100
Symmetric	α-loss	31.88	57.60
	NCE+RCE	32.06	56.82
Asymmetric	α-loss	33.16	58.98
	NCE+RCE	32.38	58.41

(3) $+ \mathscr{A}_{AUGMIX}$

$+ \ell_\alpha, \ell_+$

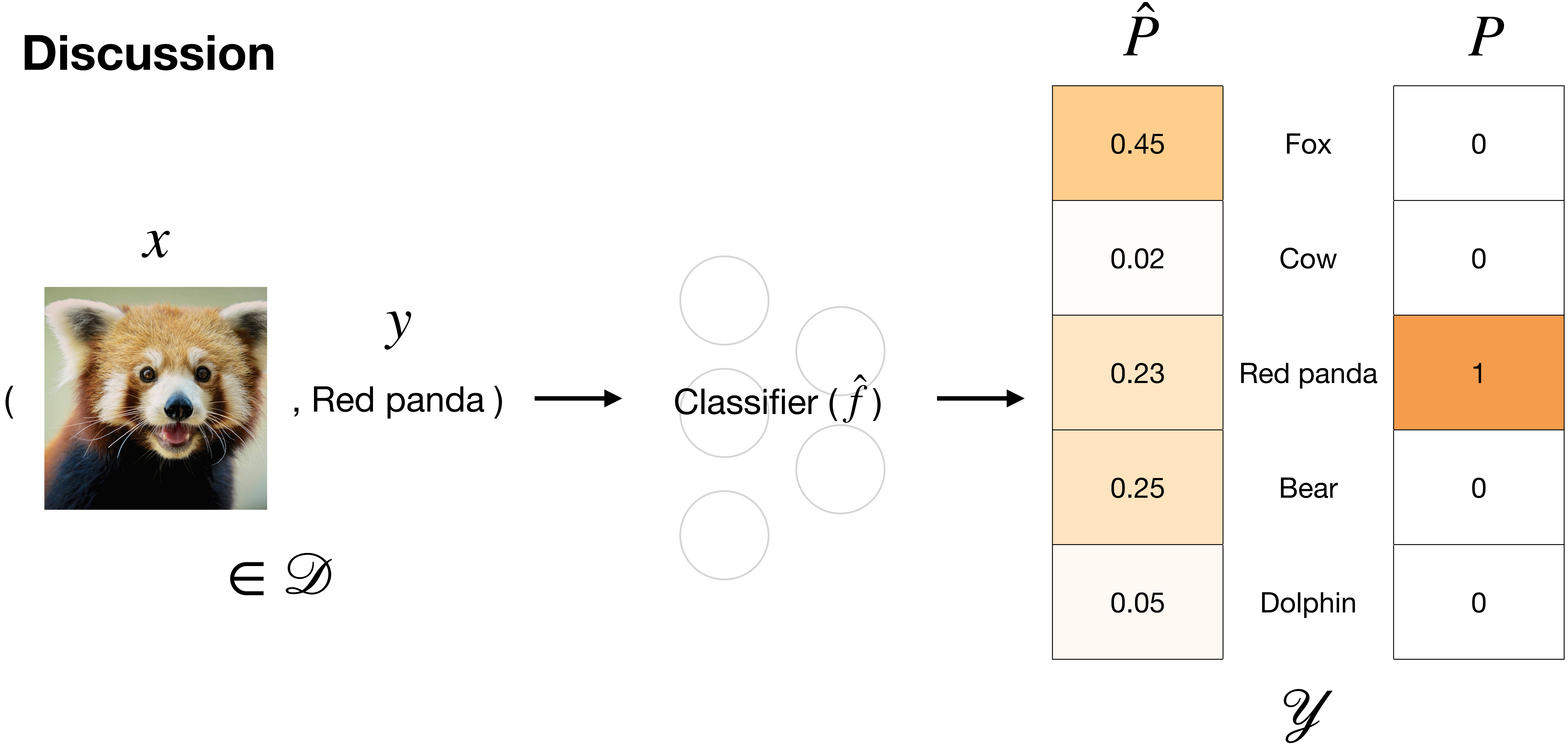
	Loss	CIFAR-10	CIFAR-100
Symmetric	α-loss	12.80	38.95
	NCE+RCE	13.04	40.17
Asymmetric	α-loss	13.46	39.30
	NCE+RCE	16.27	40.99

Agenda

- Brief introduction
- Background
  - Image classification
  - Robust loss functions
  - Data augmentation
- Empirical investigation
  - Motivation
  - Setting (control & variable)
  - Hyperparameter tuning
  - Results summary
- Discussion
  - Estimated distribution metrics
  - Backpropagation
- Takeaways
- Next step
- Q&A

Estimated Distribution Metrics

Discussion



Estimated Distribution Metrics

Discussion

- Metrics of $\hat{P}$ important to consider in the context of label corruption:
  - Estimated probability for the true class
    - $\hat{1}_x = \hat{P}(f(x) \mid x) = 0.23$
    - **Smaller values of $\hat{1}_x$ should indicate a greater likelihood of a label flip**
      - *"Red panda"* is a false-positive class

$\hat{P}$	$\mathcal{Y}$	P
0.45	Fox	0
0.02	Cow	0
0.23	Red panda	1
0.25	Bear	0
0.05	Dolphin	0

Estimated Distribution Metrics

Discussion

- Metrics of $\hat{P}$ important to consider in the context of label corruption:
  - Highest estimated probability of the false classes
    - $\hat{0}_x = \max\{\hat{P}(k \mid x) : f(x) \neq k \in \mathscr{Y}\}$
      $= 0.45$
    - **Larger values of $\hat{0}_x$ should indicate a greater likelihood of a label flip**
      - *"Fox"* is a false-negative class

$\hat{P}$	$\mathscr{Y}$	P
0.45	Fox	0
0.02	Cow	0
0.23	Red panda	1
0.25	Bear	0
0.05	Dolphin	0

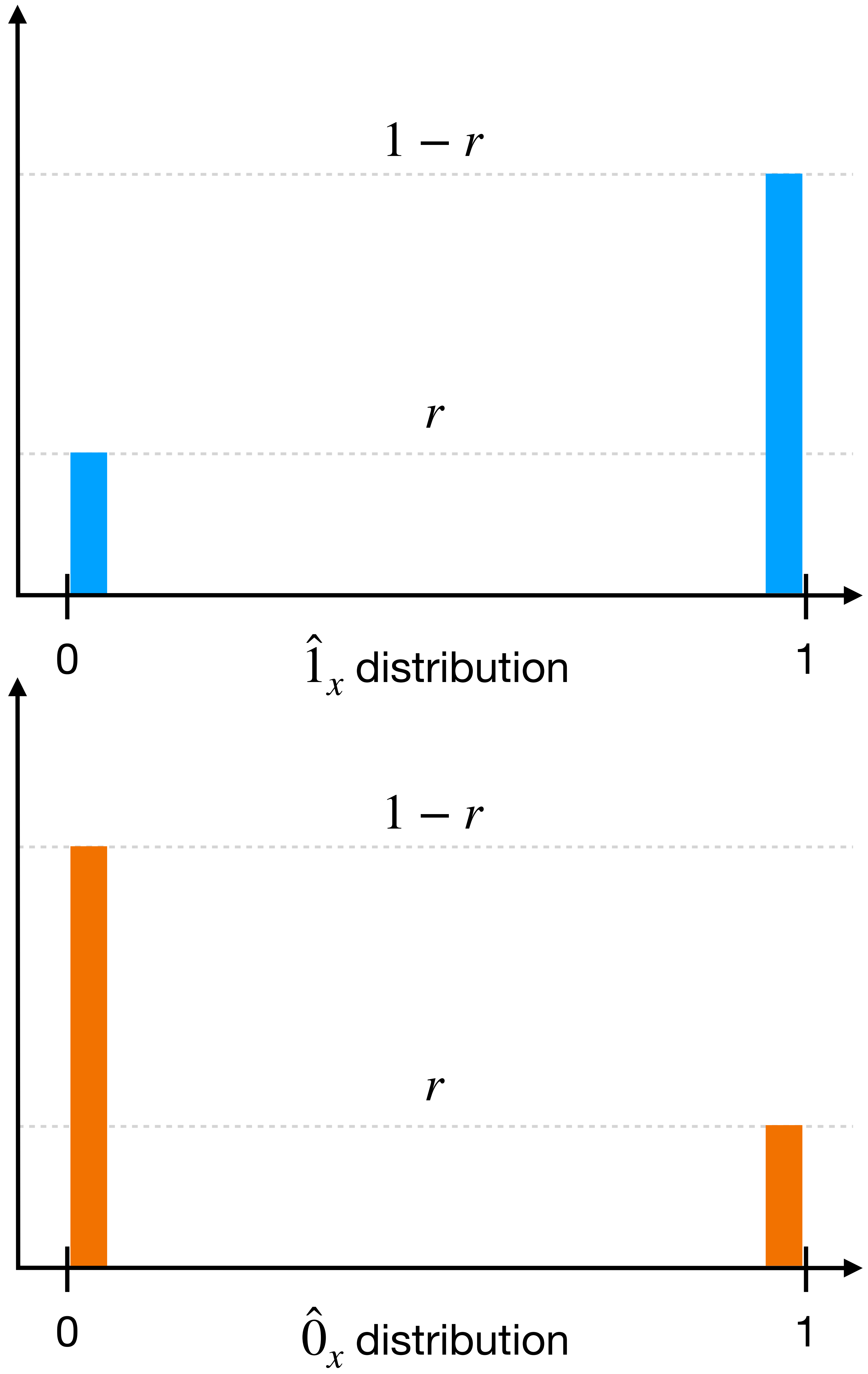
Estimated Distribution Metrics

Discussion

- A classifier $\hat{f}$ trained on set T with label noise rate r and perfectly robust ℓ_{base} must contain the following properties:

Let $\mathcal{X}_T$ denote the set of features in T. Then

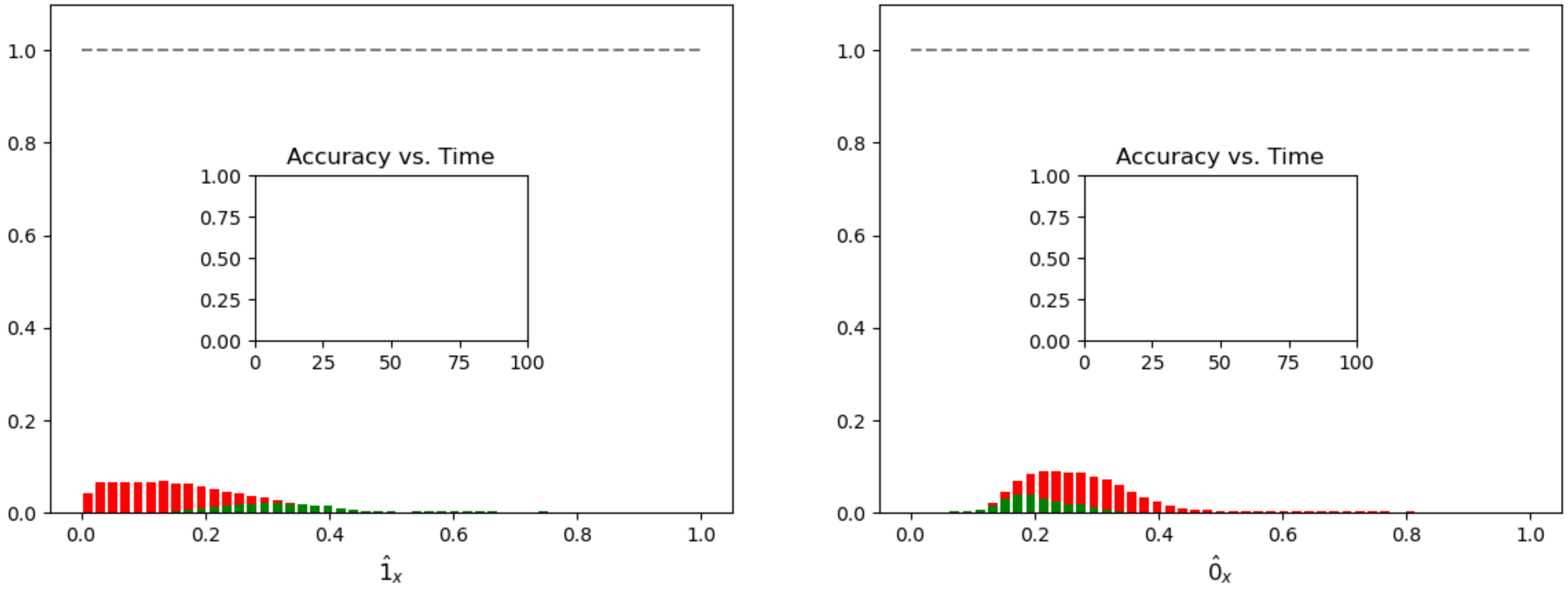
1. $\hat{1}_x \approx 0$ for $100(r)$% of $x \in \mathcal{X}_T$
2. $\hat{1}_x \approx 1$ for $100(1-r)$% of $x \in \mathcal{X}_T$
3. $\hat{0}_x \approx 0$ for $100(1-r)$% of $x \in \mathcal{X}_T$
4. $\hat{0}_x \approx 1$ for $100(r)$% of $x \in \mathcal{X}_T$



Estimated Distribution Metrics

Discussion

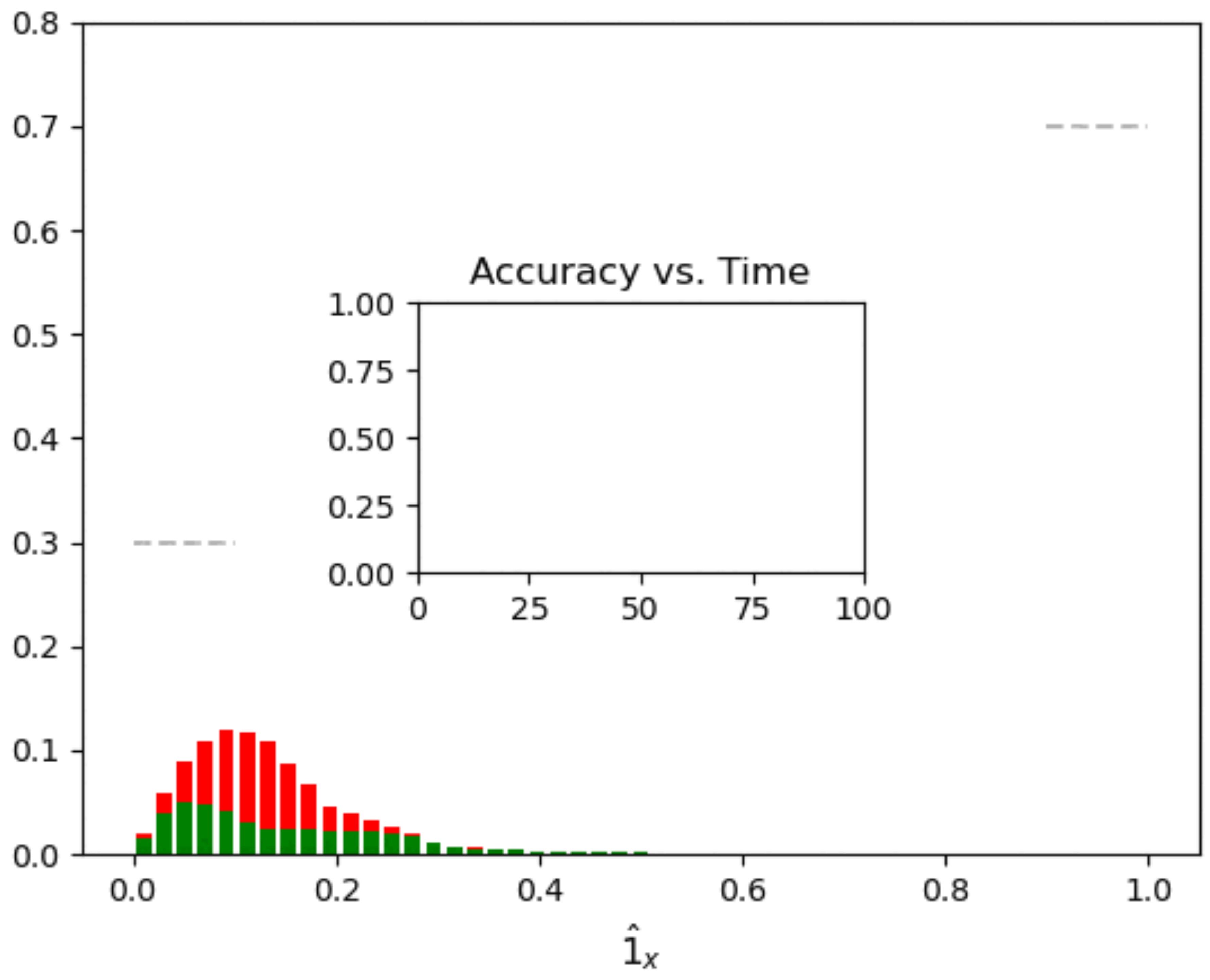
- The following example* confirms that ℓ_{CE} is robust to $r = 0.0$

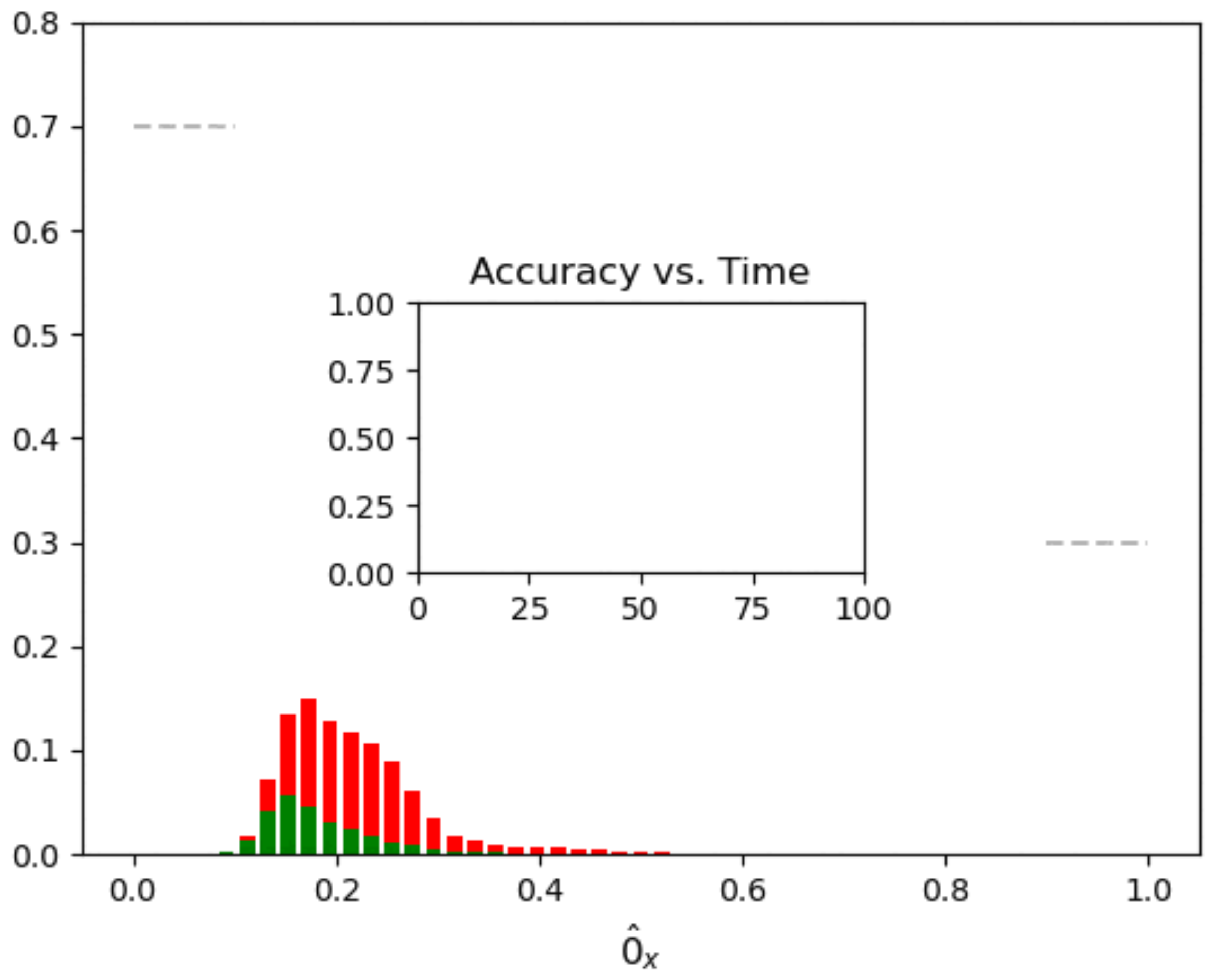


Estimated Distribution Metrics

Discussion

- However, this example* illustrates that ℓ_{CE} is **not** robust to $r = 0.3$



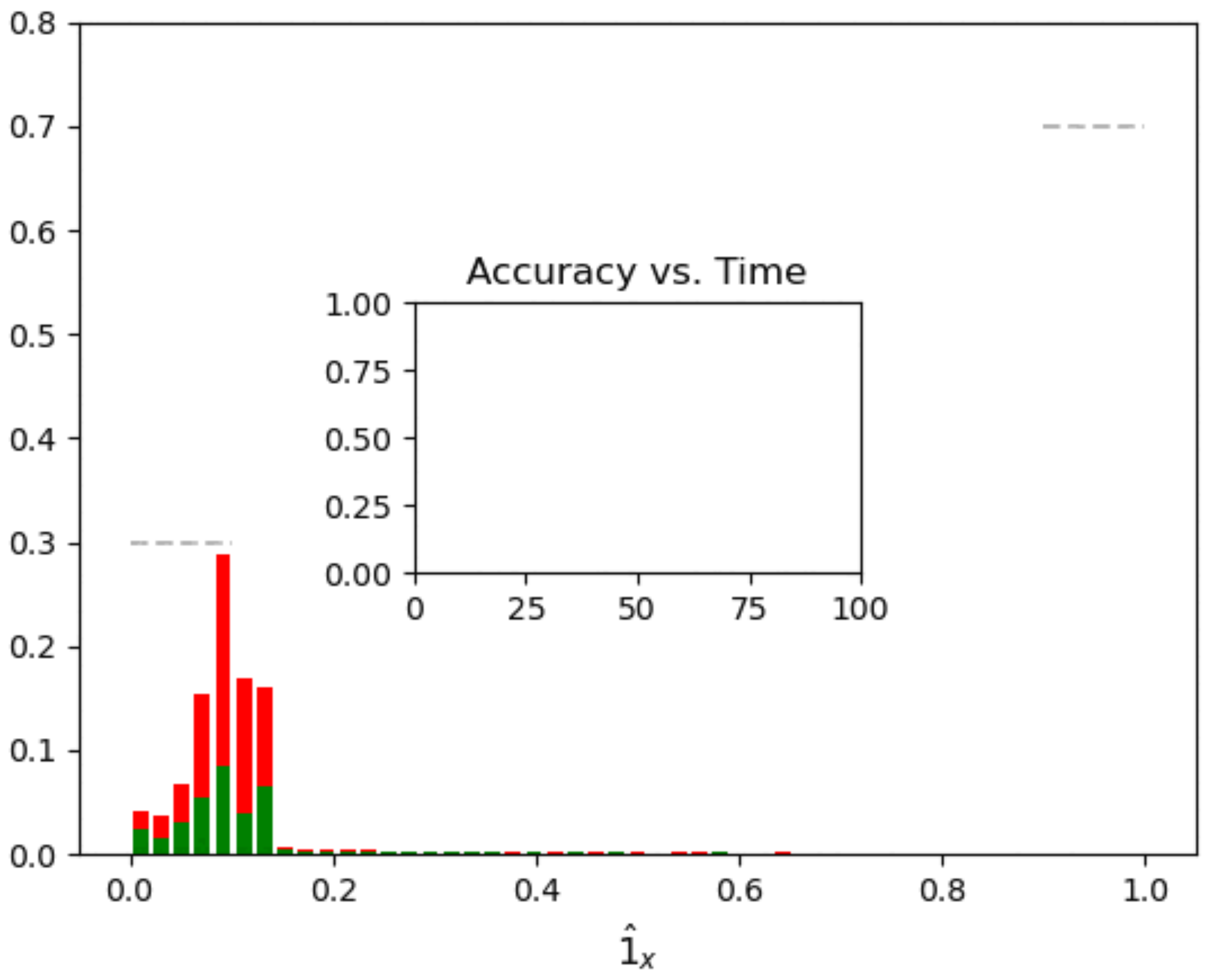


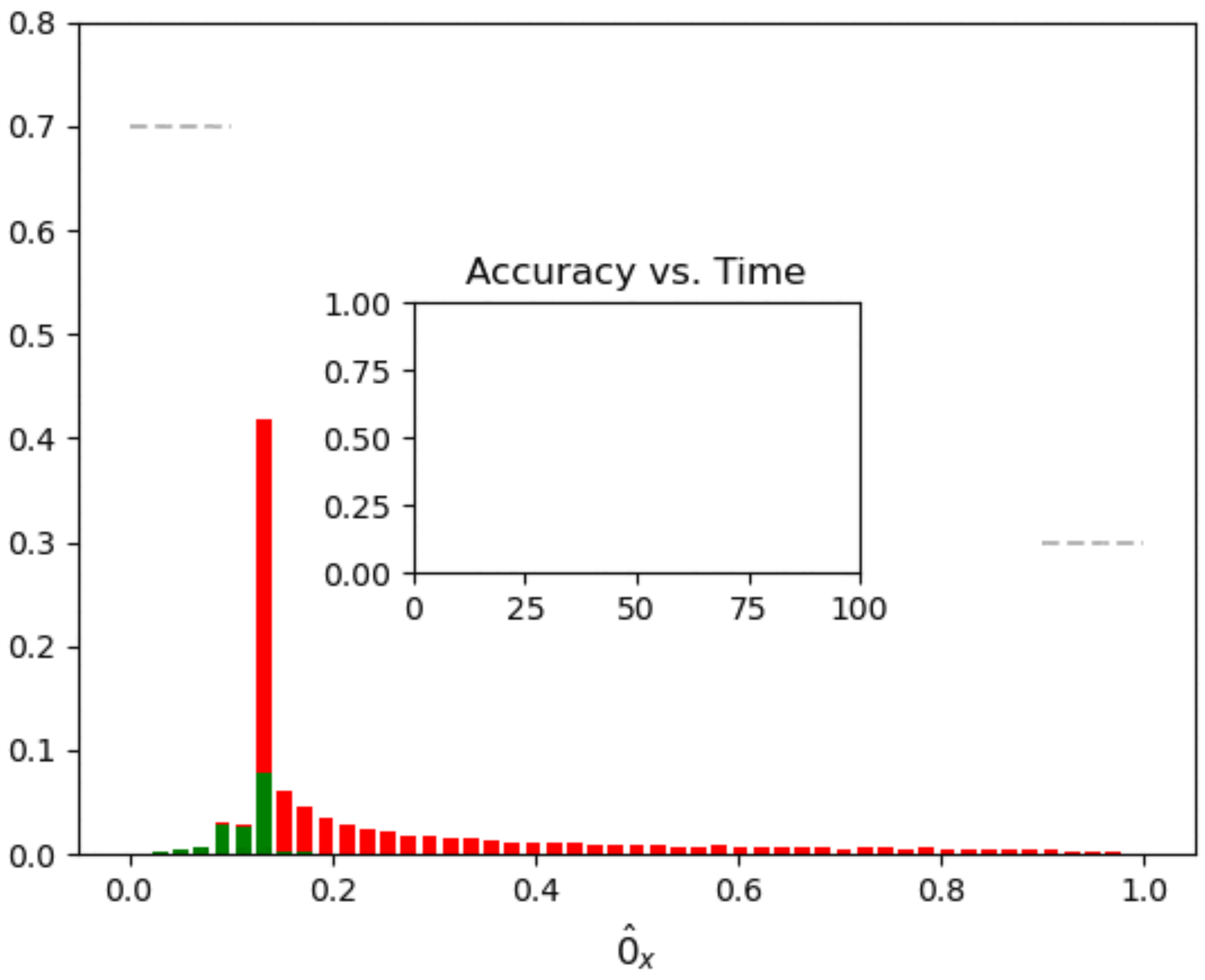
* WRN-40-2, CIFAR-10, Symmetric labeling, Standard augmentation

Estimated Distribution Metrics

Discussion

- This example* illustrates that ℓ_+ with $(k, c) = (2, 0.99)$ is robust to $r = 0.3$



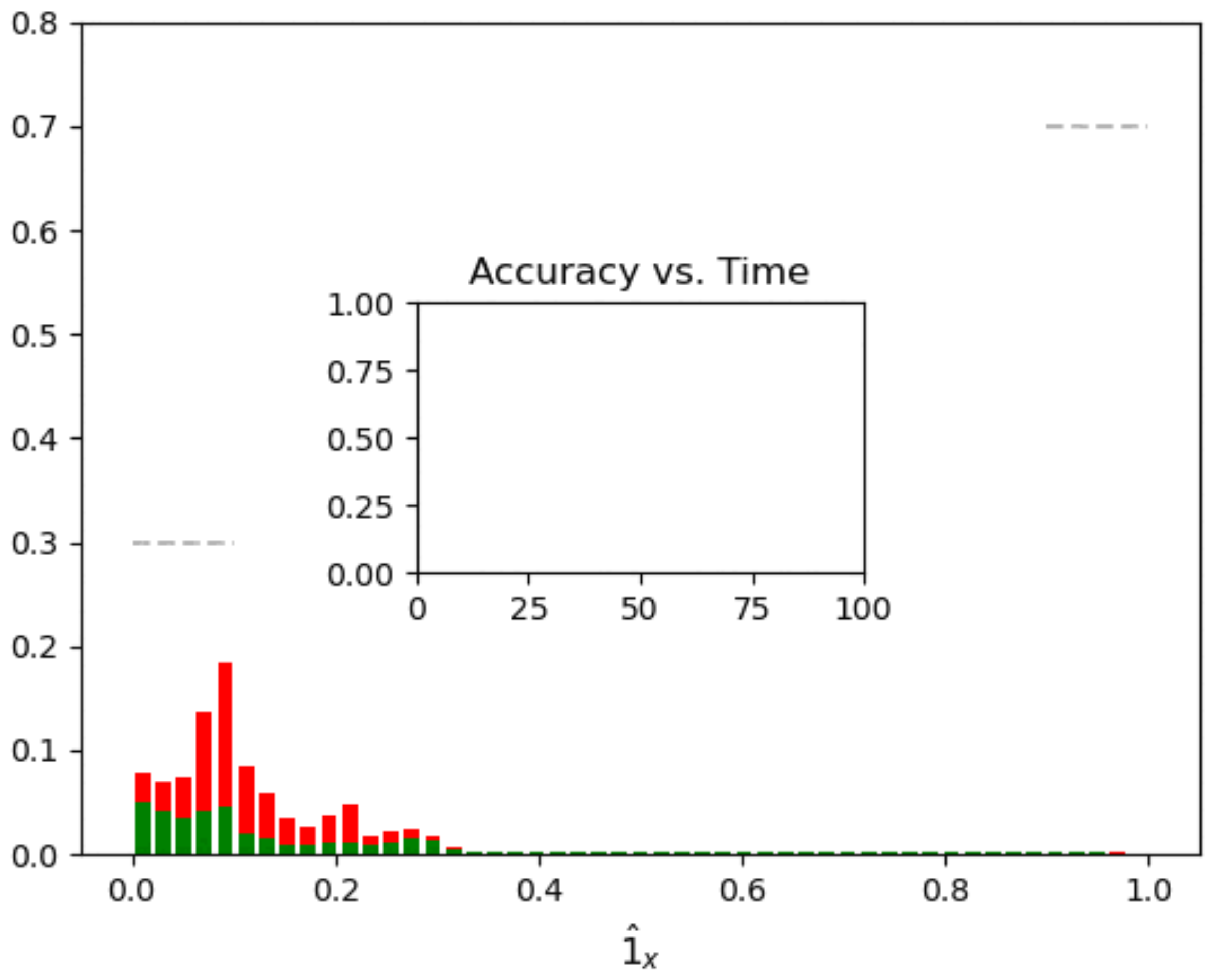


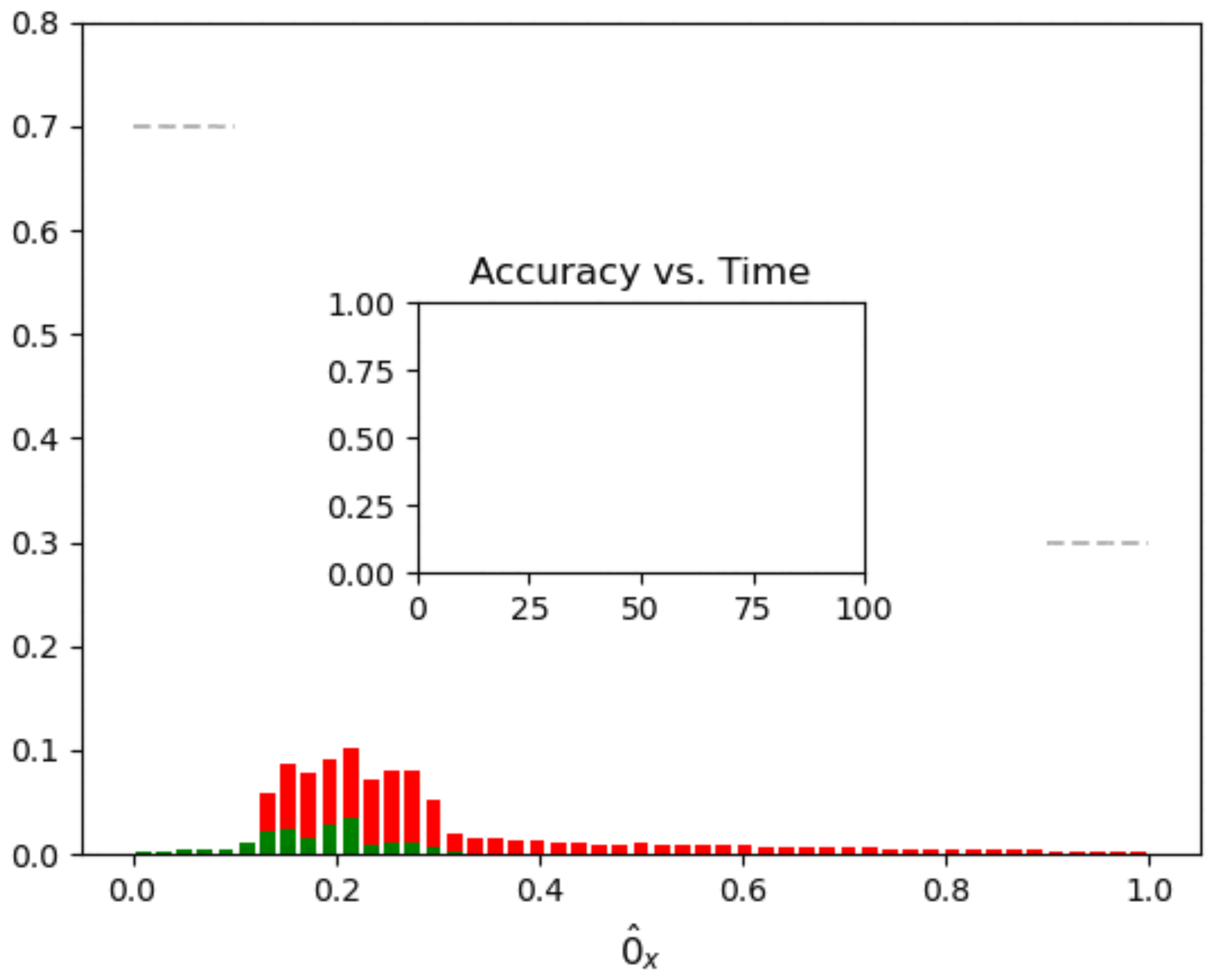
* WRN-40-2, CIFAR-10, Symmetric labeling, Standard augmentation

Estimated Distribution Metrics

Discussion

- Likewise, this example* illustrates that ℓ_α with $\alpha = 3.5$ is robust to $r = 0.3$



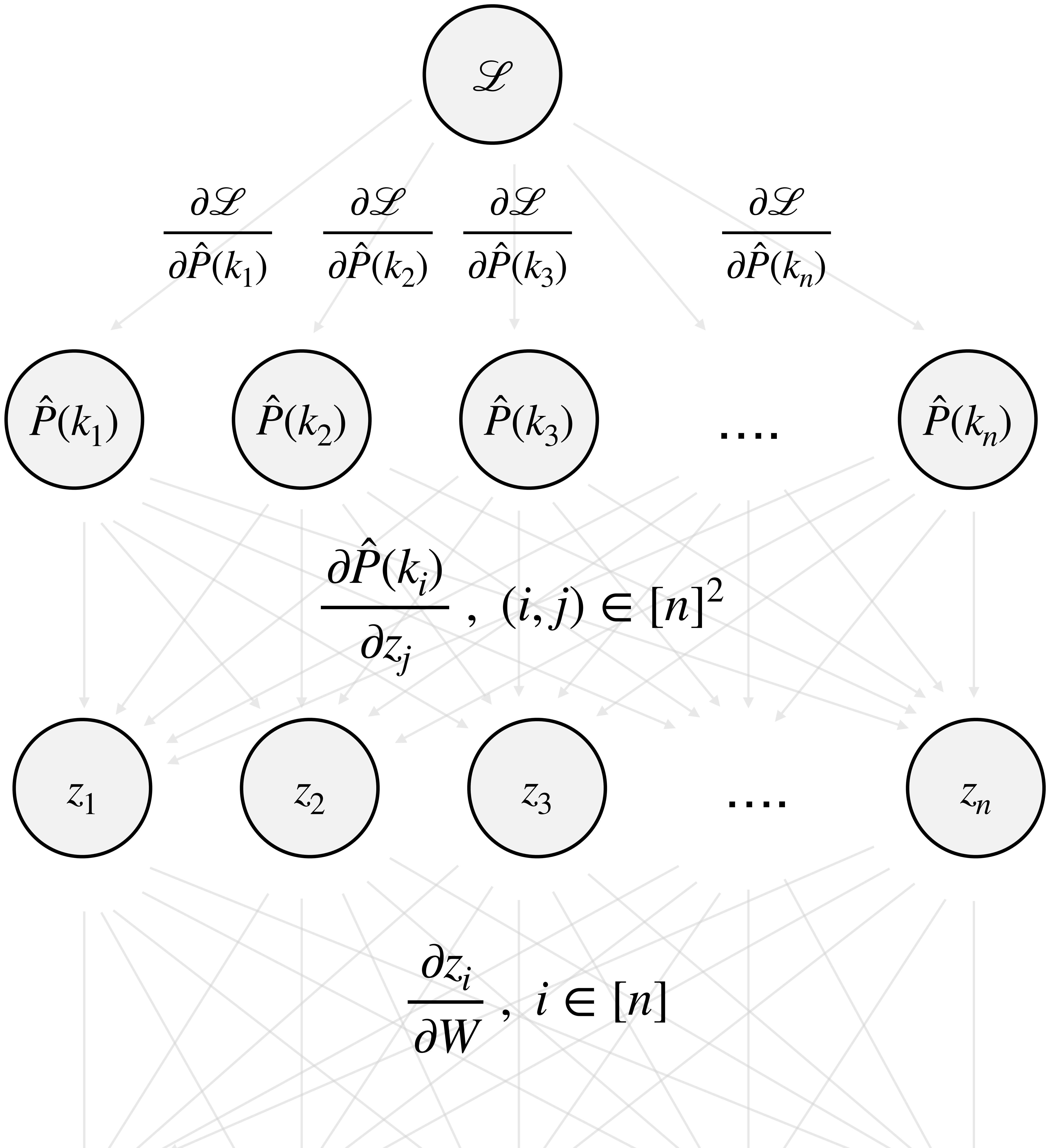


Agenda

- Brief introduction
- Background
  - Image classification
  - Robust loss functions
  - Data augmentation
- Empirical investigation
  - Motivation
  - Setting (control & variable)
  - Hyperparameter tuning
  - Results summary
- Discussion
  - Estimated distribution metrics
  - Backpropagation
- Takeaways
- Next step
- Q&A

Backpropagation
Discussion

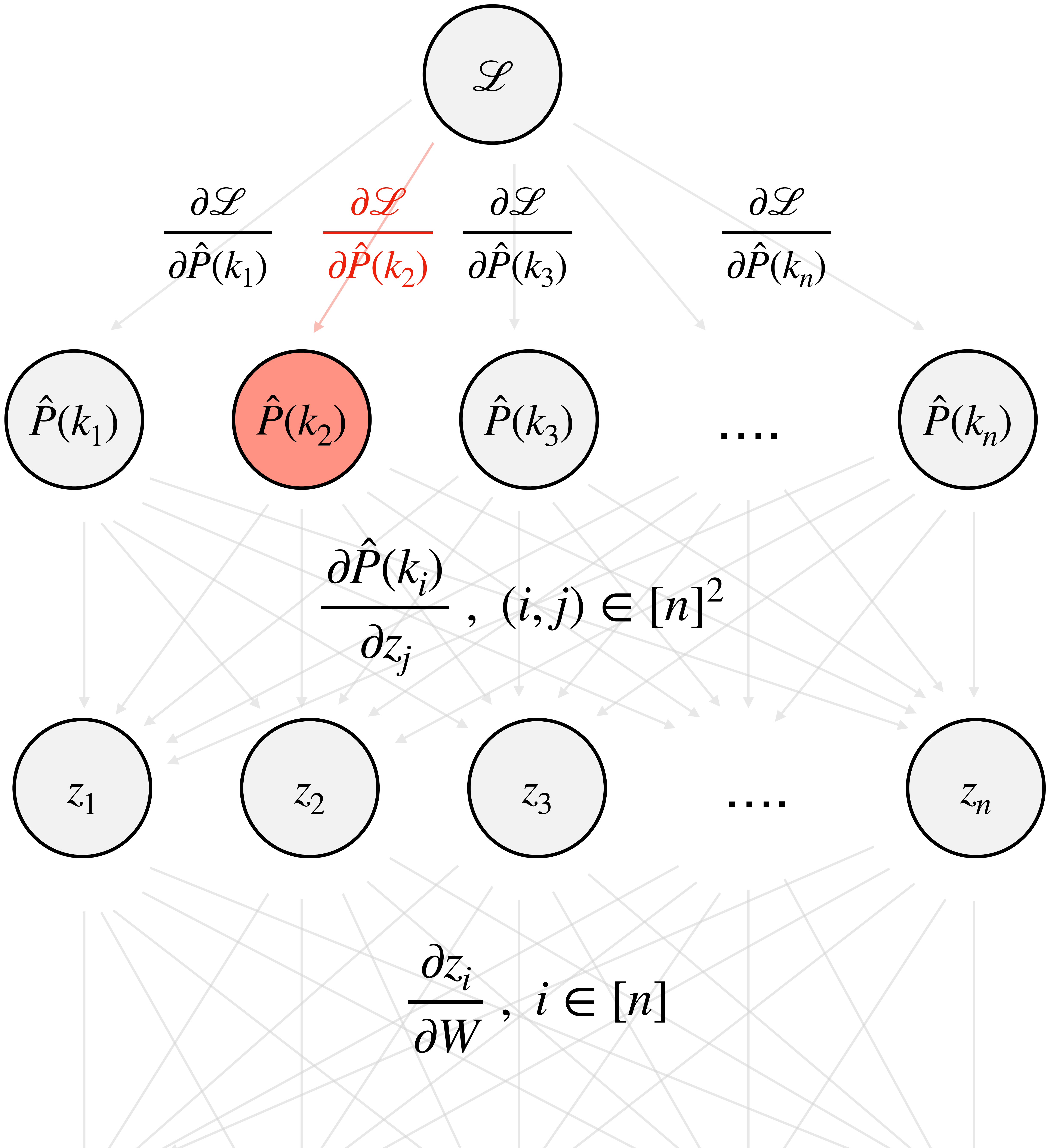
- At the top of the model, $\frac{\partial \mathscr{L}}{\partial \hat{P}(k_i)}$ is distributed to each node $\hat{P}(k_i)$ indicating **how $\hat{P}(k_i)$ should change in order to minimize $\mathscr{L}$**
- These partial derivatives ultimately contribute to the overall gradient $\frac{\partial \mathscr{L}}{\partial W}$ through the chain rule



Backpropagation

Discussion

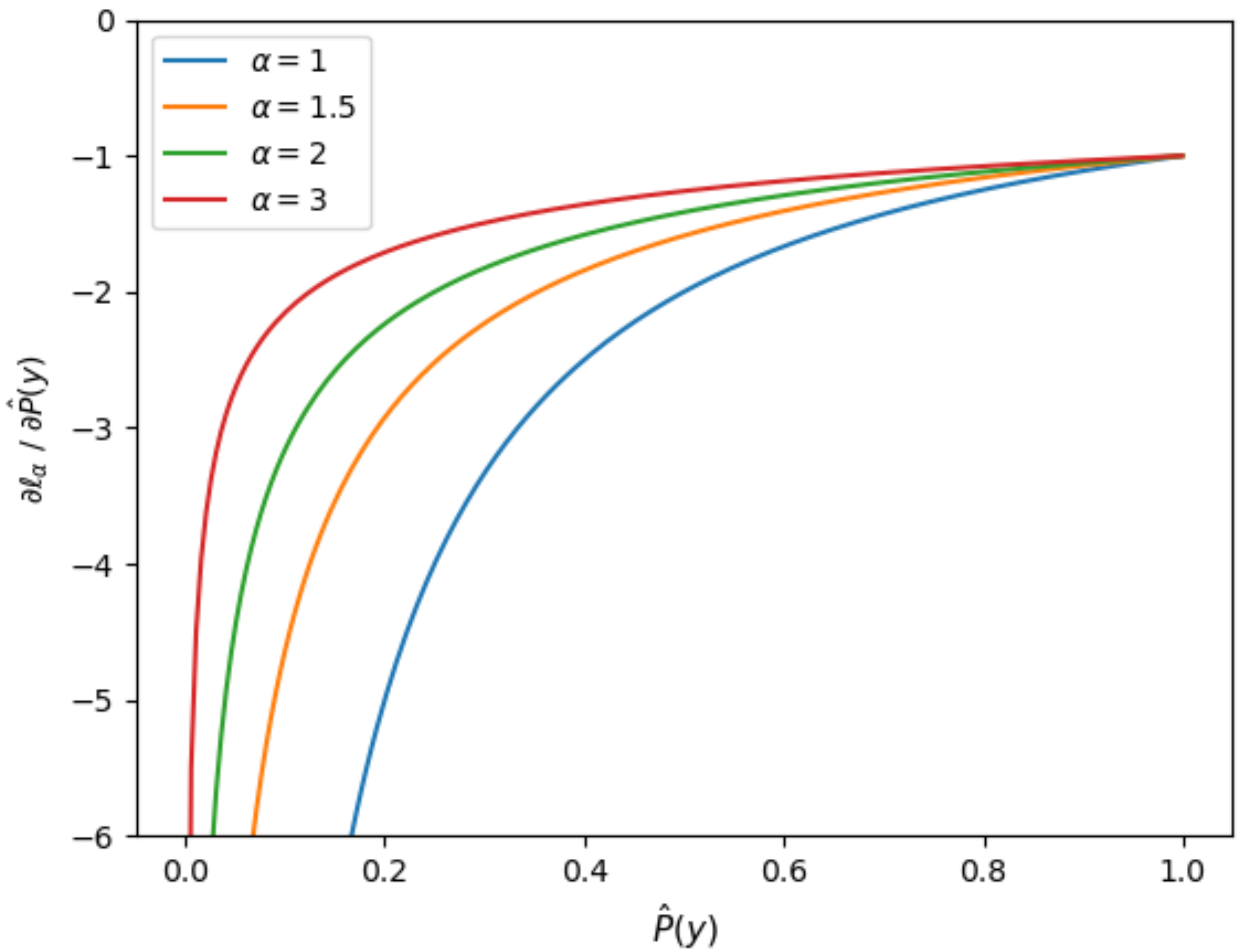
- If k_2 were to be a **false-positive class** (false class flipped to a true class), then we'd like for $\hat{P}(k_2)$ to stay low
  - Therefore $\frac{\partial \mathscr{L}}{\partial \hat{P}(k_2)}$ should be as high as possible



Backpropagation

Discussion

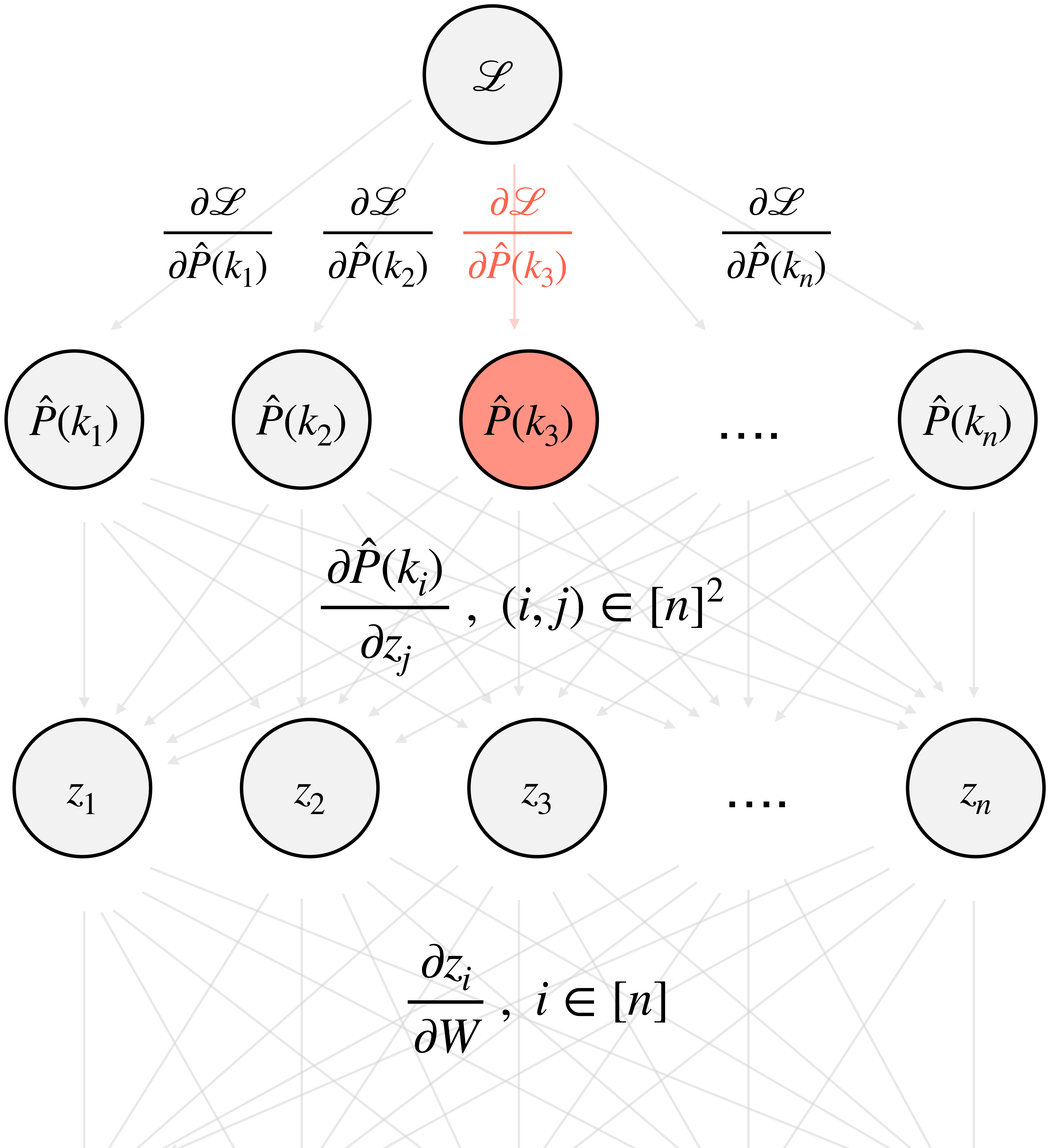
- Suppose $(x, y) \in \mathscr{D}$ and $\hat{f}(x) = \hat{P}$. Then
  - $\ell_\alpha(\hat{P}, y; \alpha) = \frac{\alpha}{\alpha - 1}\left(1 - \hat{P}(y)^{1-\frac{1}{\alpha}}\right)$
  - $\frac{\partial \ell_a}{\partial \hat{P}(y)}(\hat{P}, y; \alpha) = -\hat{P}(y)^{-\frac{1}{\alpha}}$
  - $\frac{\partial \ell_a}{\partial \hat{P}(y)}$ for $\alpha > 1$ purposely hinders the growth of $\hat{1}_x = \hat{P}(y)$ when y is a lesser-likely event ($\hat{1}_x \rightarrow 0$) and therefore more likely to be a false-positive class



Backpropagation

Discussion

- Now, if k_3 were to be a **false-negative class** (true class flipped to a false class), then we'd like for $\hat{P}(k_3)$ to stay high
    - Therefore $\frac{\partial \mathscr{L}}{\partial \hat{P}(k_3)}$ should be as low as possible



Backpropagation
Discussion

- Suppose $(x, y) \in \mathscr{D}$, $\hat{f}(x) = \hat{P}$, and let $k_0 \in \mathscr{Y}$ be the most likely false class. Then
  - $\ell_{NCE}(\hat{P}, y) = \dfrac{\log \hat{P}(y)}{\sum_{k \in \mathscr{Y}} \log \hat{P}(k)} = \dfrac{\log \hat{P}(y)}{\log \hat{P}(k_0) + \sum_{k \neq k_0} \log \hat{P}(k)}$
  - $\ell_{RCE}(\hat{P}, y) = A \sum_{k \neq y} \hat{P}(k) = A \cdot \hat{P}(k_0) + A \sum_{k \neq y, k_0} \hat{P}(k)$, where $A \in \mathbb{R}^+$ is some constant
  - Recall that $\ell_+(\hat{P}, y; \alpha, \beta) = \alpha \cdot \ell_{NCE}(\hat{P}, y) + \beta \cdot \ell_{RCE}(\hat{P}, y)$

Backprojection

Discussion

- If we fix $\hat{P}(k)$ for all $k \neq k_0$, then

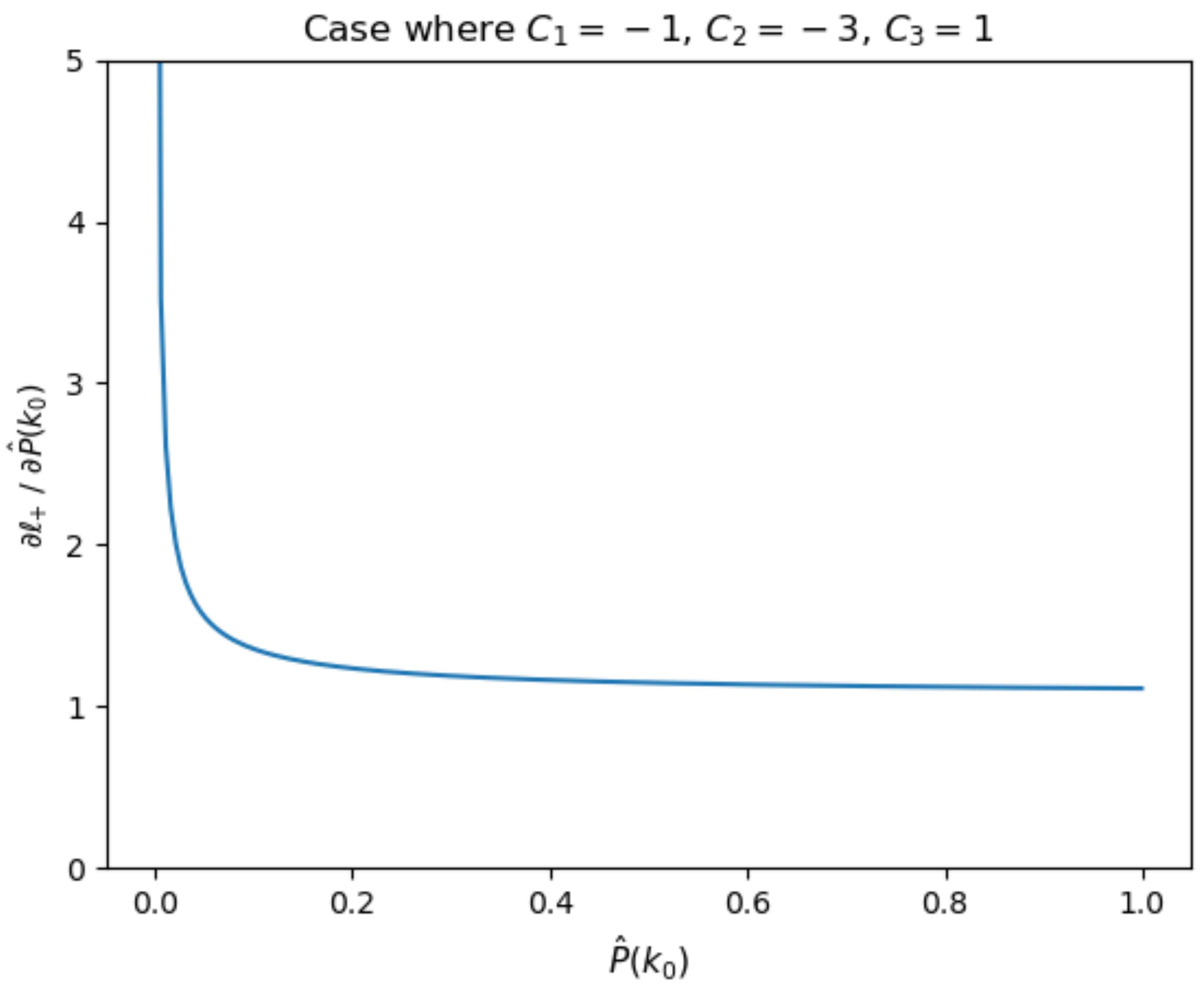
$$\ell_+(\hat{P}, y; \alpha, \beta) = \alpha \cdot \left(\frac{\log \hat{P}(y)}{\log \hat{P}(k_0) + \sum_{k \neq k_0} \log \hat{P}(k)} \right) + \beta \cdot \left(A \cdot \hat{P}(k_0) + A \sum_{k \neq y, k_0} \hat{P}(k) \right)$$

$$= \frac{C_1}{\log \hat{P}(k_0) + C_2} + C_3 \cdot \hat{P}(k_0) + C_4$$

for some constants $C_1, C_2 \in \mathbb{R}^-$, $C_3, C_4 \in \mathbb{R}^+$

Backpropagation

Discussion



- Then the partial derivative of ℓ_+ w.r.t. $\hat{P}(k_0)$ is

$$\frac{\partial \ell_+}{\partial \hat{P}(k_0)}(\hat{P}, y; \alpha, \beta) = -\frac{C_1}{x\left(\log \hat{P}(k_0) + C_2\right)^2} + C_3$$

- $\frac{\partial \ell_+}{\partial \hat{P}(k_0)}$ is a decreasing function, which hinders the decay of $\hat{0}_x = \hat{P}(k_0)$ when k_0 is a more-likely event ($\hat{0}_x \to 1$) and therefore more likely to be a false-negative class

Backpropagation

Discussion

- To summarize:

1. The α*-loss* family is robust to label corruption because ℓ_α restrains the growth of probabilities for potential false-positive classes

2. The *NCE+RCE* family is robust to label corruption because ℓ_+ restrains the decay of probabilities for potential false-negative classes

Agenda

- Brief introduction
- Background
    - Image classification
    - Robust loss functions
    - Data augmentation
- Empirical investigation
    - Motivation
    - Setting (control & variable)
    - Hyperparameter tuning
    - Results summary
- Discussion
    - Estimated distribution metrics
    - Backpropagation
- **Takeaways**
- Next step
- Q&A

Takeaways

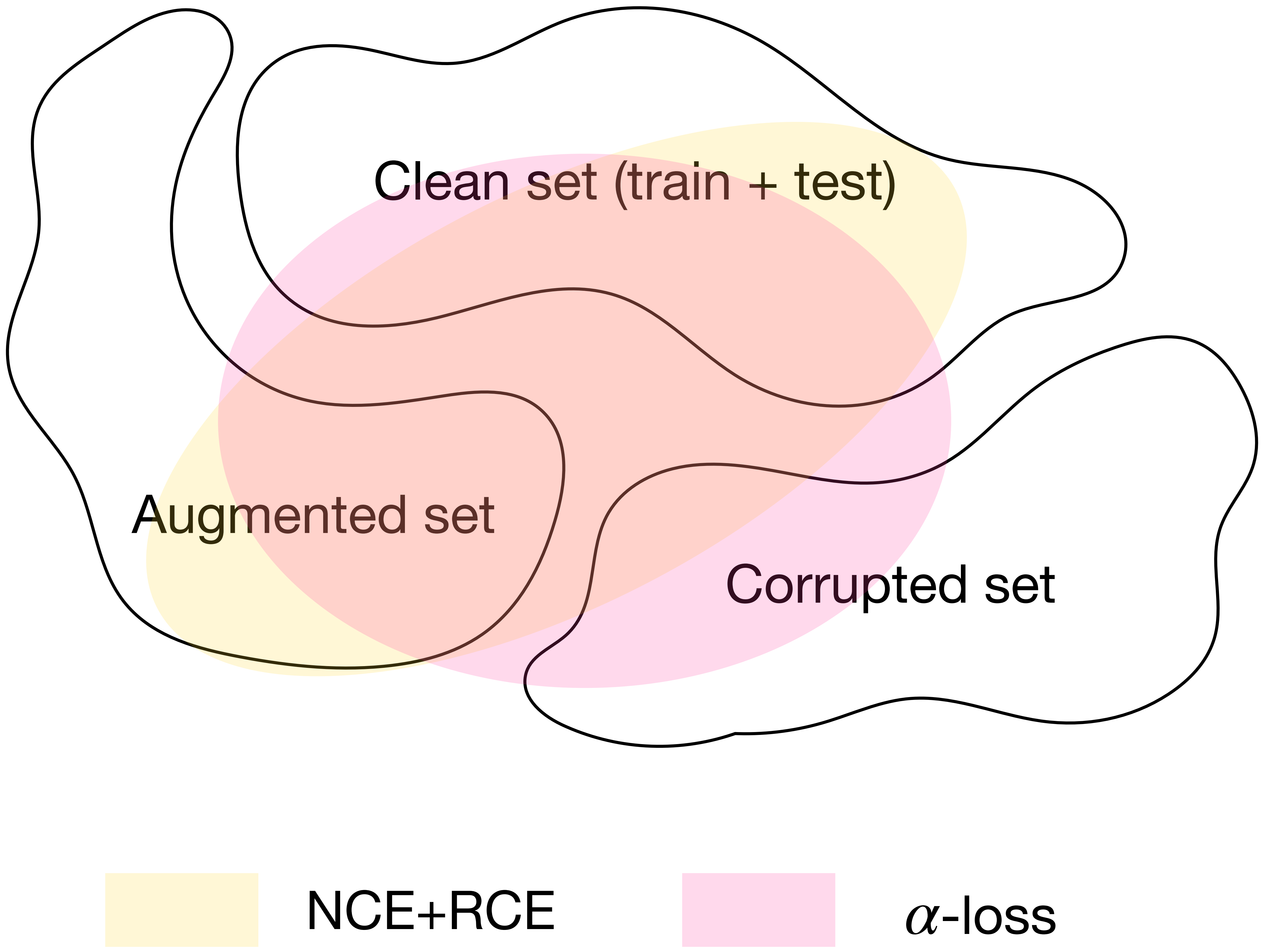
- *α-loss* requires a much-less involved optimal hyperparameter search than that of its SoTA loss function family counterpart, *NCE+RCE*
- When training on corrupted labels, *α-loss* is competitive with *NCE+RCE*
- When evaluating on corrupted features, data augmentation is essential for optimal performance
- When training on corrupted labels AND evaluating on corrupted features, *α-loss* appears to slightly (but consistently) outperform *NCE+RCE*
- The optimality of ℓ_{base} (w.r.t. test performance) does indeed depend on the choice of $\mathscr{A}$

Agenda

- Brief introduction
- Background
    - Image classification
    - Robust loss functions
    - Data augmentation
- Empirical investigation
    - Motivation
    - Setting (control & variable)
    - Hyperparameter tuning
    - Results summary
- Discussion
    - Estimated distribution metrics
    - Backpropagation
- Takeaways
- **Next step**
- Q&A

Next Step

- Give more consideration into why α-loss appears to consistently outperform NCE+RCE in our designed setting
- Potential explanation:
  - Although both families succeed at learning the clean train set, NCE+RCE slightly overfits on the clean-augmented hybrid distribution while α-loss generalizes to the overall distribution



Agenda

- Brief introduction
- Background
  - Image classification
  - Robust loss functions
  - Data augmentation
- Empirical investigation
  - Motivation
  - Setting (control & variable)
  - Hyperparameter tuning
  - Results summary
- Discussion
  - Estimated distribution metrics
  - Backpropagation
- Takeaways
- Next step
- Q&A

Q&A

The End

Thanks for viewing!