

CENG7880

Trustworthy and Responsible AI

Instructor: Sinan Kalkan

(<https://ceng.metu.edu.tr/~skalkan>)

For course logistics and materials:

<https://metu-trai.github.io>

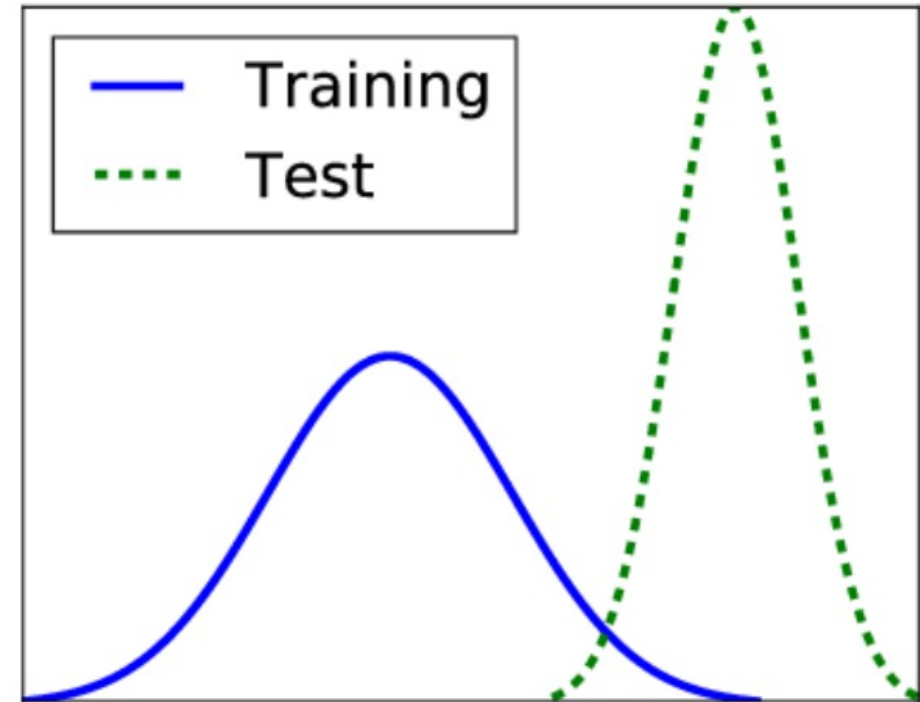


MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING



Support of Shifted Data

- **Assumption:** Support of Q is not contained in support of P
- However, this is **necessary** since we do not know anything about data outside of the support of P
- Need additional assumptions to do better
 - Focus on covariate shift



Image; Glauner et al., 2018

Learning vs. Evaluation

- For this part, we will focus on **model evaluation**

- **Learning:** Optimize $\mathbb{E}_Q[\ell(\theta; x, y)]$
- **Evaluation:** Estimate $\mathbb{E}_Q[\ell(\theta; x, y)]$

- We will see why learning is harder later

Evaluation Bounds for Covariate Shift

- Thus, we have

$$\mathbb{E}_Q[\ell(\theta; x, y)] \leq \mathbb{E}_P[\ell(\theta; x, y)] + K_{\tilde{\ell}} \cdot W(P(x), Q(x))$$

Previously on CENG7880!

Previously on CENG7880!

Solutions to Optimizing the Upper Bound

$$\mathbb{E}_Q[\ell(\theta; x, y)] \leq \mathbb{E}_P[\ell(\theta; x, y)] + K_{\tilde{\ell}} \cdot W(P(x), Q(x))$$

1. Distributionally Robust Optimization: Assume a family/ball of plausible Q and try to minimize data loss for that family of Q as well.

Deterministic Optimization

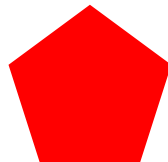
$$\min_{\theta} \ell(x; \theta)$$



x

Robust Optimization

$$\min_{\theta} \max_{x \in X} \ell(x; \theta)$$

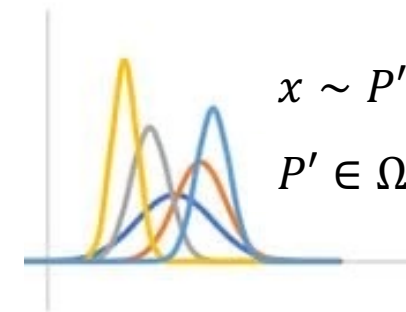


$x \in X$

X : formally
characterizable changes

Distributionally Robust Optimization

$$\inf_{\theta} \sup_{P' \in \Omega} \mathbb{E}_{P'}[\ell(x; \theta)]$$



$x \sim P'$

$P' \in \Omega$

Previously on CENG7880!

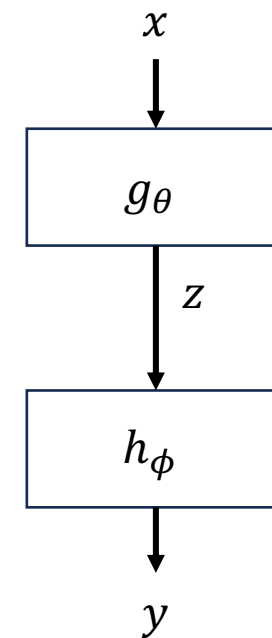
Solutions to Optimizing the Upper Bound

$$\mathbb{E}_Q[\ell(\theta; x, y)] \leq \mathbb{E}_P[\ell(\theta; x, y)] + K_{\tilde{\ell}} \cdot W(P(x), Q(x))$$

2. Formulate $W()$ in the latent feature space, instead of raw input.

- For a fixed input space $\mathcal{X}$, split the model into a feature extractor $g_\theta: \mathcal{X} \rightarrow \mathcal{Z}$ and a classifier/prediction head $h_\phi: \mathcal{Z} \rightarrow \mathcal{Y}$.
- The bound is then applied in the latent feature space $\mathcal{Z}$, turning the distance $W(g_\theta(x_P), g_\theta(x_Q))$ into a function of the parameter θ :

$$\min_{\theta, \phi} \mathbb{E}_P[\ell(h_\phi(g_\theta(x)), y)] + \lambda \cdot W(g_\theta(x_P), g_\theta(x_Q))$$



Methods: Wasserstein Distance Guided Representation Learning, Domain-Adversarial Neural Networks, ...

Previously on CENG7880!

Solutions to Optimizing the Upper Bound

$$\mathbb{E}_Q[\ell(\theta; x, y)] \leq \mathbb{E}_P[\ell(\theta; x, y)] + K_{\tilde{\ell}} \cdot W(P(x), Q(x))$$

3. Invariant Risk Minimization:

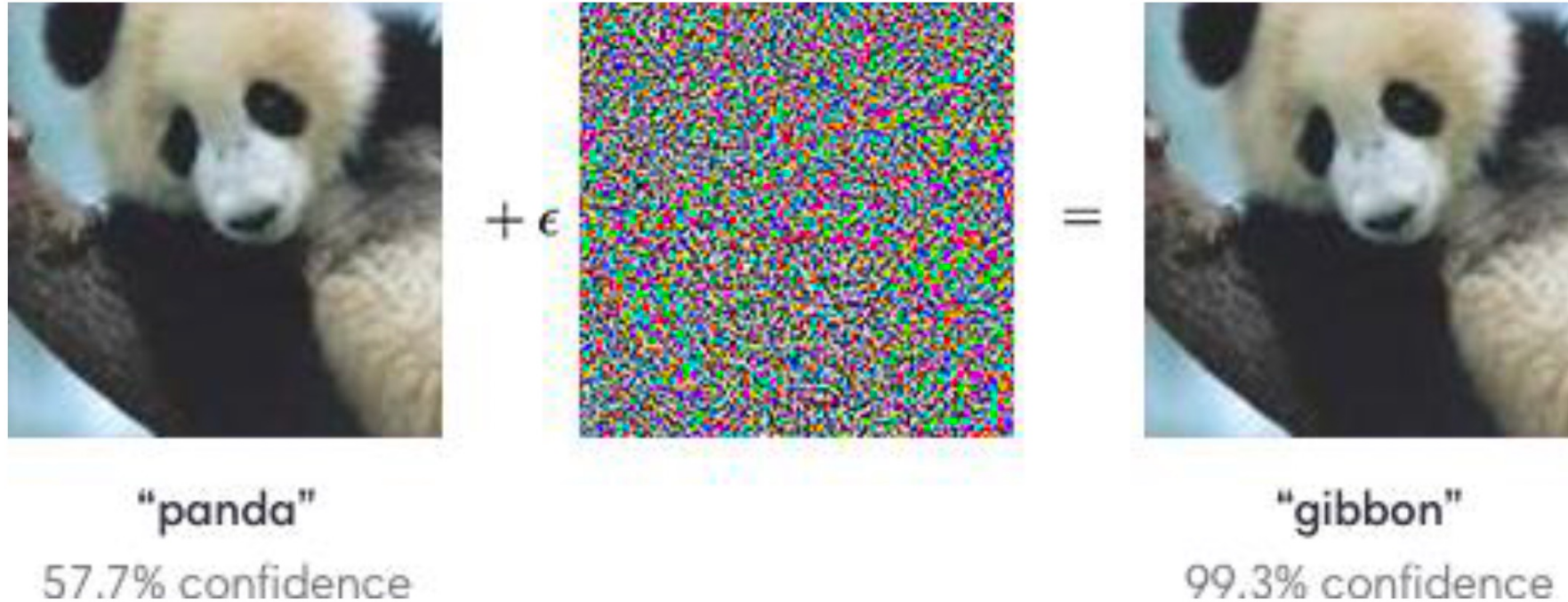
- Learn causal factors/variables/features $z = g_\theta(x)$ such that a linear classifier $h_\phi(z)$ constructed on top of z is optimal across different distributions/environments.

$$\min_{\theta, \phi} \sum_{e \in \mathcal{E}_{tr}} (\mathbb{E}_e[\ell(h_\phi(g_\theta(x)), y)] + \lambda \cdot \|\nabla_{w|w=1.0} \mathbb{E}_e[\ell(w \cdot h_\phi(g_\theta(x)), y)]\|^2) + K \cdot W(g_\theta(x_{\mathcal{E}_{tr}}), g_\theta(x_Q))$$

Minimize prediction loss & its weighted form over different environments/domains

Previously on CENG7880!

Deep networks are “sensitive”

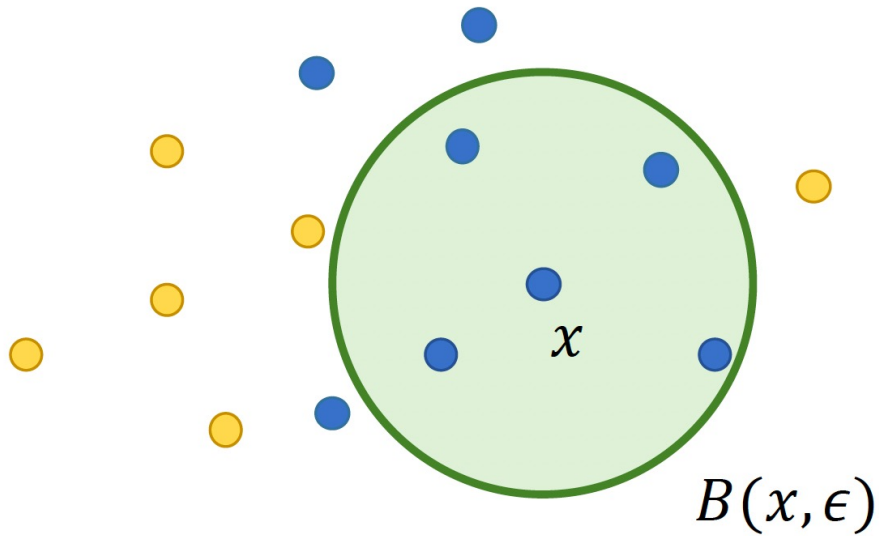


Szegedy et al., Intriguing Properties of Neural Networks, 2014

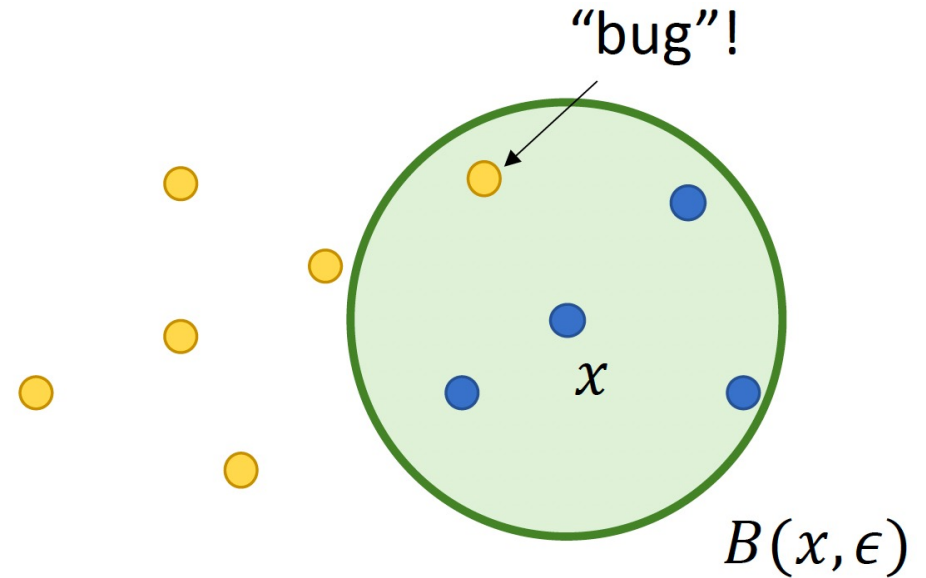
Previously on CENG7880!

robustness: similar images $\Rightarrow$ same label

robustness: $\|x - x'\|_{\infty} \leq \epsilon \Rightarrow$ same label



ϵ -robust at x



not ϵ -robust at x

How to implement desired gradient descent ?

- Key step in learning:

Computing the gradient $\nabla_{\theta} \text{Loss}(x, y; \theta)$ using **backpropagation**

- Question: How will you compute $\nabla_{\delta} \text{Loss}(x + \delta, y; \theta)$?

Recall we are optimizing

$$\max_{\delta \in \Delta} \text{Loss}(x + \delta, y; \theta)$$

- Question: To find the (locally) optimal value, is it ok to repeatedly update x to $x + \alpha \nabla_{\delta} \text{Loss}(x + \delta, y; \theta)$, where α is the learning rate?
- No! We want $\delta < \Delta$, so this is an instance of “constrained optimization”

Projected gradient descent

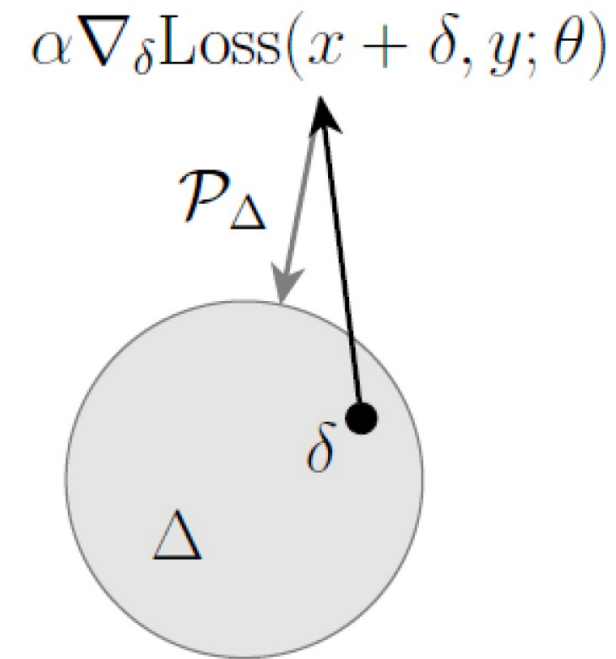
Recall we are optimizing

$$\max_{\delta \in \Delta} \text{Loss}(x + \delta, y; \theta)$$

We can employ a projected gradient descent method:

- take gradient step and
- project back into feasible set Δ

$$\delta := \mathcal{P}_{\Delta}[\delta + \nabla_{\delta} \text{Loss}(x + \delta, y; \theta)]$$



Source: Tutorial on Adversarial robustness by Kolter and Madry

Fast Gradient Sign Method (FGSM)

Note that:

$$\|\mathbf{x}\|_{\infty} := \max_i |x_i|$$

To be more concrete,

- take Δ to be the ℓ_{∞} ball:

$$\Delta = \{\delta: \|\delta\|_{\infty} \leq \epsilon\}$$

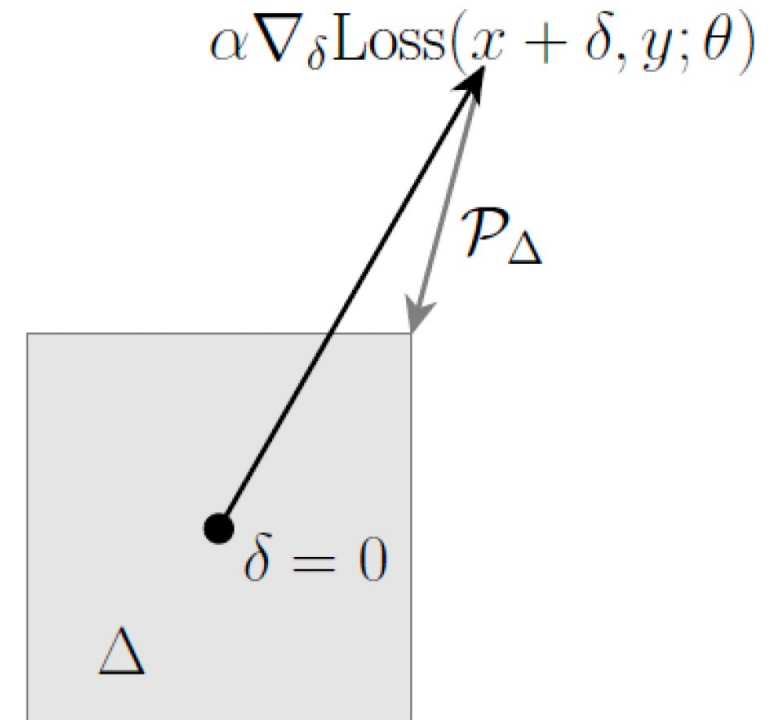
- then, projection takes the form

$$P_{\Delta}(\delta) = \text{Clip}(\delta, [-\epsilon, \epsilon])$$

As $\alpha \rightarrow \infty$, we always reach “corner” of the box, called fast gradient sign method (FGSM)

[Goodfellow et al., 2014]

$$\delta = \epsilon \cdot \text{sign}(\nabla_{\delta} \text{Loss}(x + \delta, y; \theta))$$



Source: Tutorial on Adversarial robustness by Kolter and Madry

Previously on CENG7880!

Targeted Attack

Also possible to explicitly try to change label to a *particular* class

$$\max_{\delta \in \Delta} (\text{Loss}(x + \delta, y; \theta) - \text{Loss}(x + \delta, y_{\text{targ}}; \theta))$$

Consider multi-class cross entropy loss

$$\text{Loss}(x + \delta, y; \theta) = \log \left(\sum_i \exp h_{\theta}(x + \delta)_i \right) - h_{\theta}(x + \delta)_y$$

Then, combining the two, we obtain:

$$\max_{\delta \in \Delta} \left(h_{\theta}(x + \delta)_{y_{\text{targ}}} - h_{\theta}(x + \delta)_y \right)$$

Source: Tutorial on Adversarial robustness by Kolter and Madry

Adversarial Training Algorithm

Repeat:

1. Select a minibatch B
2. For each (x, y) in B , compute the adversarial example $\delta^*(x)$

Recall FGSM method of steepest descent to compute adversarial examples:

$$\delta^* = \epsilon \cdot \text{sign}(\nabla_{\delta} \text{Loss}(x + \delta, y; \theta))$$

3. Update parameters:

$$\theta := \theta - \frac{\alpha}{|B|} \sum_{x, y \in B} \nabla_{\theta} \text{Loss}(x + \delta^*(x), y; \theta)$$

Note: in practice, one can mix standard updates and adversarial updates

Smoothed classifier

- Given a base classifier f , its smoothed version g maps an input x to the majority prediction of f on many Gaussian-perturbed images $x + \eta$:

$$g(x) = \arg \max_y \mathbb{P}_\eta[f(x + \eta) = y]$$

Voting

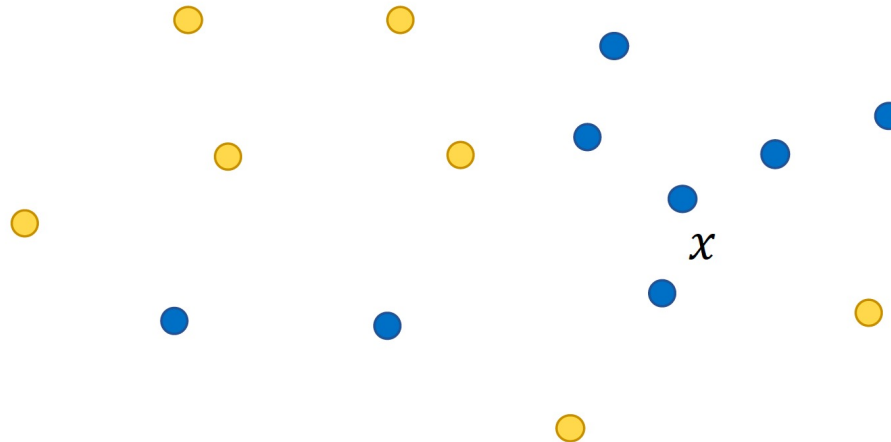
Probability that the base classifier estimates class y for the Gaussian-perturbed image. Can be estimated with N perturbations as follows:

$$\hat{\mathbb{P}}_\eta[f(x + \eta) = y] \approx \frac{\text{Count of times } f(x + \eta_i) = y}{N}$$

Previously on CEN 7880!

Randomized Smoothing

- Method works for an arbitrary f , including complex neural networks
- The smoothed version g of a given classifier f turns out to be empirically robust
- The bound Δ on adversarial robustness radius is related to the parameter σ in Gaussian noise
- Intuitively: large random noise can be used to drown out small adversarial perturbation



Previously on CENG7880!

Randomized Smoothing Guarantee

Certified robust radius by [Cohen et al.'19] (for binary classification)

- Given any input $x \in \mathbb{R}^d$, let
 - η be Gaussian noise $\mathcal{N}(0, \sigma^2 I)$
 - $p = \max_y \mathbb{P}_\eta[f(x + \eta) = y]$ $\Rightarrow$ Confidence of majority vote
- Then $g(x) = g(x + \delta)$ for any δ such that $\|\delta\|_2 \leq \Phi^{-1}(p)\sigma$,
 - where Φ is CDF of standard Gaussian.

Recall:

$$g(x) = \operatorname{argmax}_y \mathbb{P}_\eta [f(x + \eta) = y]$$

Computable certified radius for x

Certified Robustness Tutorial

https://colab.research.google.com/drive/1yclWTo1VkuF4w8GxdfIEgRJm0hszR_AW#scrollTo=3959f522

by Ugur Yalcin

<https://drive.google.com/file/d/1JGUBL5-KCmDRzAW6BMSrRlOyc5wqqoKr/view?usp=sharing>

by Melih Darcan

Agenda

- Robustness
 - Adversarial Defenses
 - Verifying Robustness
 - Uncertainty
 - Calibration
 - Conformal Prediction
 - Uncertainty Types & Quantification (next week)

Administrative Notes

- Selecting papers
 - ~~Form: <https://forms.gle/Z8SFya9hQtdzP7jG6>~~
 - ~~Deadline for choosing papers: 16 October.~~

Adversarial Defenses

Motivation for Adversarial Defense Mechanisms

- Limitations of Adversarial Training / Robustness
 - Accuracy vs robustness trade-off
 - Computationally expensive:
 - Adversarial training requires generating adversarial examples through gradient updates
 - If you train a network to be robust against a type of attack, we can come up with another type of attack.
- Alternative:
 - “Tackle” perturbation before processing it by the ML model.

Adversarial Defense Mechanisms

- Option 1: Detect attacks before feeding them to the model
 - Consider perturbed images as out-of-distribution
 - E.g., train an encoder-decoder network on “clean” input and use reconstruction error as a sign of “anomaly”.
- Option 2: Purify perturbed inputs
 - Remove perturbations/attacks
 - Transform perturbed input to its closest “clean” version
 - E.g., use denoising autoencoders

Wang, Y., Sun, T., Li, S., Yuan, X., Ni, W., Hossain, E., & Poor, H. V. (2023). Adversarial Attacks and Defenses in Machine Learning-Powered Networks: A Contemporary Survey. arXiv preprint arXiv:2303.06302.

Denoised Smoothing

Salman et al., “Denoised Smoothing: A Provable Defense for Pretrained Classifiers”, NeurIPS 2020.

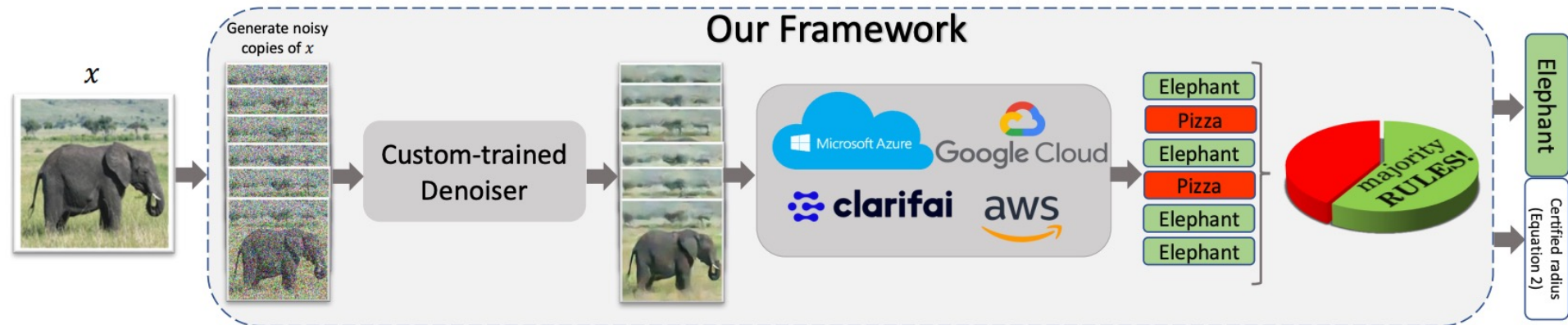
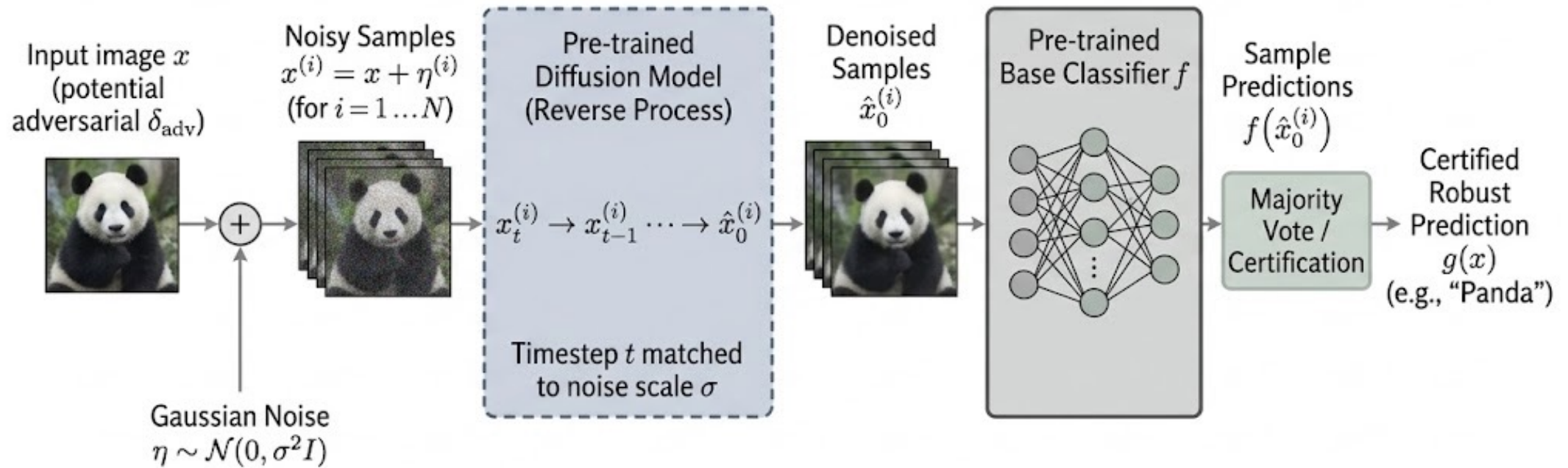


Figure 1: Given a clean image x , our denoised smoothing procedure creates a smoothed classifier by appending a denoiser to any pretrained classifier (e.g. online commercial APIs) so that the pipeline predicts in majority the correct class under Gaussian noise corrupted-copies of x . The resultant classifier is *certifiably* robust against ℓ_2 -perturbations of its input.

Diffusion Denoised Smoothing

Carlini et al., “(Certified!!) Adversarial Robustness for Free!”, ICLR 2023.

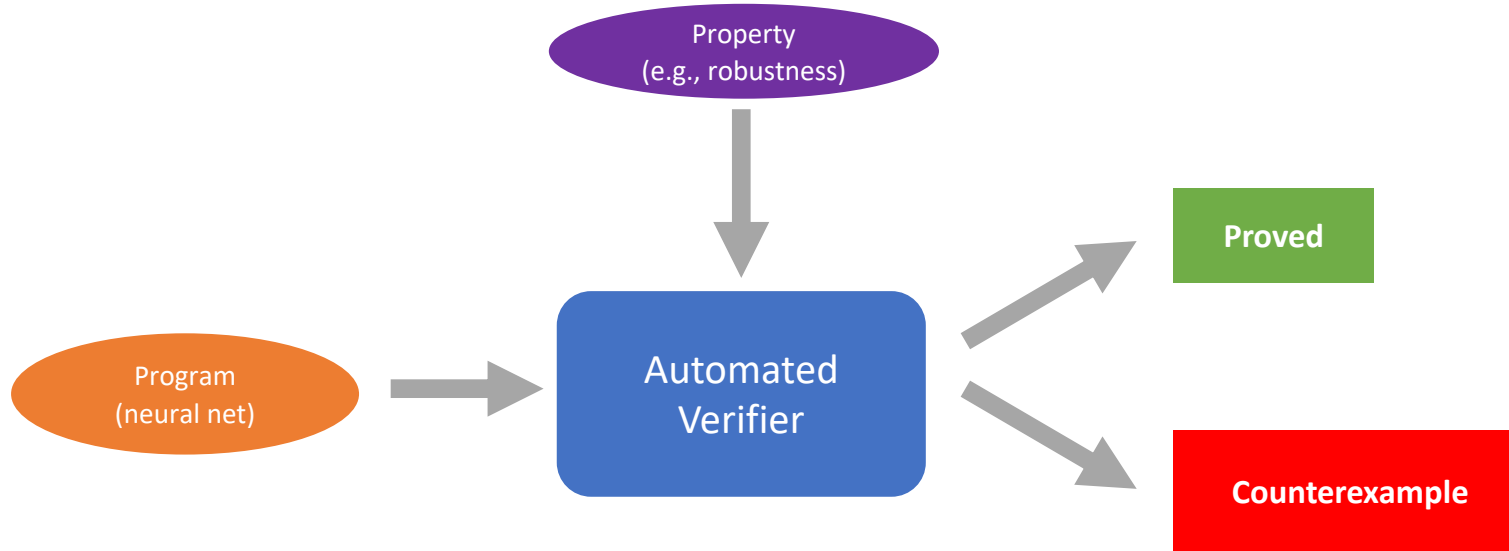


Adversarial Defense Mechanisms: Purification Methods

Wang, Y., Sun, T., Li, S., Yuan, X., Ni, W., Hossain, E., & Poor, H. V. (2023). Adversarial Attacks and Defenses in Machine Learning-Powered Networks: A Contemporary Survey. arXiv preprint arXiv:2303.06302.

Defense	Attack type	Brief description	Input	Invisibility Metric	Strength	Weakness
XEnsemble [198]	White-box, OOD	Improve DNN robustness against OOD inputs and adversarial examples by using diversity ensemble verification.	Image	ℓ_p	Automatically validate any input to the predictive model, be attack-agnostic. Superior in robustness and defensiveness, with a high defense rate of adversarial samples and a high detection rate of OOD inputs.	Need to include randomization at the input denoising integration layer and the output model validation layer to increase resistance to internal attacks, as well as expand to new media.
SAD [202]	White-box, Black-box	Achieve adversarial robustness via analyzing inputs using saliency maps and updating BN statistics without AT.	Image	ℓ_2	Widen the average distance between the processed data and the updated decision boundary, significantly smooth the landscape, and be more effective than AT. Reduce the training time significantly, not rely on gradient masking.	Need to fine-tune the processed data and adjust the significance map intensity related to the sample to further improve the performance.
Image super-resolution [203]	White-box, Black-box, Gray-box	A super-resolution-based image restoration technique that projects off-the-manifold adversarial samples into the benign image manifold.	Image	ℓ_1	No need for training or tuning many hyper-parameters. Do not cause gradient masking. Perform well for both black-box and white-box attacks. Support unknown attacks.	Its robustness against white-box attacks is weaker than that in the gray-box settings.
CAP-GAN [194]	White-box, Black-box	Use the pixel-level and feature-level consistency for GAN's cycle-consistent learning to achieve adequate purification.	Image	KL	The introduction of feature-level items fairly enhance model robustness. In both black-box and white-box conditions, CAP-GAN beats alternative preprocessing-based defenses on the CIFAR-10 dataset.	Adversarial interference is mitigated at the cost of removing important information from clean images, making DNN models less accurate for clean samples.
CAFD [196]	White-box, Unseen Attack	Devise a self-supervised AT technique in the class activation feature space to eliminate adversarial noise.	Image	CAFA	Compared to previous SOTA methods, the confrontation robustness is significantly enhanced, especially against unknown adversarial and adaptive attacks.	For white-box attacks, the defense model is completely leaked to the adversarial, and the protection capability of the defense model is destroyed. The defense against white-box adaptive attacks needs to be strengthened.
D2Defend [166]	White-box	Maintain the essential high-frequency image structure and filter out adversarial perturbations.	Image	SSIM	It is independent of DNN models and deploy-friendly. Good transferability among different commonly used networks and adversarial attack methods. The clean accuracy degradation is acceptable. More stable performance.	The defense effect under C&W attack is not optimal.

Robustness Verification

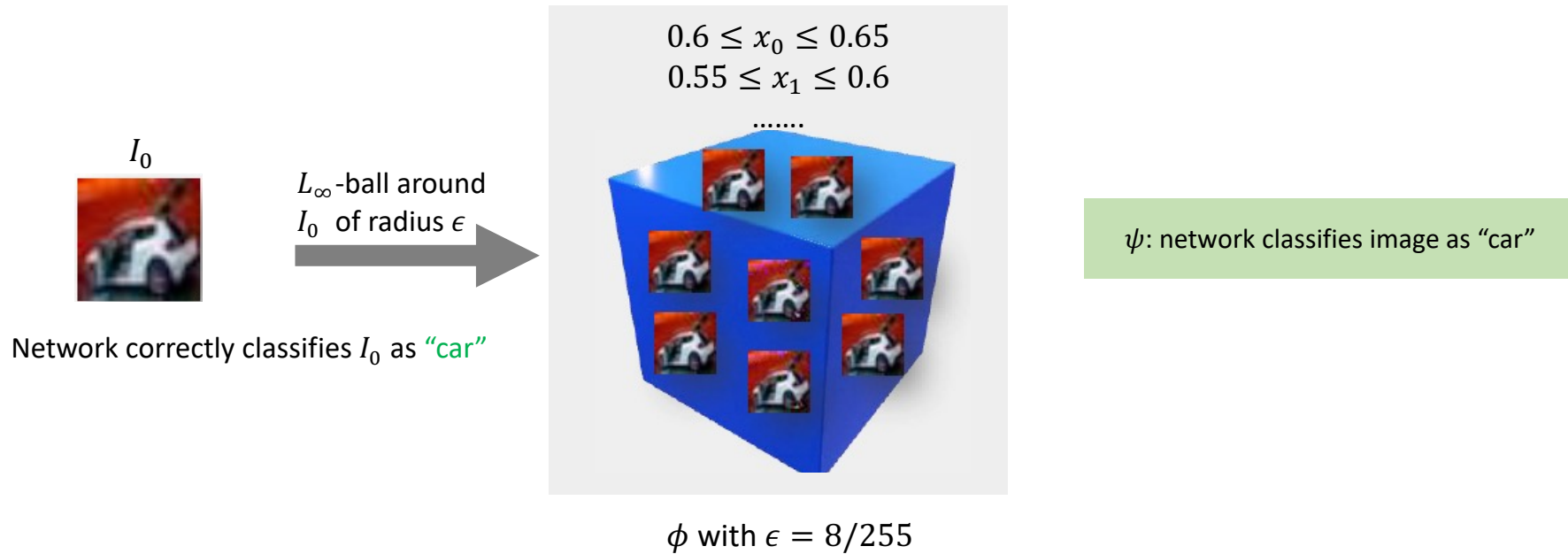


The general problem is computationally intractable, therefore we may need to provide an approximate answer

▪ Acknowledgement for slides:

- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

Robustness against adversarial perturbations



▪ Acknowledgement for slides:

- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

Verifying Programs: Specifications for programs

Input: int x , int y

Output: int r

$z := x + y;$

$z' := x - y;$

$r := z * z'$

Post-condition / assertion / "ensures" (in Dafny): $r = x^2 - y^2$

Cannot define correctness without specifications (logical constraints)

▪ Acknowledgement for slides:

- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

Verifying Programs:

Verification Conditions

Input: int x, int y

Output: int r

$z := x + y;$

$z' := x - y;$

$r := z * z'$

Ensures $r = x^2 - y^2$

Verifying correctness means checking validity of the following logical formula, called VC (Verification Condition):

$$(z = x+y \ \&\& \ z' = x-y \ \&\& \ r = z*z') \rightarrow r = x^2 - y^2$$

▪ Acknowledgement for slides:

- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

Verifying Programs: Logic for Specifications

Specification is a logical formula constructed from expressions over program variables using logical operators AND $\&\&$, OR $|$, NOT $\sim$

Different choices based on:

- Allowed types of variables:
 - Boolean (bool), integer (int), natural numbers (nat), real numbers (real)
 - Enumerated types
 - Bit-vectors (fixed precision integers): $\text{int}[32]$, $\text{int}[64]$
 - Arrays and matrices
- Operators allowed in expressions, e.g. for integers, three classes:
 - Difference constraints: $x - y \leq 5$
 - Linear constraints: $2x + 3y - 5z \leq 7$
 - Full arithmetic: $r = x^2 - y^2$
- Whether quantifiers over variables are allowed

The more restricted the specification logic, easier it is for a tool to solve the resulting constraint satisfaction problem

▪ Acknowledgement for slides:

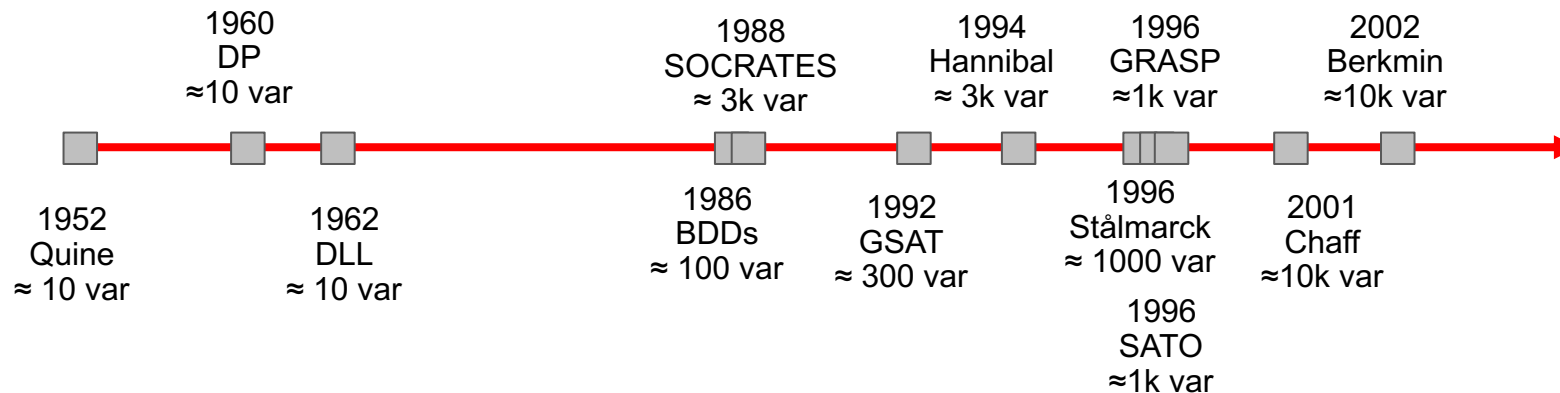
- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

Verifying Programs:

SAT Solvers

Propositional Satisfiability: Given a formula over Boolean variables, is there an assignment of 0/1's to variables which makes the formula true

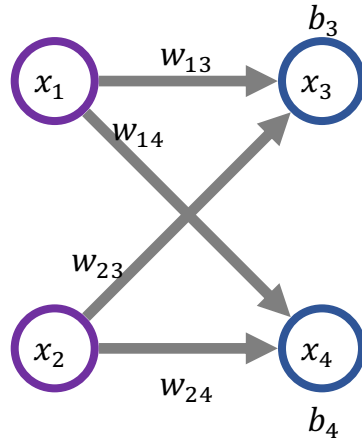
- Canonical NP-hard problem (Cook 1973)
- Enormous progress in tools that can solve instances with thousands of variables and millions of clauses
- Also at the core of SMT solvers (extensions to richer classes of constraints)



■ Acknowledgement for slides:

- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

Each layer is a function



$$(x_3, x_4) \leftarrow f_1(x_1, x_2) \text{ where}$$
$$x_3 = w_{13} \cdot x_1 + w_{23} \cdot x_2 + b_3$$
$$x_4 = w_{14} \cdot x_1 + w_{24} \cdot x_2 + b_4$$

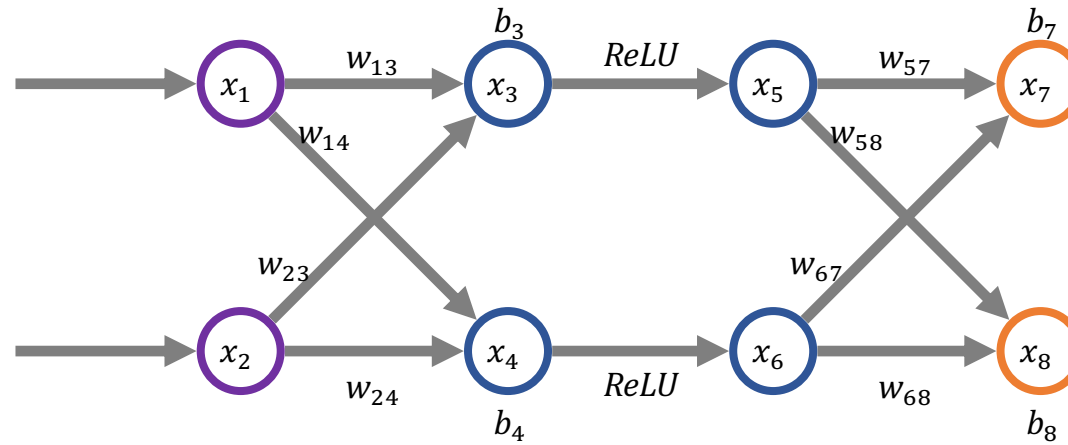


$$(x_5, x_6) \leftarrow f_2(x_3, x_4) \text{ where}$$
$$x_5 = \text{ReLU}(x_3) = \max(0, x_3)$$
$$x_6 = \text{ReLU}(x_4) = \max(0, x_4)$$

■ Acknowledgement for slides:

- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

DNN is composition of layerwise functions



$$(x_7, x_8) \leftarrow f(x_1, x_2) = f_3 \circ f_2 \circ f_1(x_1, x_2)$$

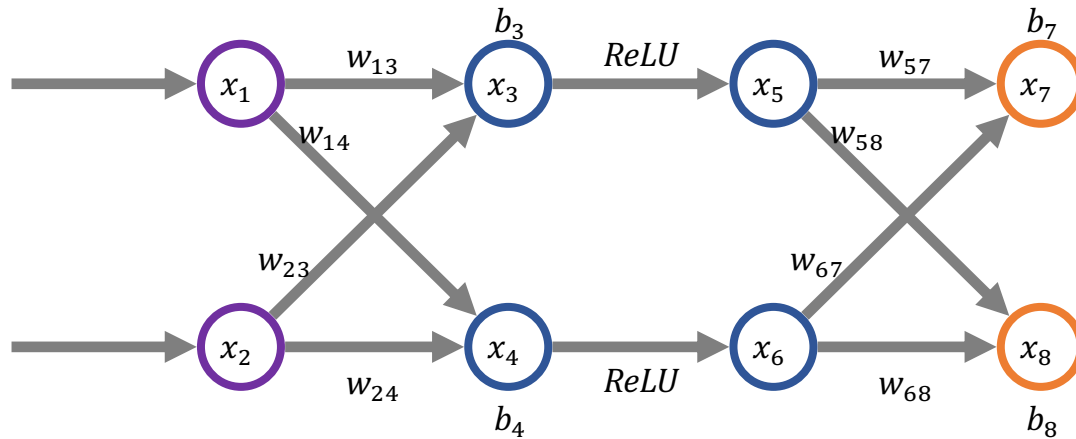
where f_3 is the function computed by the third layer

■ Acknowledgement for slides:

- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

DNNs and Programs

- DNNs can be seen as straight-line programs (programs without loops)



def $f(x_1, x_2)$:

$$x_3 = w_{13} \cdot x_1 + w_{23} \cdot x_2 + b_3$$

$$x_4 = w_{14} \cdot x_1 + w_{24} \cdot x_2 + b_4$$

$$x_5 = \max(0, x_3)$$

$$x_6 = \max(0, x_4)$$

$$x_7 = w_{57} \cdot x_5 + w_{67} \cdot x_6 + b_7$$

$$x_8 = w_{58} \cdot x_5 + w_{68} \cdot x_6 + b_8$$

return x_7, x_8

▪ Acknowledgement for slides:

- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

Specifications over DNNs

Precondition

$$\forall x_1, x_2. l_1 \leq x_1 \leq u_1, l_2 \leq x_2 \leq u_2$$

DNN f

```
def  $f(x_1, x_2)$ :  
   $x_3 = w_{13} \cdot x_1 + w_{23} \cdot x_2 + b_3$   
   $x_4 = w_{14} \cdot x_1 + w_{24} \cdot x_2 + b_4$   
   $x_5 = \max(0, x_3)$   
   $x_6 = \max(0, x_4)$   
   $x_7 = w_{57} \cdot x_5 + w_{67} \cdot x_6 + b_7$   
   $x_8 = w_{58} \cdot x_5 + w_{68} \cdot x_6 + b_8$   
  return  $x_7, x_8$ 
```

Postcondition

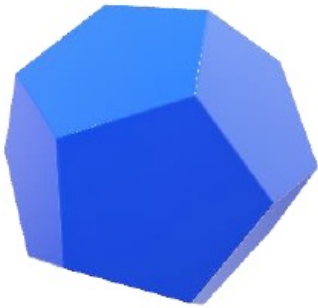
$$x_7 > x_8$$

Either prove that the network output satisfies the postcondition for all inputs in the pre-condition or find a counterexample

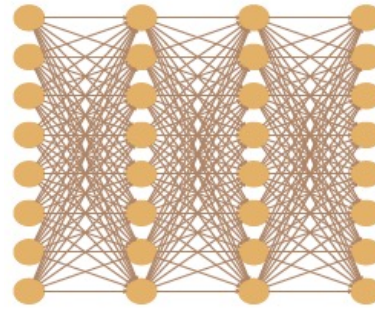
▪ Acknowledgement for slides:

- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

Neural network certification: problem statement



Precondition over
network inputs ϕ



Network f

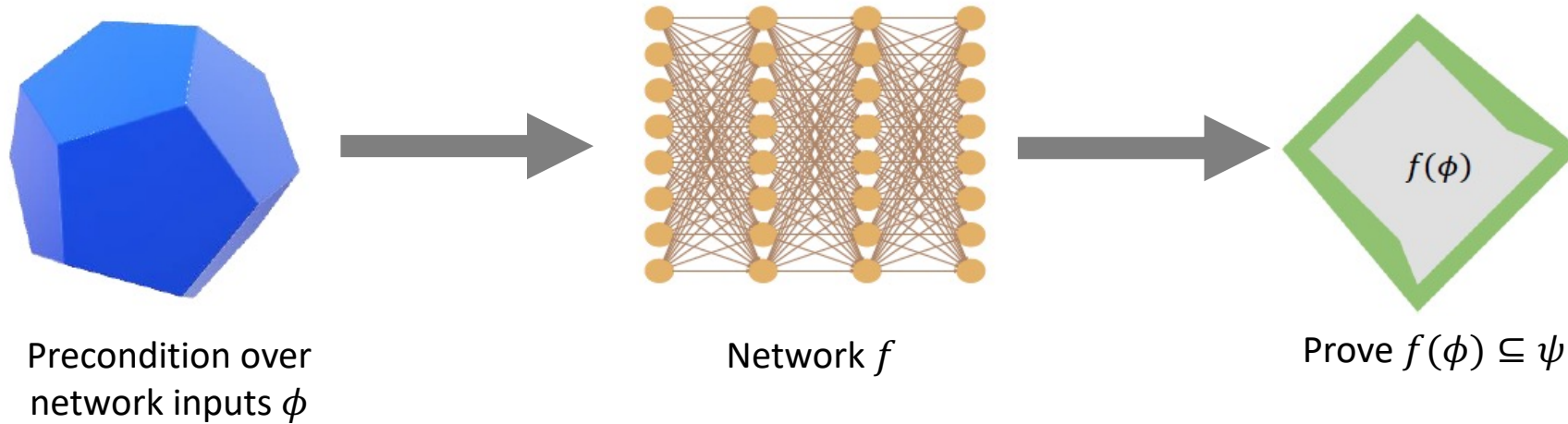


Postcondition over
network outputs ψ

▪ Acknowledgement for slides:

- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

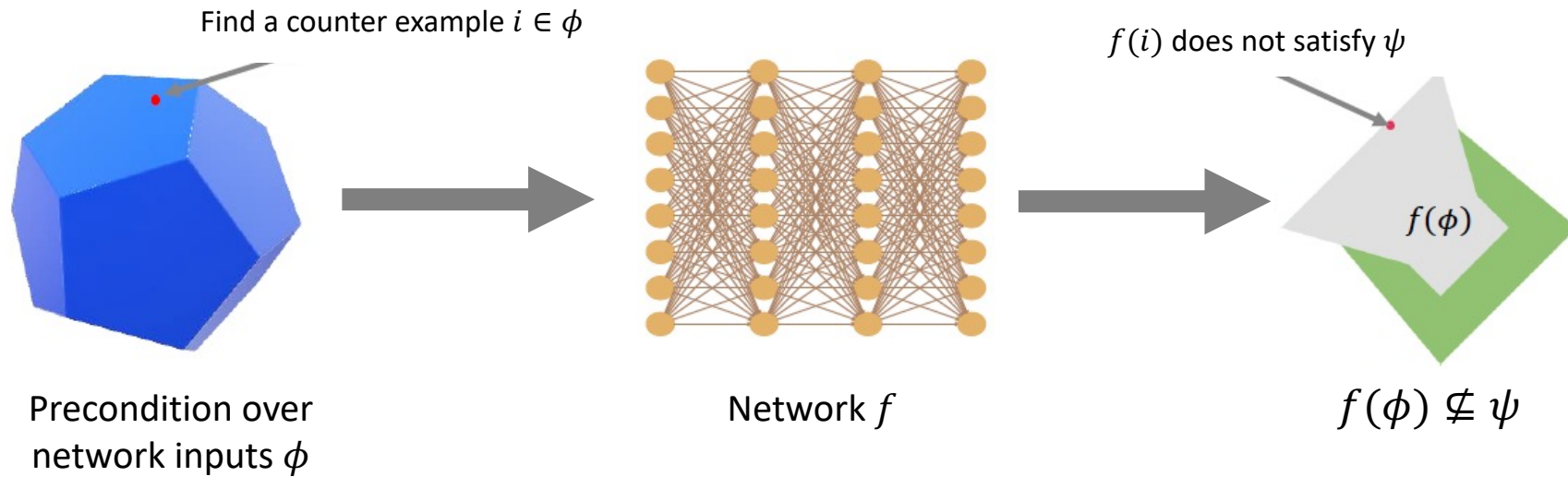
Neural network certification: problem statement



▪ Acknowledgement for slides:

- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

Neural network certification: problem statement



▪ Acknowledgement for slides:

- CIS 6730 (Computer Aided Verification) Lectures
- Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)

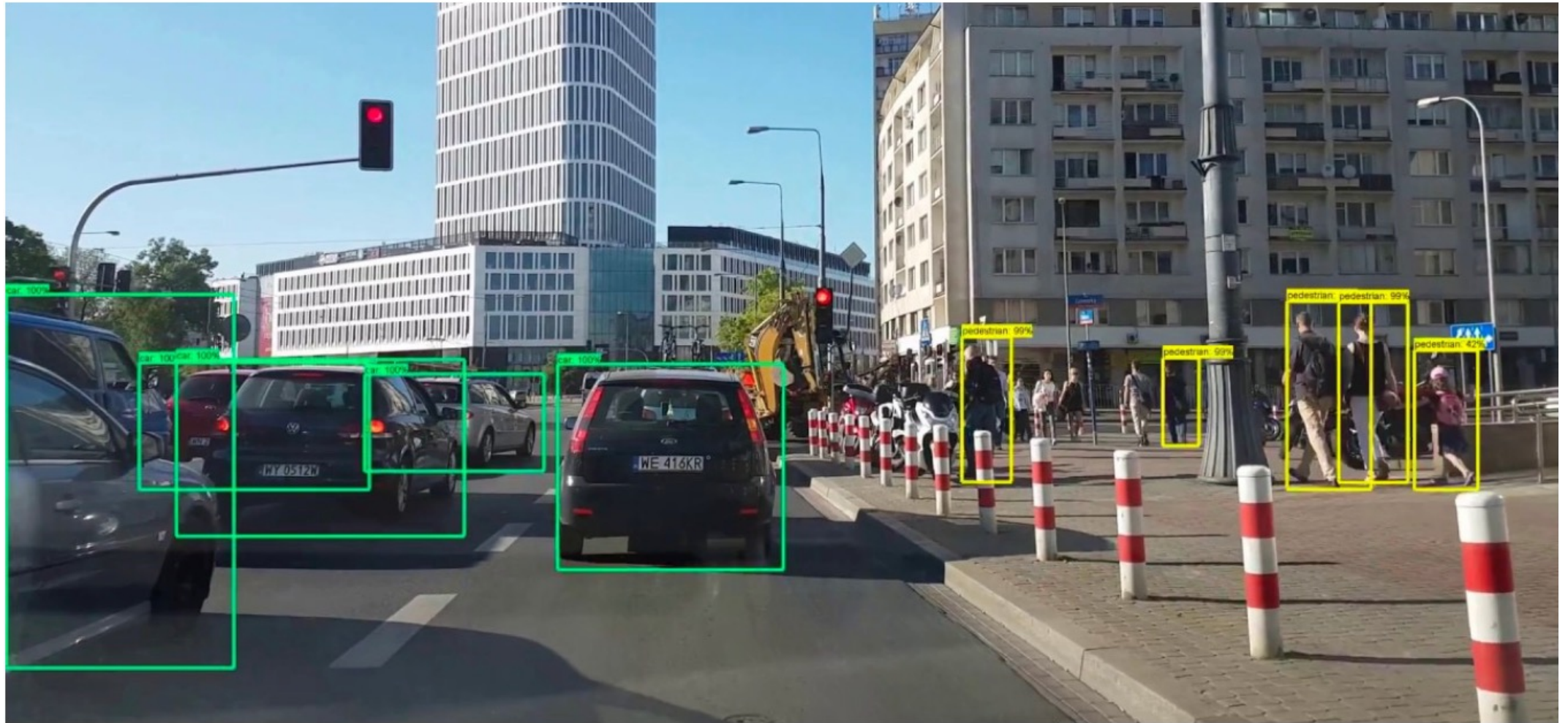
Uncertainty

+ Calibration, Conformal Prediction

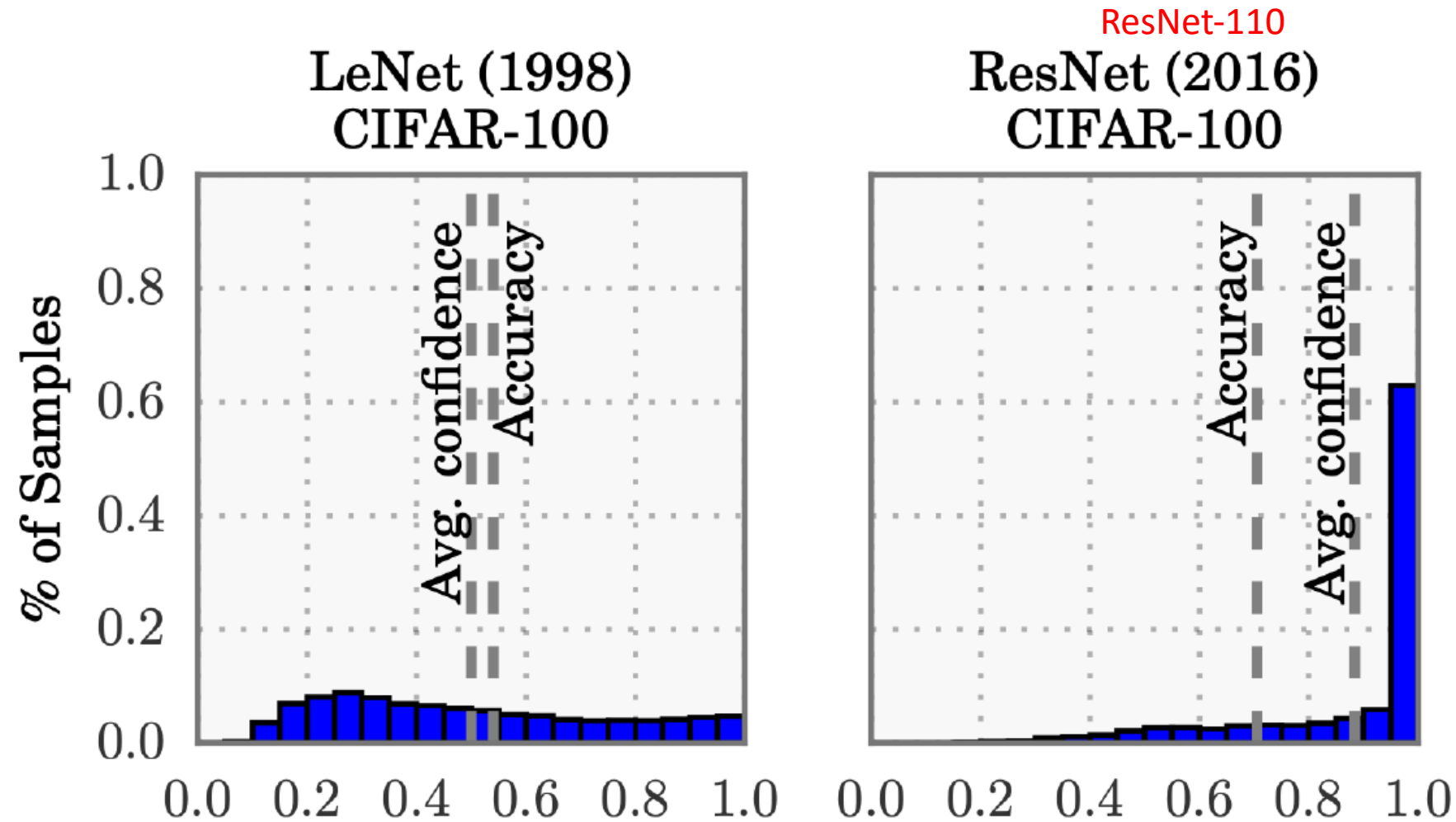
Robustness vs. Uncertainty Quantification

- Robustness aims to ensure the model performs well on shifted inputs
- Doesn't say anything about performance on original inputs!
 - A model that always predicts “dog” is robust, but not very useful
- What can we guarantee for performance on original inputs?
 - In general, we can't guarantee much (maybe the problem is just really hard!)
 - But, we can give **uncertainty quantification** (“knows what it doesn't know”)
 - Initially focus on **on-distribution** (no shifts, no adversarial attacks, etc.)

Uncertainty Quantification



Modern Neural Networks are Overconfident



The plots are from this paper: <https://arxiv.org/pdf/1706.04599>

Uncertainty Quantification

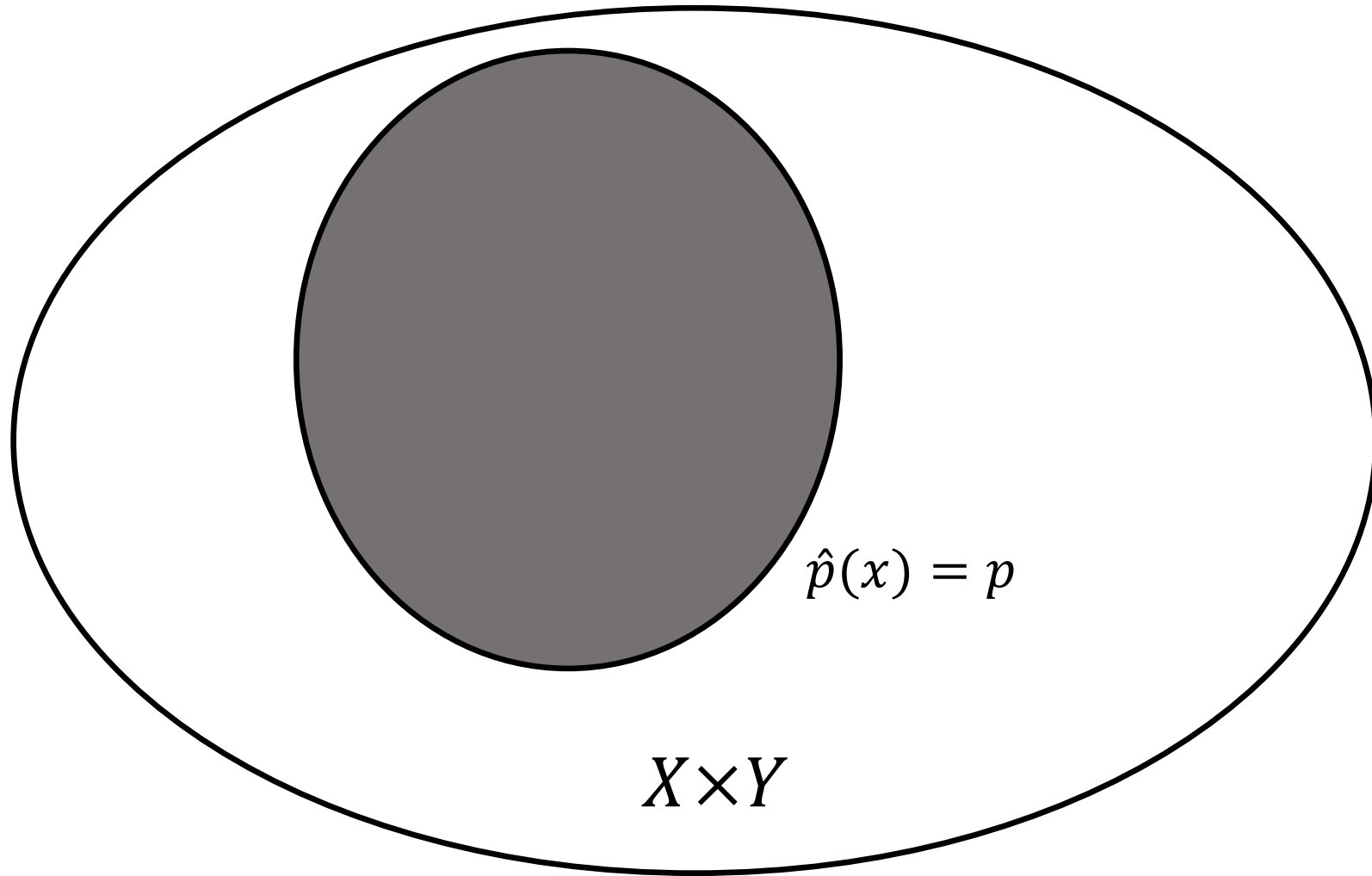
- **Calibrated Prediction (Platt 1999, Guo 2017)**
 - Predict a **probability** $\vec{p}(x)_y$ for each label y
 - What does it mean for the probabilities to be correct?
- **Prediction Sets**
 - Predict a **set** $\hat{C}(x) \subseteq Y$ of possible labels
 - Set is correct if $y^* \in \hat{C}(x)$

Calibrated Prediction

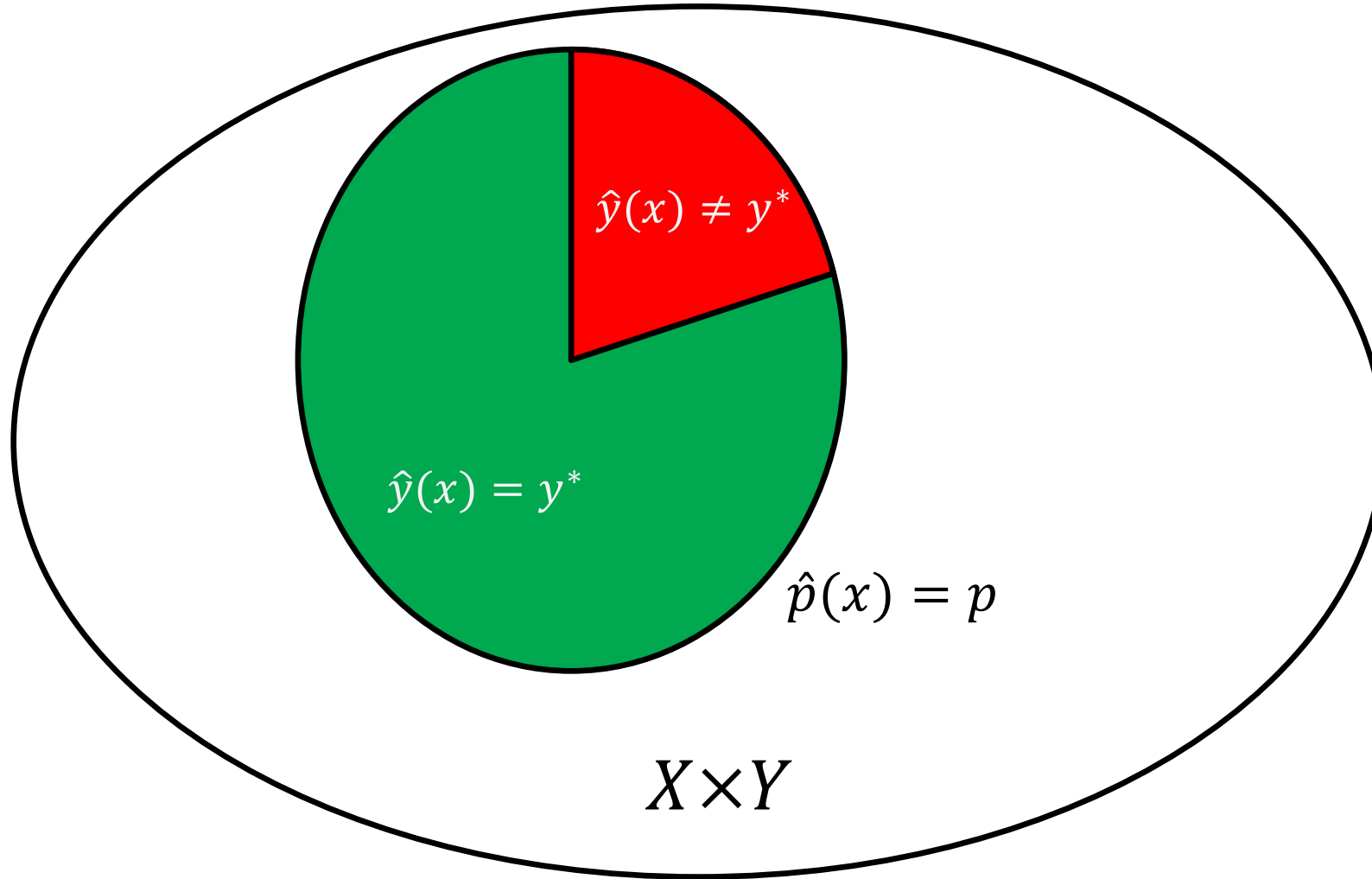
- Consider a probability predictor $\vec{p}: X \rightarrow [0,1]^{|Y|}$
 - Let $\hat{y}(x) = \arg \max_{y \in Y} \vec{p}(x)_y$ denote the corresponding labeling function
 - Let $\hat{p}(x) = \vec{p}(x, \hat{y}(x))$ be the probability of the predicted label
- We say $\hat{p}$ is **calibrated** if for all $p \in [0,1]$, we have

$$p = \Pr_{p(x, y^*)} [\hat{y}(x) = y^* \mid \hat{p}(x) = p]$$

Calibrated Prediction



Calibrated Prediction



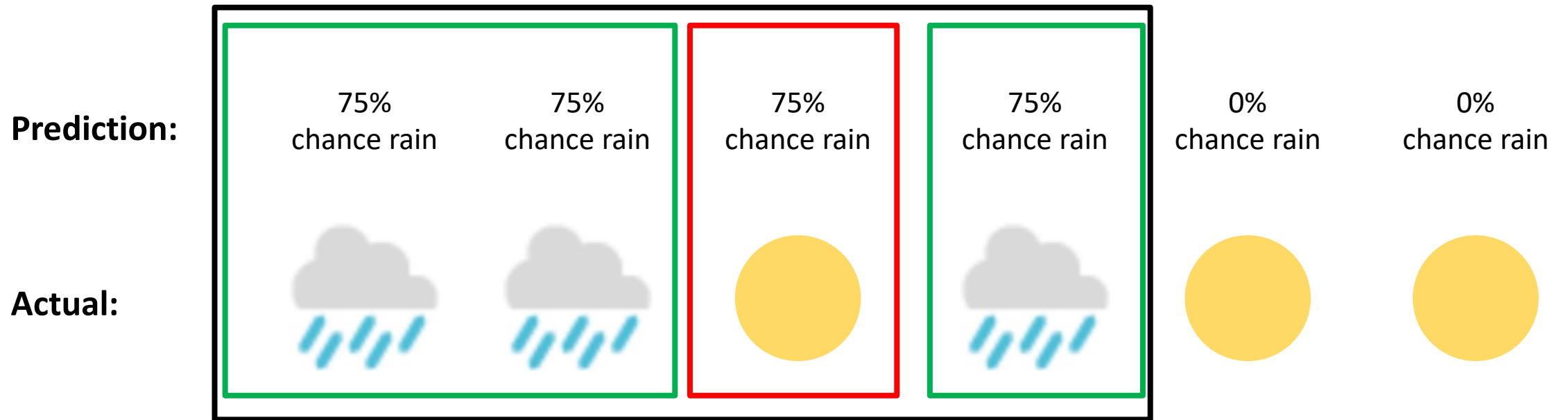
Goal: $\frac{\text{green}}{\text{green} + \text{red}} = p$

Calibrated Prediction

- What does “40% chance of rain” mean?
- Among all days with 40% chance of rain, it rains in 40% of them

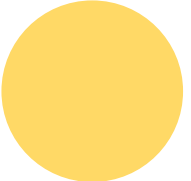
Mon 26	57°/39°		Partly Cloudy	 7%	 SW 7 mph	
Tue 27	63°/54°		PM Showers	 48%	 SSE 13 mph	
Wed 28	65°/39°		Showers	 60%	 SSW 19 mph	
Thu 29	42°/28°		AM Showers	 33%	 NW 17 mph	
Fri 01	47°/32°		Mostly Sunny	 7%	 SSW 9 mph	
Sat 02	56°/40°		Partly Cloudy	 18%	 SSE 8 mph	
Sun 03	61°/43°		AM Showers	 48%	 ENE 7 mph	
Mon 04	61°/48°		Showers	 43%	 E 9 mph	
Tue 05	63°/51°		Showers	 49%	 SE 11 mph	
Wed 06	63°/48°		Showers	 58%	 SSE 13 mph	
Thu 07	59°/47°		Showers	 40%	 SW 11 mph	
Fri 08	60°/44°		Showers	 42%	 NNW 10 mph	

Calibrated Prediction



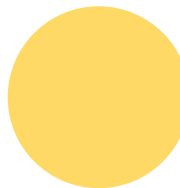
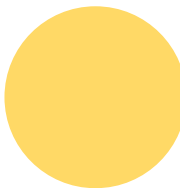
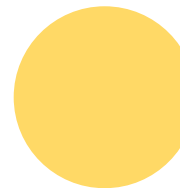
$$\Pr_{p(y^*)} [y^* = \text{rain} \mid \hat{p} = 0.75] = 0.75$$

Calibrated Prediction

Prediction:	75% chance rain	75% chance rain	75% chance rain	75% chance rain	0% chance rain	0% chance rain
Actual:						

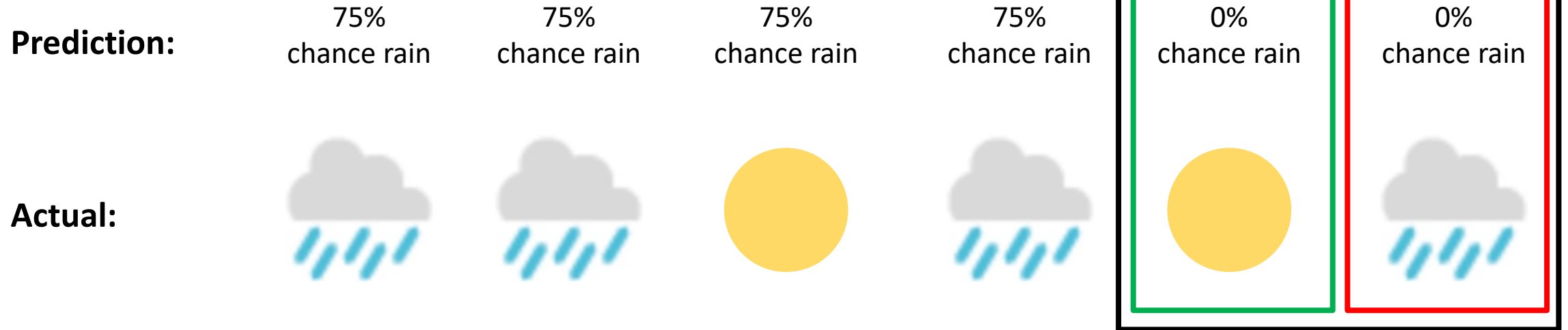
$$\Pr_{p(y^*)} [y^* = \text{rain} \mid \hat{p} = 0] = 0$$

Calibrated Prediction

Prediction:	75% chance rain	75% chance rain	75% chance rain	75% chance rain	25% chance rain	25% chance rain
Actual:						

$$\Pr_{p(y^*)} [y^* = \text{rain} \mid \hat{p} = 0.25] = 0$$

Calibrated Prediction



$$\Pr_{p(y^*)} [y^* = \text{rain} \mid \hat{p} = 0] = 0.5$$

Calibration and Binning

- **Recall:** Calibration is defined as

$$\Pr_{p(x,y^*)} [\hat{y}(x) = y^* \mid \hat{p}(x) = p] = p \quad (\forall p \in [0,1])$$

- Conditions on potentially zero probability event (the denominator becomes zero)
- In practice, two inputs may never have exactly the same probability $\hat{p}(x)$

- **Idea:** Bin probabilities into bins $P_i = [p_{\text{low},i}, p_{\text{high},i})$ instead:

$$\Pr_{p(x,y^*)} [\hat{y}(x) = y^* \mid \hat{p}(x) \in P_i] = \text{Conf}(P_i) \quad (\forall i \in \{1, \dots, k\})$$

Calibration and Binning

- **Idea:** Bin probabilities into bins $P_i = [p_{\text{low},i}, p_{\text{high},i})$ instead:

$$\Pr_{p(x,y^*)} [\hat{y}(x) = y^* \mid \hat{p}(x) \in P_i] = \text{Conf}(P_i) \quad (\forall i \in \{1, \dots, k\})$$

- $\text{Conf}(P) = \mathbb{E}_{p(x,y^*)}[\hat{p}(x) \mid \hat{p}(x) \in P]$ is the average probability of bin P

Measuring Calibration

- **Recall:** Binned calibration is defined as

$$\Pr_{p(x,y^*)} [\hat{y}(x) = y^* \mid \hat{p}(x) \in P_i] = \text{Conf}(P_i) \quad (\forall i \in \{1, \dots, k\})$$

- Chances of equality holding exactly is effectively zero
 - How to **measure** calibration error (instead of requiring exact calibration)?
- **Idea:** Use mean absolute error (called **expected calibration error**):

$$\text{ECE}(\hat{p}) = \mathbb{E}_{p(P)} \left[\left| \Pr_{p(x,y^*)} [\hat{y}(x) = y^* \mid \hat{p}(x) \in P] - \text{Conf}(P) \right| \right]$$

Measuring Calibration

- **Idea:** Use mean absolute error (called **expected calibration error**):

$$\text{ECE}(\hat{p}) = \mathbb{E}_{p(P)} \left[\left| \Pr_{p(x,y^*)} [\hat{y}(x) = y^* \mid \hat{p}(x) \in P] - \text{Conf}(P) \right| \right]$$

- $P = [p_{\min}, p_{\max}) \subseteq [0,1]$ is a bin in probability space
- $p(P) = \Pr_{p(x,y^*)} [\hat{p}(x) \in P]$ is the probability of bin P
- $\text{Conf}(P) = \mathbb{E}_{p(x,y^*)} [\hat{p}(x) \mid \hat{p}(x) \in P]$ is the average probability of bin P

Measuring Calibration

- **Idea:** Use mean absolute error (called **expected calibration error**):

$$\text{ECE}(\hat{p}) = \mathbb{E}_{p(P)} \left[\left| \Pr_{p(x, y^*)} [\hat{y}(x) = y^* \mid \hat{p}(x) \in P] - \text{Conf}(P) \right| \right]$$

- Equivalently, we have

$$\text{ECE}(\hat{p}) = \mathbb{E}_{p(P)} [| \text{Acc}(P) - \text{Conf}(P) |]$$

Measuring Calibration

- **Idea:** Use mean absolute error (called **expected calibration error**):

$$\text{ECE}(\hat{p}) = \mathbb{E}_{p(P)} \left[\left| \Pr_{p(x, y^*)} [\hat{y}(x) = y^* \mid \hat{p}(x) \in P] - \text{Conf}(P) \right| \right]$$

- On a held-out test set Z , we have the approximation

$$\text{ECE}(\hat{p}; Z) = \sum_{i=1}^k \frac{|B_i|}{|Z|} \cdot \left| \frac{1}{|B_i|} \sum_{(x, y^*) \in B_i} 1(\hat{y}(x) = y^*) - \frac{1}{|B_i|} \sum_{(x, y^*) \in B_i} \hat{p}(x) \right|$$

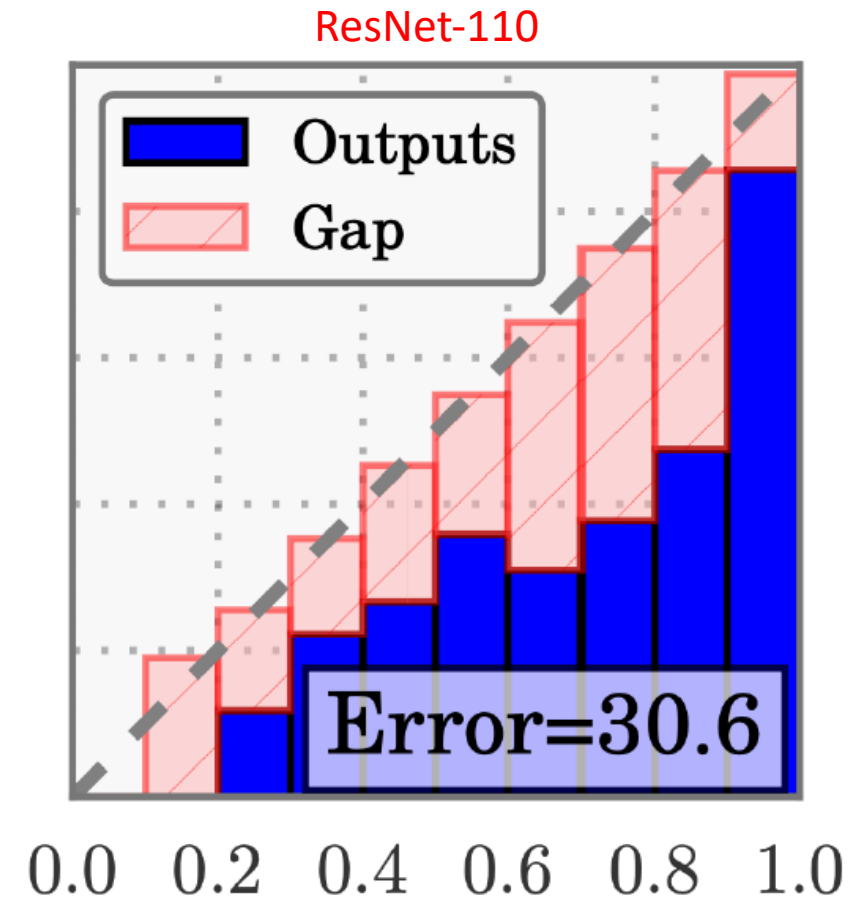
- Here, $B_i = \{(x, y^*) \in Z \mid \hat{p}(x) \in P_i\}$ is the bin in feature space

Reliability Diagrams

- For each bin P_i , plot accuracy

$$\Pr_{p(x, y^*)} [\hat{y}(x) = y^* \mid \hat{p}(x) \in P]$$

$$\approx \frac{1}{|B_i|} \sum_{(x, y^*) \in B_i} 1(\hat{y}(x) = y^*)$$



The plots are from this paper: <https://arxiv.org/pdf/1706.04599>

Miscalibration of Neural Networks

- Typical approach in deep learning
 - However, most state-of-the-art models have high calibration error
- **Potential explanation**
 - Models need to be **overparameterized** to aid optimization
 - Overparameterization leads to overfitting probabilities (even if accuracy is good!)

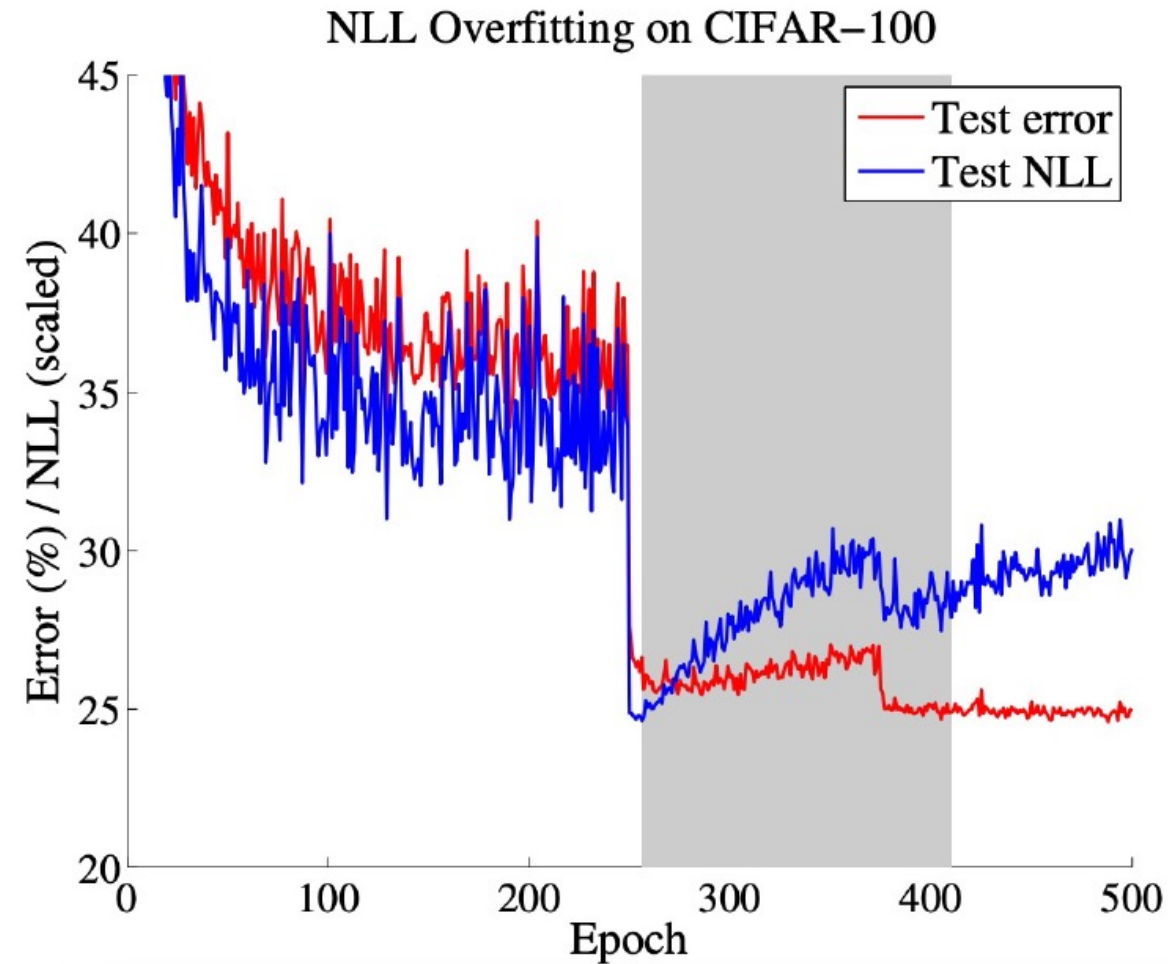
Recall: Negative Log Likelihood

- Negative log likelihood is

$$\text{NLL}(\vec{p}) = - \sum_{(x, y^*) \in Z} \log \vec{p}(x)_{y^*}$$

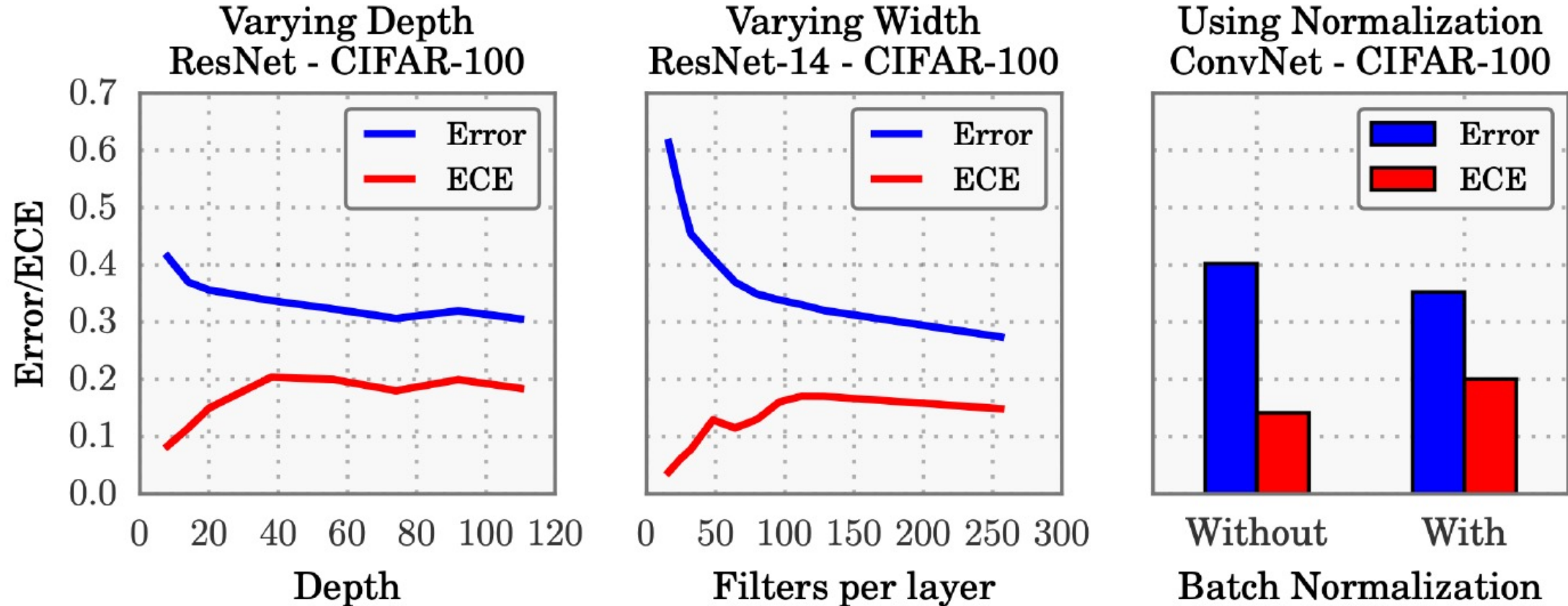
- NLL is zero if and only if $\vec{p} = p^*$, where p^* are the “true” probabilities
 - Thus, good NLL roughly corresponds to good calibration

Miscalibration and Overfitting



The plots are from this paper: <https://arxiv.org/pdf/1706.04599>

Miscalibration and Overfitting



Improving Calibration

- How can we fix the problem?
 - Better training algorithms
 - Regularization
 - Post-hoc modifications (called **recalibration**)
 - Histogram binning
 - Temperature scaling
- (E.g.: label smoothing, mixup, weight decay, focal loss, explicit regularization loss, ..)

Recalibration

- **Goal:** Rescale outputs using a function to minimize ECE
- **Inputs:** Probability predictor $\hat{p}$, held-out calibration dataset Z
- **Output:** For some function $\phi: [0,1] \rightarrow [0,1]$, define new probabilities

$$\hat{q}(x) = \phi(\hat{p}(x))$$

- Works well since ϕ is a 1D transformation, so it is “simple”

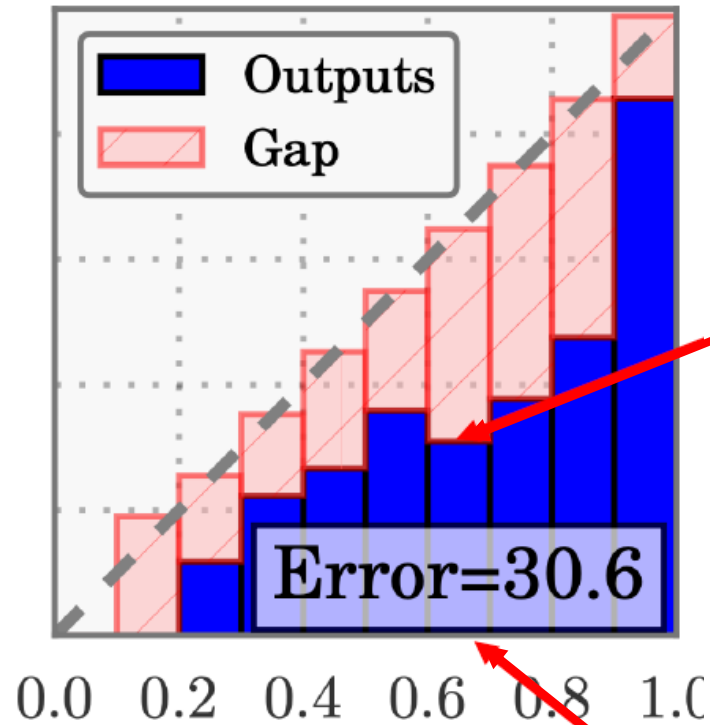
Histogram Binning

- **Idea:** Use a nonparametric ϕ obtained via binning
- **Algorithm:**
 - **Input:** Probability predictor $\hat{p}$, calibration dataset Z , example x
 - Let P be the probability bin containing x (i.e., $\hat{p}(x) \in P$)
 - Let $B = \{(x', y^*) \in Z \mid \hat{p}(x') \in P\}$ be the corresponding bin over Z
 - **Output:** Define the following (values can be precomputed for each bin):

Set transformed
probabilities as:

$$\phi(x) = \frac{1}{|B|} \sum_{(x', y^*) \in B} 1(\hat{y}(x) = y^*)$$

Histogram Binning



Given x

$$\text{Acc}(P) = \frac{1}{|B|} \sum_{(x', y^*) \in B} 1(\hat{y}(x') = y^*) = 0.35$$

$$\phi(x) = \text{Acc}(P) = 0.35$$

$$\hat{p}(x) = 0.62 \in [0.6, 0.7) := P$$

Histogram Binning

- **Why does this work?**

- After transformation, all points with $\hat{p}(x) = P$ now have $\hat{q}(x) = \text{Acc}(P)$
 - Their new bin is Q , where $\hat{q}(x) \in Q$
 - We also have $\text{Acc}(Q) = \text{Acc}(P)$ (ignoring other points in Q for simplicity)
 - Putting the above together, we have $\text{Acc}(Q) = \hat{q}(x)$
 - Thus, $\text{ECE}(\hat{q}) = 0$
-
- The derivation uses the true accuracy, but our algorithm uses the empirical accuracy, so there can be some error during evaluation

Temperature Scaling

- Only fits a **single parameter** τ , even for the multi-class setting
 - Called the **temperature**
- Works best when **ordering of probabilities is good**

Temperature Scaling

- Consider the model family

$$\vec{q}_\tau(x) = \text{softmax}\left(\frac{\text{logits}(x)}{\tau}\right)$$

- Taking $\tau = 1$ recovers the original model: $\vec{q}_1(x) = \vec{p}(x)$
 - Taking $\tau > 1$ decreases confidence ($\tau \rightarrow \infty$ yields probabilities equal to $\frac{1}{k}$)
 - Taking $\tau < 1$ increases confidence ($\tau \rightarrow 0$ yields probabilities equal in $\{0,1\}$)
- Choose τ to minimize NLL of $\vec{q}_\tau$ on calibration dataset Z
 - Can use grid search to do so (i.e., search over fixed set of choices for τ)

Recall: Softmax with a Scaling Factor

- Softmax is a smooth version of arg max:

$$\arg \max (s_1, s_2, \dots, s_n) = (y_1, y_2, \dots, y_n) = (0, 0, \dots, 0, 1, 0, \dots, 0)$$

- The base in Softmax can be changed to have more “peaky” (or distributed) values for the largest input ($e^\beta = b$):

$$sm_\beta(s_i) = \frac{e^{\beta s_i}}{\sum_j e^{\beta s_j}}$$

- When $\beta \rightarrow \infty$, Softmax converges to arg-max.

e.g.

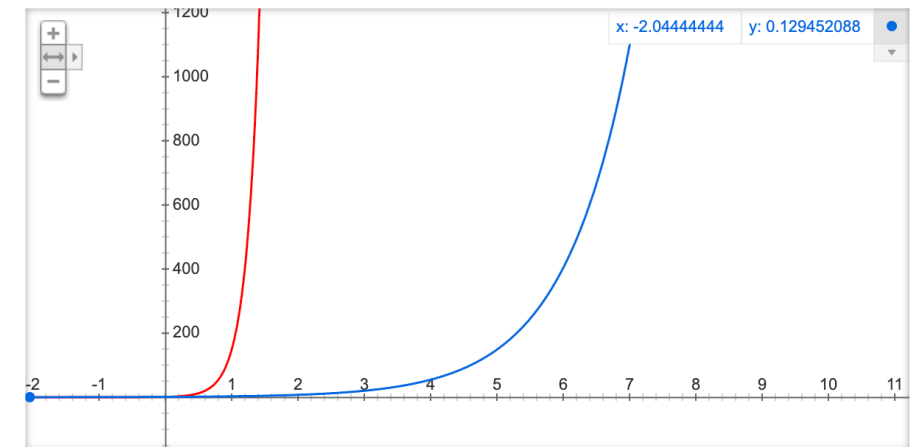
$$sm_{\beta=1}([1, 1.1]) = [0.475 \ 0.524]$$

$$sm_{\beta=2}([1, 1.1]) = [0.451 \ 0.550]$$

$$sm_{\beta=5}([1, 1.1]) = [0.378 \ 0.622]$$

$$sm_{\beta=100}([1, 1.1]) = [0.001 \ 0.999]$$

Graph for $\exp(x)$, $\exp(5 \cdot x)$



More info

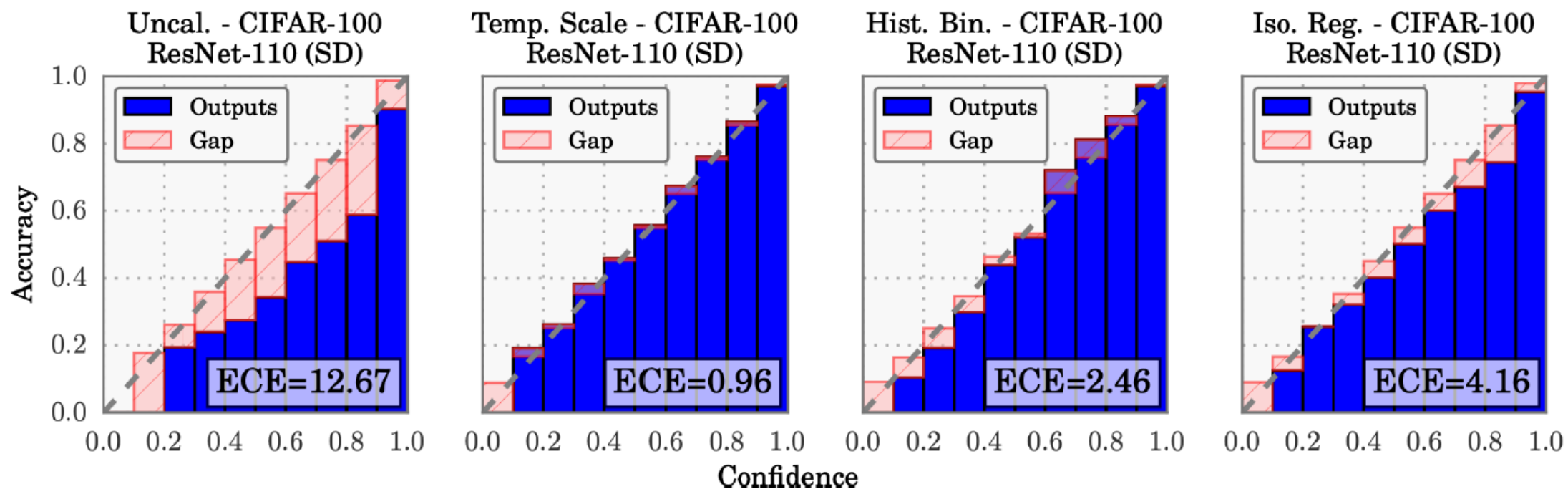
Recall: Softmax with Temperature

- Softmax with temperature is Softmax with $\beta = 1/T$:

$$sm_{1/T}(s_i) = \frac{e^{s_i/T}}{\sum_j e^{s_j/T}}$$

- Interpretation:
 - Increase $T \Rightarrow$ decrease $\beta \Rightarrow$ decrease the peak around the largest value.
- Low T yields more confident (may be over confident) probability distribution.
- Especially in training sequence models where we perform sampling from the output distribution, in order to allow diversity, we can increase T .

Reliability Diagrams for Re-Calibration



Conformal Prediction (Inference)

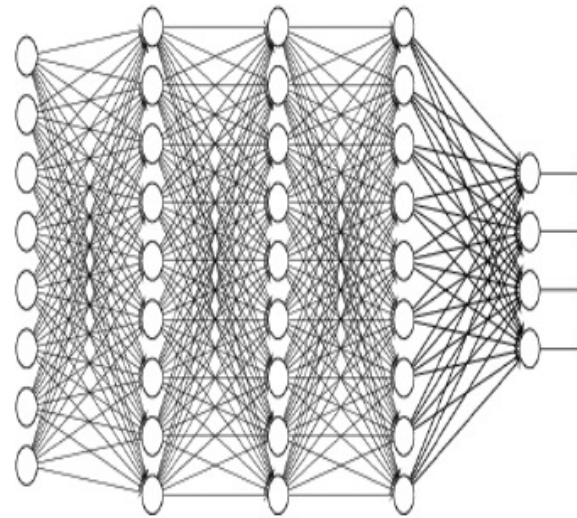
Some resources:

- <https://arxiv.org/pdf/2107.07511>
- <https://mindfulmodeler.substack.com/p/e-mail-course-on-conformal-prediction>

Conformal Prediction



Image x



DNN f

Incorrect!

(Ground truth label: y^* = "plunger")



"toilet seat"

Prediction

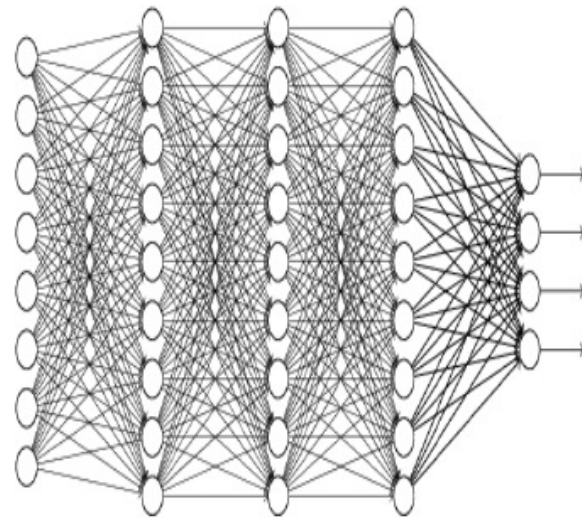
$$\hat{y} = \max_y f(y \mid x)$$

Idea: Modify DNN f to predict **sets of labels**

Conformal Prediction



Image x



Prediction Set $\tilde{f}$



{
barber chair,
hand blower,
medicine chest,
paper towel,
plunger,
shower curtain,
soap dispenser,
toilet seat,
tub, washbasin,
washer, toilet tissue
}

Output $Y = \tilde{f}(x)$

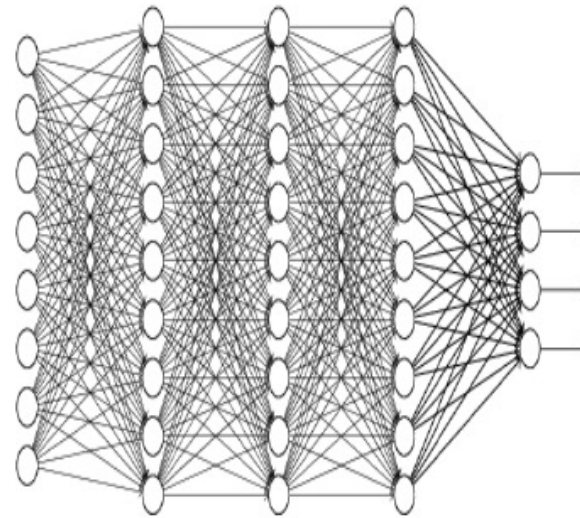
Idea: Modify DNN f to predict **sets of labels**

Conformal Prediction

Now, we have $y^* \in \tilde{f}(x)$ (**coverage**)



Image x



Prediction Set $\tilde{f}$



barber chair,
hand blower,
medicine chest,
paper towel,
plunger,
shower curtain,
soap dispenser,
toilet seat,
tub, washbasin,
washer, toilet tissue

Output $Y = \tilde{f}(x)$

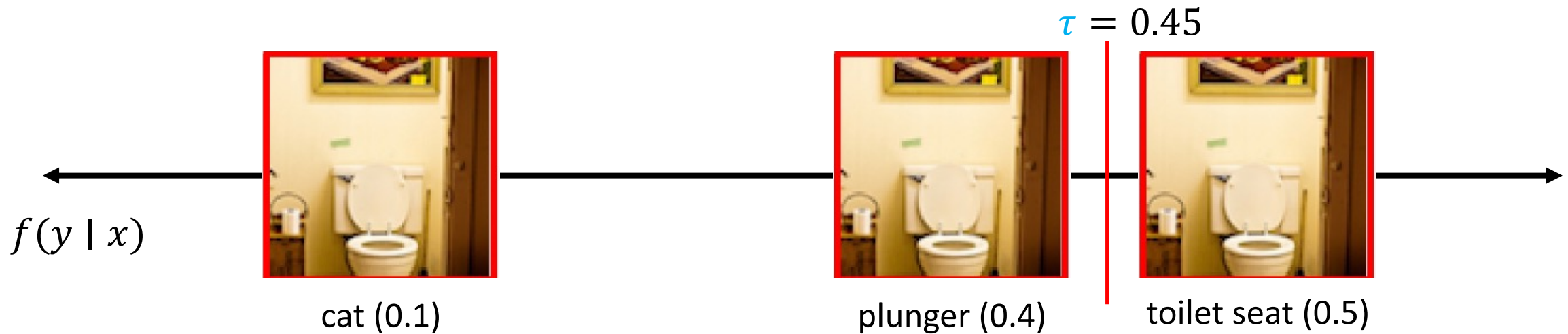
Idea: Modify DNN f to predict **sets of labels**

Conformal Prediction Problem

- **Parametric model family of prediction sets**

- We construct prediction sets based on an **existing** DNN $f(y | x)$
- Consider prediction sets that are **level sets** of f :

$$\tilde{f}_{\tau}(x) = \{y \mid f(y | x) \geq \tau\}$$



$$\tilde{f}_{\tau}(x) = \{\text{toilet seat}\}$$

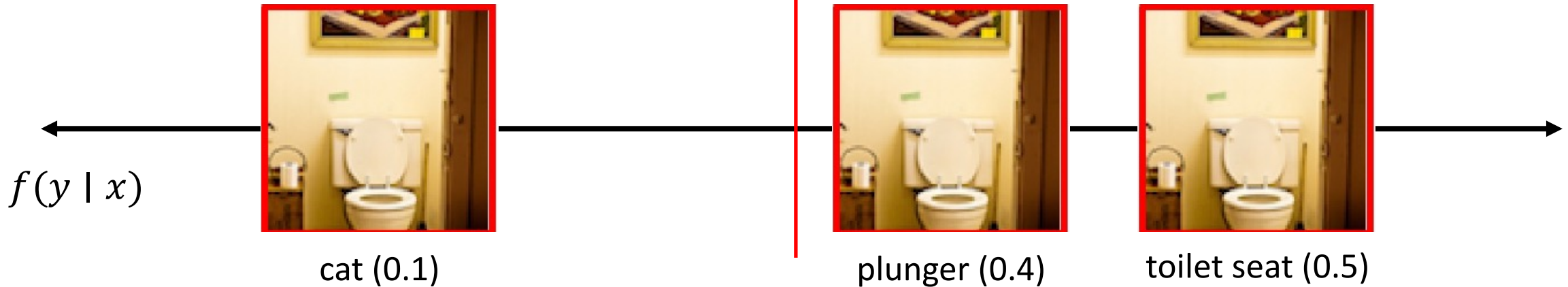
Conformal Prediction Problem

- **Parametric model family of prediction sets**

- We construct prediction sets based on an **existing** DNN $f(y | x)$
- Consider prediction sets that are **level sets** of f :

$$\tilde{f}_{\tau}(x) = \{y \mid f(y | x) \geq \tau\}$$

$$\tau = 0.35$$



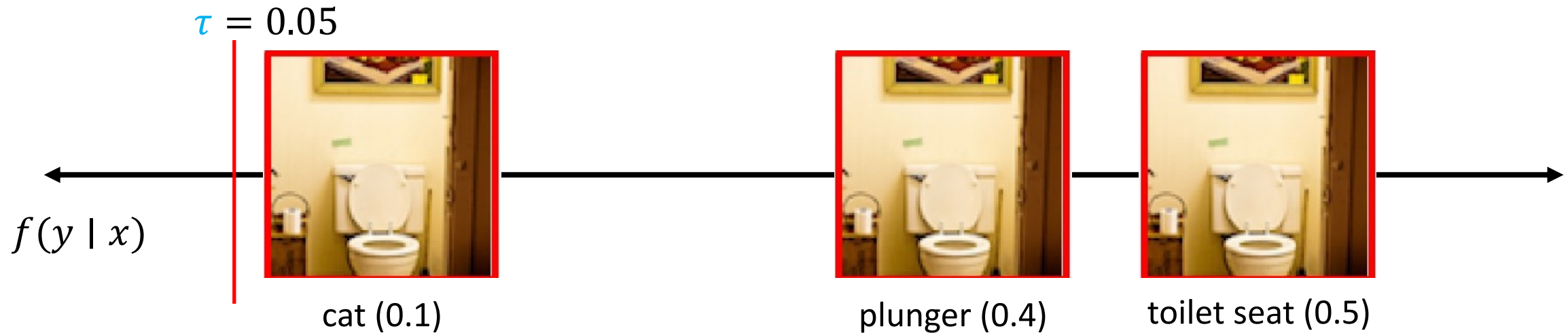
$$\tilde{f}_{\tau}(x) = \{\text{toilet seat, plunger}\}$$

Conformal Prediction Problem

- **Parametric model family of prediction sets**

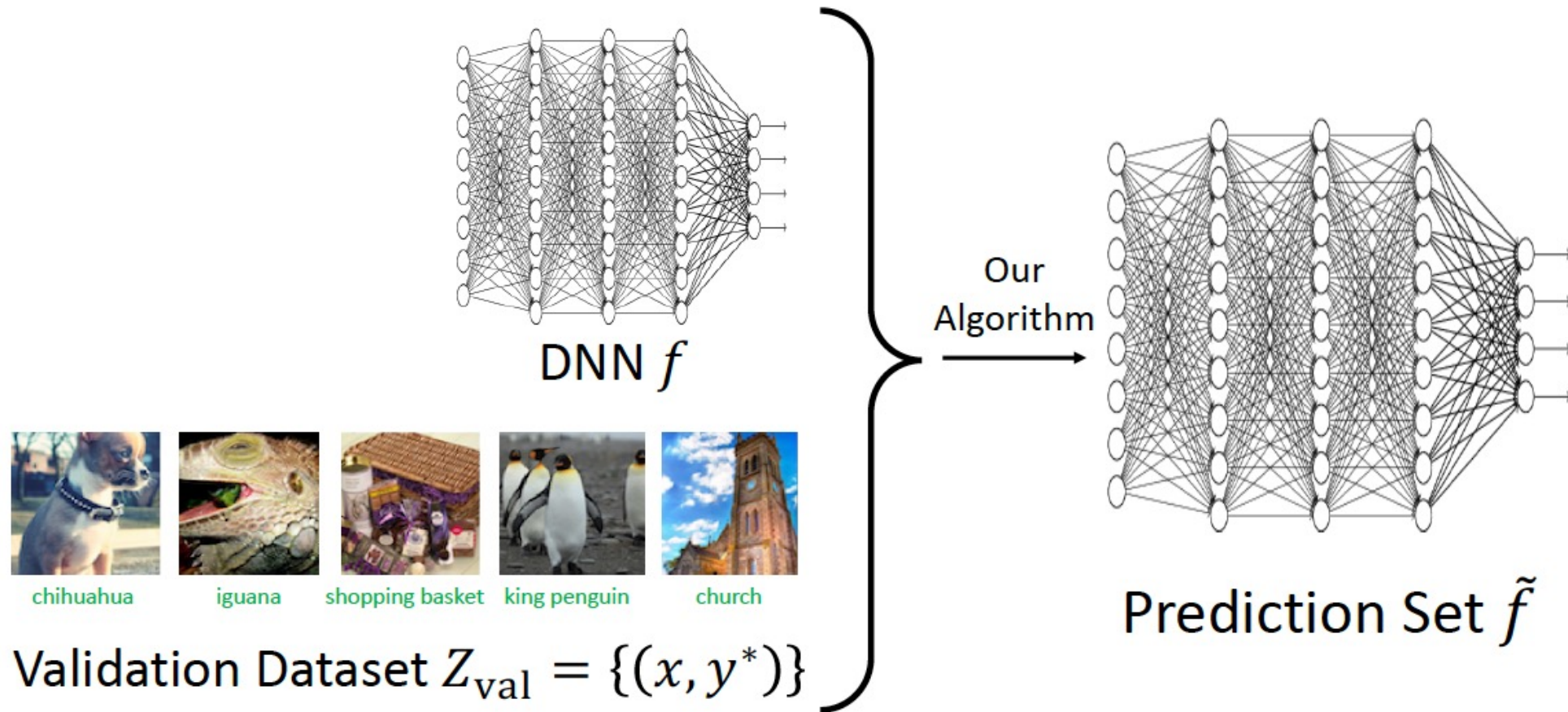
- We construct prediction sets based on an **existing** DNN $f(y | x)$
- Consider prediction sets that are **level sets** of f :

$$\tilde{f}_{\tau}(x) = \{y \mid f(y | x) \geq \tau\}$$



$$\tilde{f}_{\tau}(x) = \{\text{toilet seat, plunger, cat}\}$$

Conformal Prediction Problem



and α : User-specified error rate threshold

Conformal Prediction: Goal

- Given a predictor f , modify it to obtain a set predictor $\tilde{f}$ such that it predicts a set.
- In such a way that the correct label is within the set with $1 - \alpha$ reliability/confidence:

$$P\left(Y_{test} \in \tilde{f}(X_{test})\right) \geq 1 - \alpha$$

Why Conformal Prediction?

Conformal Prediction vs Uncertainty Estimation

- Uncertainty estimation requires
 - Additional effort
 - Models with uncertainty estimation capabilities
- Conformal Prediction is
 - Fast
 - Model-agnostic
 - Few lines of code
 - Statistical guarantees
 - No retraining required

<https://mindfulmodeler.substack.com/p/quantify-the-uncertainty-of-predictive>

Why Conformal Prediction?

Conformal Prediction vs Calibration

Shortcomings of Calibration

- Unintuitive/hard to reason about probabilities
 - Both for humans and for algorithms
- Structured prediction (e.g., sentences, object detection, etc.)
 - Probabilities of complex outputs quickly become small
 - Probabilities of different portions of the output can be highly correlated
- **Conformal prediction**
 - Represents of uncertainty using **prediction sets**, which can be more intuitive
 - Also easier to reason about algorithmically

Why Conformal Prediction?

- **“distribution-free**: the only assumption is that data points are exchangeable.
- **model-agnostic**: conformal prediction can be applied to any predictive model.
- **coverage guarantee**: the resulting prediction sets come with guarantees of covering the true outcome with a certain probability.”

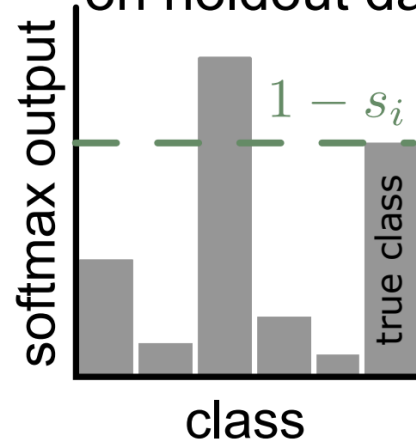
<https://mindfulmodeler.substack.com/p/week-1-getting-started-with-conformal>

Conformal Prediction Recipe

Recall our goal:

$$P(Y_{test} \in \tilde{f}(X_{test})) \geq 1 - \alpha$$

(1) compute scores
on holdout data

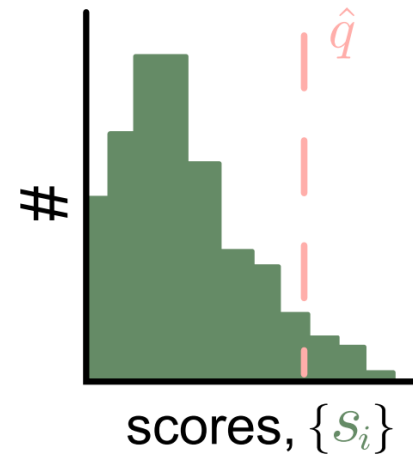


(1) Score the held-out data

$$s_i = 1 - f(y_i^* | x_i), \quad i = 1, \dots, n$$

low softmax on the true class $\Rightarrow$ large score

(2) get quantile

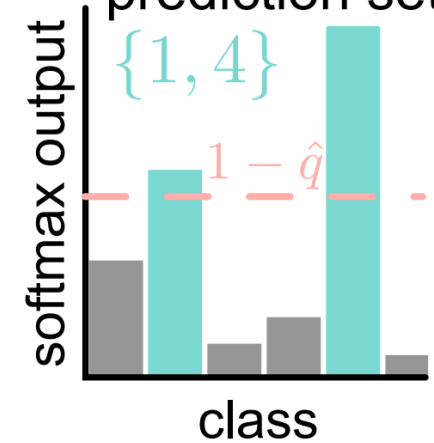


(2) Threshold: a corrected quantile

$$\hat{q} = s_{(k)}, \quad k = \lceil (n + 1)(1 - \alpha) \rceil$$

k -th smallest score $\approx$ the $(1 - \alpha)$ quantile

(3) construct
prediction set



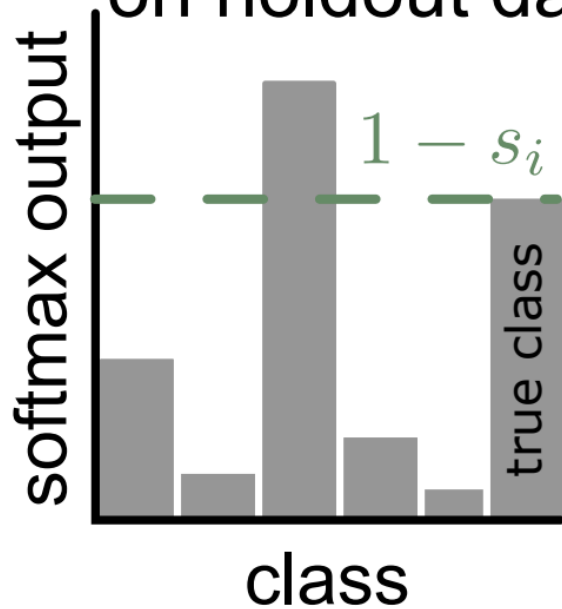
(3) Predict a set for a new input

$$\tilde{f}(x_{test}) = \{y: f(y | x_{test}) \geq 1 - \hat{q}\}$$

a level set of f with threshold $\tau = 1 - \hat{q}$

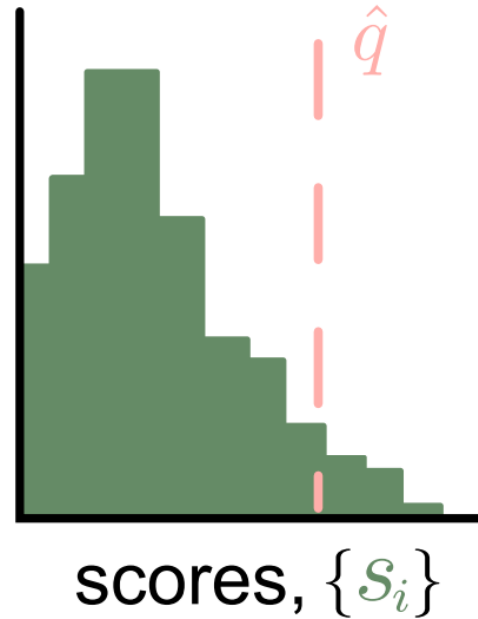
Interpretation: with $\alpha = 0.1$, at least 90% of held-out true-class softmax values lie above $1 - \hat{q} \Rightarrow$ keeping every class above $1 - \hat{q}$ captures the true class $\geq 90\%$ of the time.

(1) compute scores on holdout data

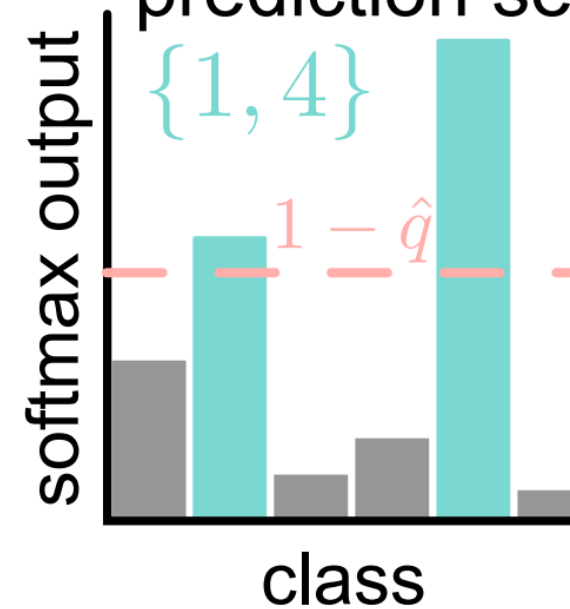


#

(2) get quantile



(3) construct prediction set



```
# 1: get conformal scores. n = calib_Y.shape[0]
cal_smx = model(calib_X).softmax(dim=1).numpy()
cal_scores = 1-cal_smx[np.arange(n), cal_labels]
# 2: get adjusted quantile
q_level = np.ceil((n+1)*(1-alpha))/n
qhat = np.quantile(cal_scores, q_level, method='higher')
val_smx = model(val_X).softmax(dim=1).numpy()
prediction_sets = val_smx >= (1-qhat) # 3: form prediction sets
```

Recall our goal:

$$P(Y_{test} \in \tilde{f}(X_{test})) \geq 1 - \alpha$$

Material:

<https://arxiv.org/pdf/2107.07511>

Code & formulation

```
# 1: get conformal scores. n = calib_Y.shape[0]
cal_smx = model(calib_X).softmax(dim=1).numpy()
cal_scores = 1-cal_smx[np.arange(n),cal_labels]
# 2: get adjusted quantile
q_level = np.ceil((n+1)*(1-alpha))/n
qhat = np.quantile(cal_scores, q_level, method='higher')
val_smx = model(val_X).softmax(dim=1).numpy()
prediction_sets = val_smx >= (1-qhat) # 3: form prediction sets
```

- **(n+1)** instead of n : the test point is the $(n + 1)$ -th sample
- **method='higher'**: never interpolate below the target quantile $\Rightarrow$ keeps the guarantee
- **No retraining**: one forward pass over the calibration set

Code	Math notation
cal_smx[i, y]	$f(y x_i)$: softmax on the calibration set
cal_scores	$s_i = 1 - f(y_i^* x_i)$, with $y_i^* = \text{cal_labels}[i]$
q_level	k/n , with $k = \lceil (n + 1)(1 - \alpha) \rceil$
qhat	$\hat{q} = s_{(k)}$, the k -th smallest score
prediction_sets	$\tilde{f}(x) = \{y: f(y x) \geq 1 - \hat{q}\}$ as a 0/1 mask

Worked example

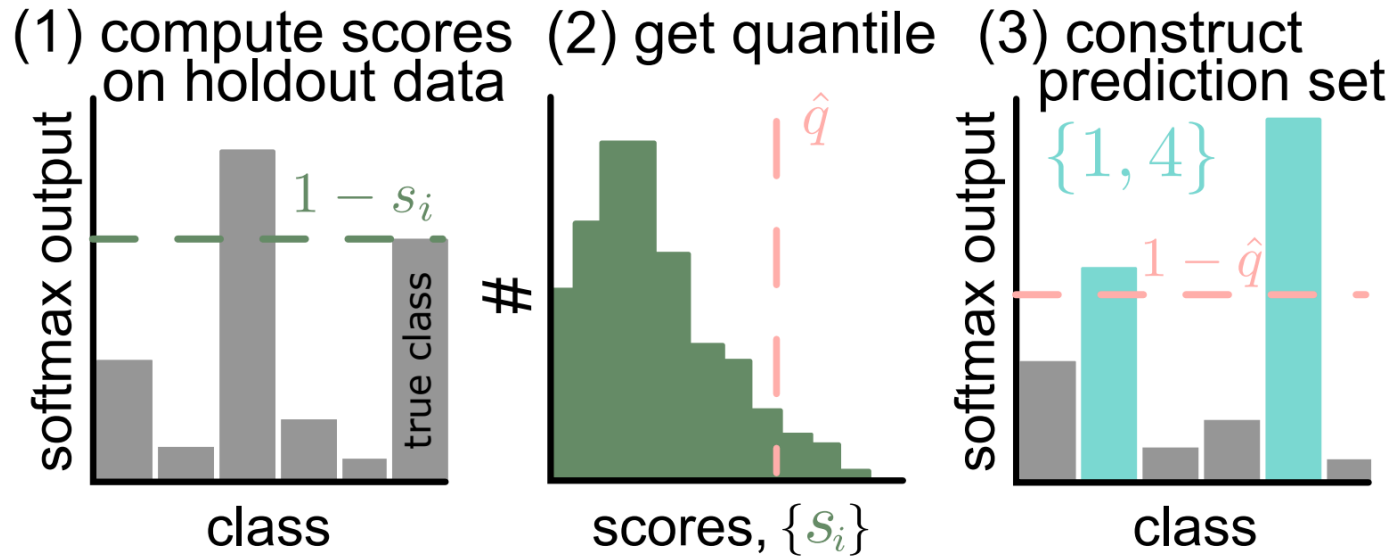
$n = 1000, \alpha = 0.1$

$k = \lceil 1001 \times 0.9 \rceil = 901$

$q_level = 0.901$

$\Rightarrow \hat{q}$ is the 901st smallest of the 1000 scores

Material: <https://arxiv.org/pdf/2107.07511>

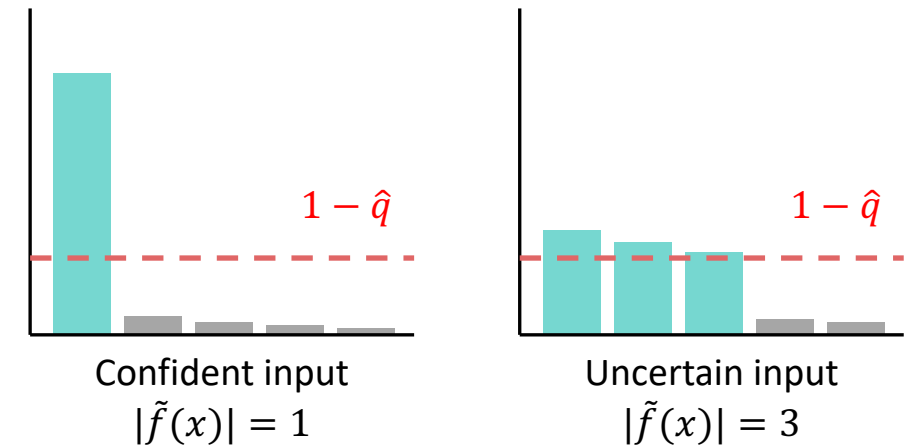


```
# 1: get conformal scores. n = calib_Y.shape[0]
cal_smx = model(calib_X).softmax(dim=1).numpy()
cal_scores = 1-cal_smx[np.arange(n),cal_labels]
# 2: get adjusted quantile
q_level = np.ceil((n+1)*(1-alpha))/n
qhat = np.quantile(cal_scores, q_level, method='higher')
val_smx = model(val_X).softmax(dim=1).numpy()
prediction_sets = val_smx >= (1-qhat) # 3: form prediction sets
```

- We use a hold-out/val set since prediction probabilities on the training set are not reliable.
- Conformal score $(s_i) = 1 - \hat{p}_i$
 - But we can devise other measures based on x and y
- Set threshold $\hat{q}$ as $\hat{q} = 1 - \alpha$ quantile of the scores
- Interpretation: if $\alpha = 0.1$, 90% of GT softmax outputs are above $1 - \hat{q}$.

Properties of the Prediction Set

- **Set size $\leftrightarrow$ uncertainty**: a small set means a confident model, a large set an uncertain one
- The set keeps the most plausible labels: all y with $f(y|x) \geq 1 - \hat{q}$
- **Coverage guarantee** (for exchangeable data):



$$1 - \alpha \leq P(Y_{test} \in \tilde{f}(X_{test})) \leq 1 - \alpha + \frac{1}{n + 1}$$

- **Lower bound**: always holds (needs only exchangeability)
- **Upper bound**: holds if scores have no ties $\Rightarrow$ sets are not overly conservative
- Example: $\alpha = 0.1, n = 1000 \Rightarrow \text{coverage} \in [0.900, 0.901]$
- Probability over both the calibration set and the test point (marginal)

Conformal Prediction for Any/All Problems

- Whether the inputs/outputs are discrete/continuous or not

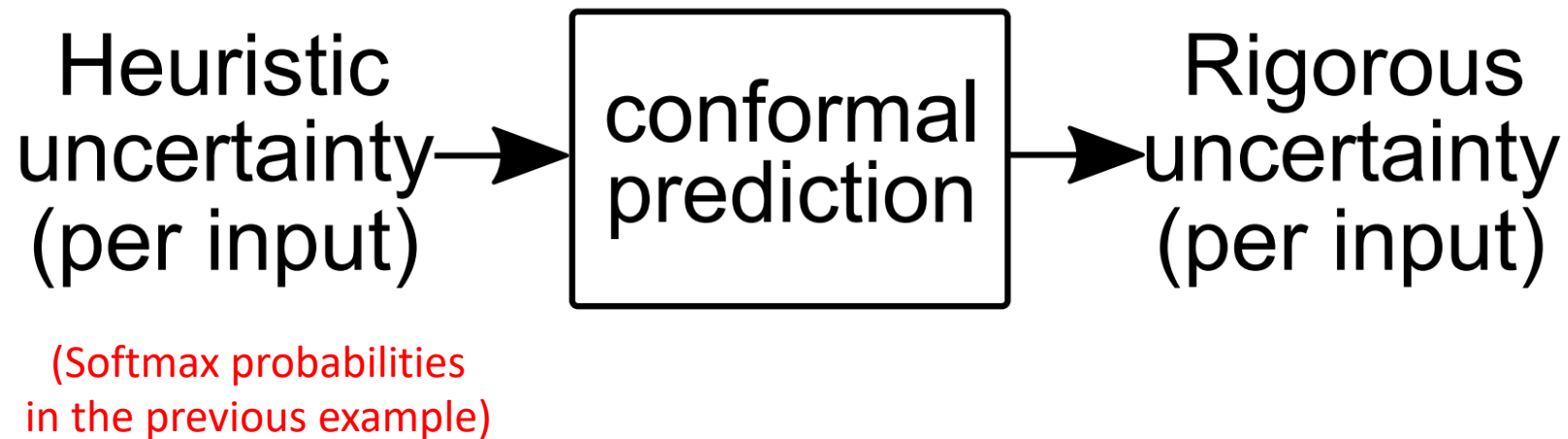


Fig & material: <https://arxiv.org/pdf/2107.07511>

We next outline conformal prediction for a general input x and output y (not necessarily discrete).

1. Identify a heuristic notion of uncertainty using the pre-trained model.
2. Define the score function $s(x, y) \in \mathbb{R}$. (Larger scores encode worse agreement between x and y .)
3. Compute $\hat{q}$ as the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile of the calibration scores $s_1 = s(X_1, Y_1), \dots, s_n = s(X_n, Y_n)$.
4. Use this quantile to form the prediction sets for new examples:

$$\mathcal{C}(X_{\text{test}}) = \{y : s(X_{\text{test}}, y) \leq \hat{q}\}. \quad (2)$$

Theorem 1 (Conformal coverage guarantee; Vovk, Gammerman, and Saunders [5]). *Suppose $(X_i, Y_i)_{i=1, \dots, n}$ and $(X_{\text{test}}, Y_{\text{test}})$ are i.i.d. and define $\hat{q}$ as in step 3 above and $\mathcal{C}(X_{\text{test}})$ as in step 4 above. Then the following holds:*

$$P\left(Y_{\text{test}} \in \mathcal{C}(X_{\text{test}})\right) \geq 1 - \alpha.$$

- Restriction on s_i :
 - It cannot be random
 - It must already sort the predictions correctly

Marginal Coverage

- The coverage guarantee only holds on average across all data points, also called “marginal” coverage.
- There is no guarantee that the coverage holds for each variety.
- Coverage = percentage of prediction sets that contain the true label

Intuition for Conformal Prediction

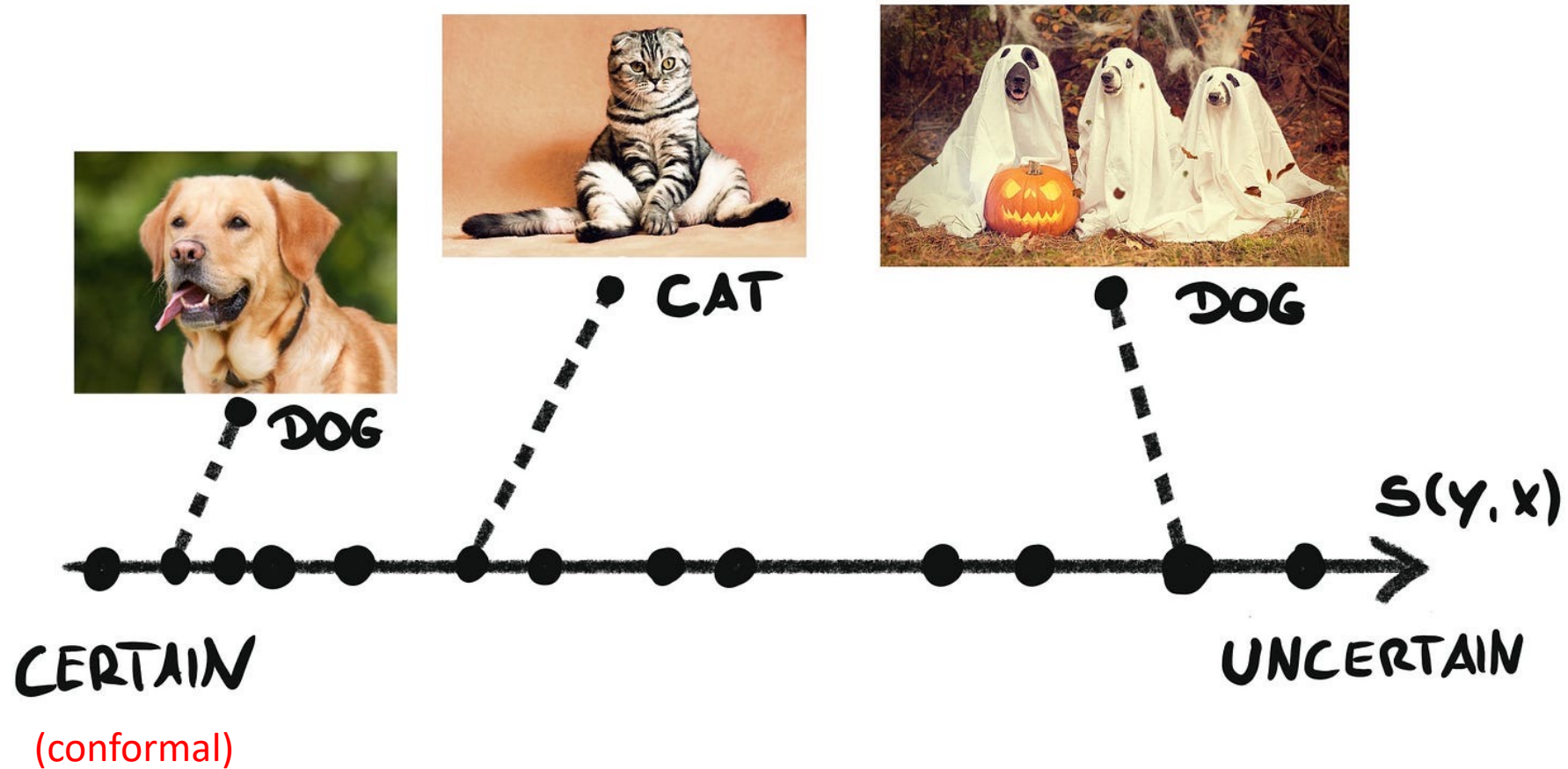
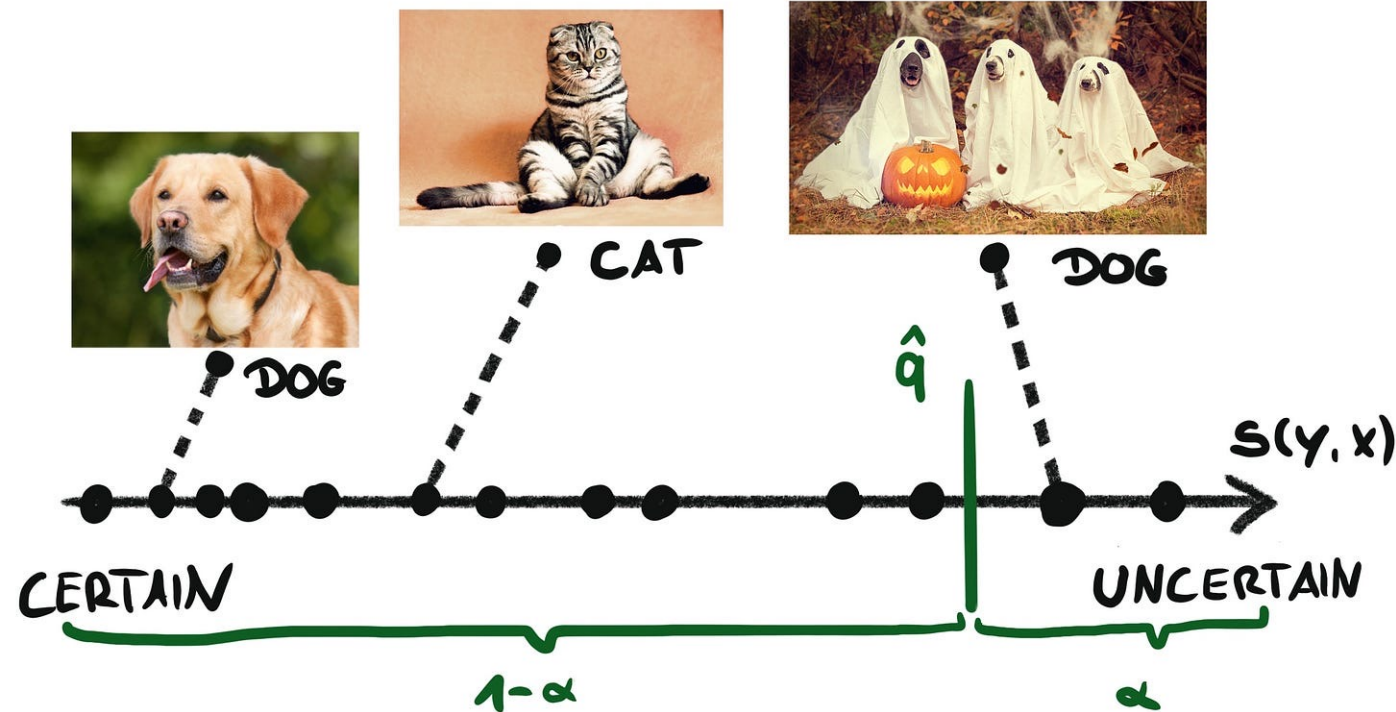


Fig: <https://mindfulmodeler.substack.com/p/week-2-intuition-behind-conformal>

Intuition for Conformal Prediction



“If $\alpha = 0.1$, then we want to have 90% of the non-conformity scores in the “certain” section. Finding the threshold is easy because it means computing the quantile q : the score value where 90% of images are below and 10% are above”

Material: <https://mindfulmodeler.substack.com/p/week-2-intuition-behind-conformal>

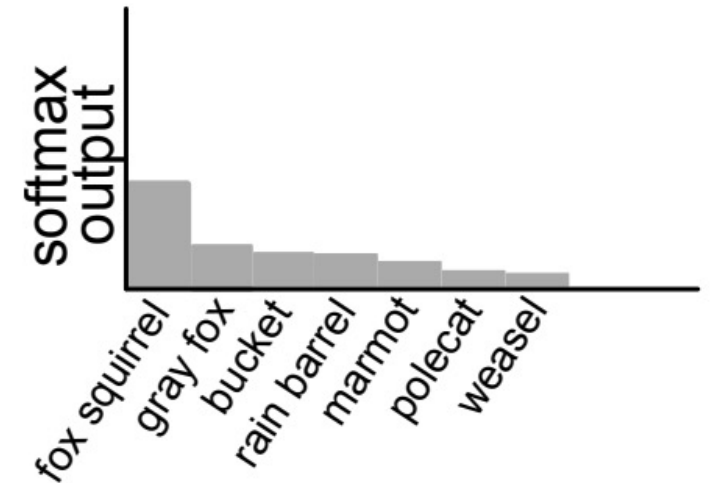
Adaptive Prediction Sets

- **Problem:** the score $1 - f(y^*|x)$ uses one threshold for all inputs $\Rightarrow$ sets too large for easy inputs, too small for hard ones ("average" coverage)
- **Goal:** conditional coverage, i.e., coverage for every input, not only on average
- **Idea (APS):** score = softmax mass collected, in sorted order, until the true class is reached
 - confident input: $\hat{q}$ is reached after 1–2 classes $\Rightarrow$ small set
 - uncertain input: mass is spread out $\Rightarrow$ many classes $\Rightarrow$ large set

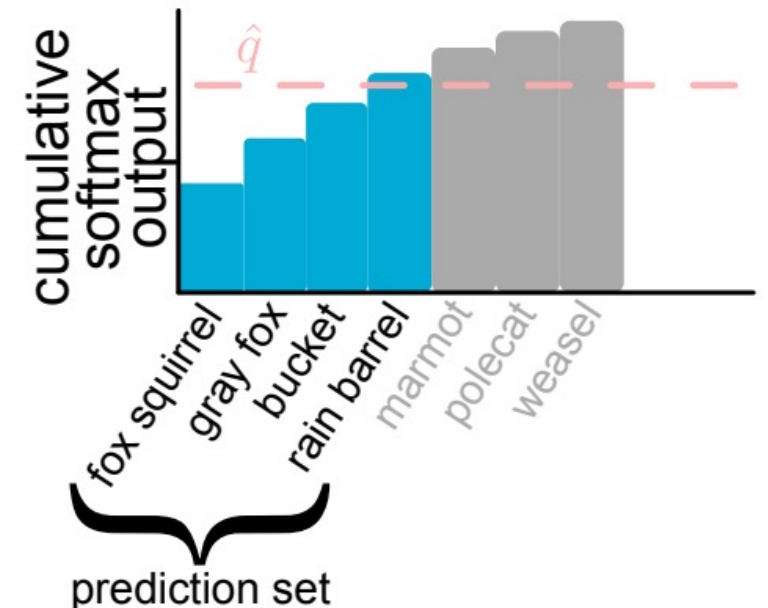
Marginal (have): $P(Y_{test} \in \tilde{f}(X_{test})) \geq 1 - \alpha$

Conditional (want): $P(Y_{test} \in \tilde{f}(X_{test}) | X_{test} = x) \geq 1 - \alpha \quad \forall x$

Exact conditional coverage is impossible in general; APS approximates it.



softmax of one hard input, sorted



Adaptive Prediction Sets: The Algorithm

Sort the classes of an input x by softmax:

$$f(y_1|x) \geq f(y_2|x) \geq \dots \geq f(y_K|x)$$

(1) Score: Sum up to and including the true class

$$s_i = \sum_{j=1}^r f(y_j|x_i), \quad \text{where } y_r = y_i^*$$

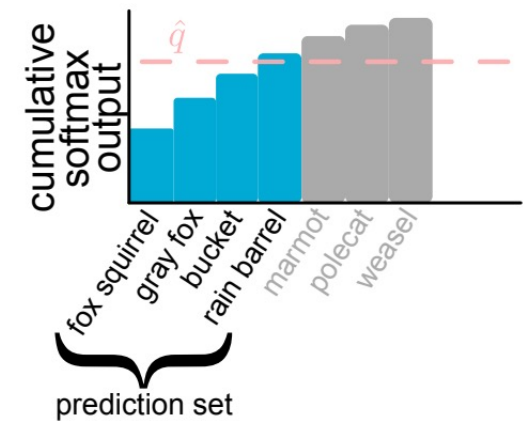
(2) Threshold: exactly as before

$$\hat{q} = s_{(k)}, \quad k = \lceil (n+1)(1-\alpha) \rceil$$

(3) Predict: add classes in sorted order while the mass stays $\leq \hat{q}$

$$\tilde{f}(x_{test}) = \{y_1, \dots, y_r : \sum_{j=1}^r f(y_j|x_{test}) \leq \hat{q}\}$$

Same method with new score: $\tilde{f}(x) = \{y : s(x, y) \leq \hat{q}\}$



Example (illustrative, $\hat{q} = 0.85$)

Class	$f(y_j x)$	Cumul.	In set?
fox squirrel	0.40	0.40	yes
gray fox	0.15	0.55	yes
bucket	0.13	0.68	yes
rain barrel	0.12	0.80	yes
marmot	0.08	0.88	no
polecat	0.06	0.94	no
weasel	0.06	1.00	no

- First 4 classes form the set (as in the figure)
- Confident input, e.g. $f(y_1|x) = 0.9 \Rightarrow$ set of size 1
- Original APS randomizes the boundary class (omitted)

Conformal Prediction for Regression

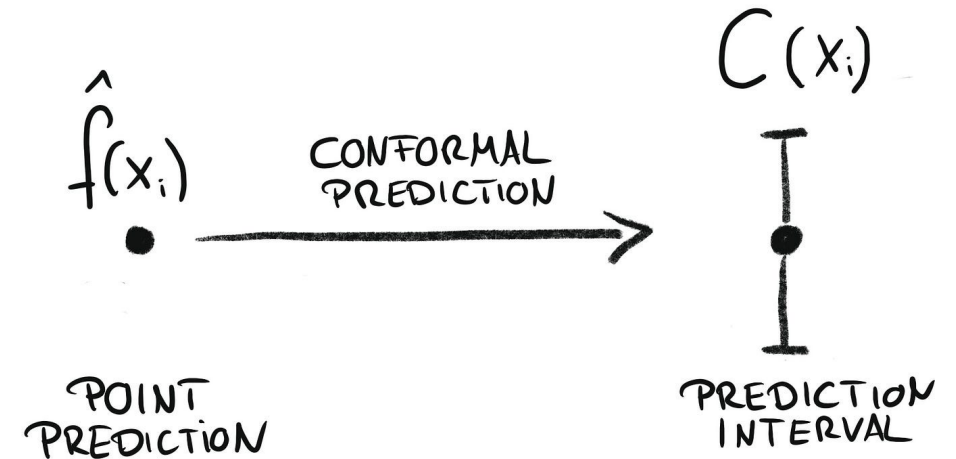
Goal: Turn a point prediction $\hat{f}(x)$ into an interval $C(x)$ that contains the true y with probability $\geq 1 - \alpha$.

In other words: the true value lies in

$$C(x) = [\hat{f}(x) - \hat{q}, \hat{f}(x) + \hat{q}]$$

with probability $\geq 1 - \alpha$.

- $\hat{q}$: half-width, calibrated on held-out data (next slide)
- α : user-chosen error rate, e.g., $\alpha = 0.1 \Rightarrow 90\%$ intervals
- Applicable to any regressor; no retraining.

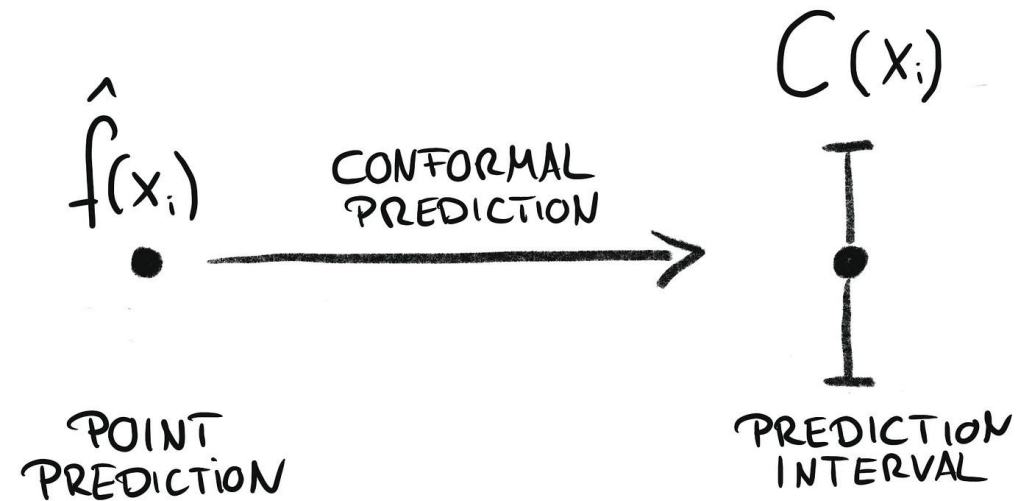


<https://mindfulmodeler.substack.com/p/week-3-conformal-prediction-for-regression>

Conformal Prediction for Regression

Goal: Turn a point prediction $\hat{f}(x)$ into an interval $C(x)$ that contains the true y with probability $\geq 1 - \alpha$.

- 1 **Split** the data: training set D_{train} , calibration set $Z_{val} = \{(x_i, y_i)\}_{i=1}^n$
- 2 **Train** any regressor $\hat{f}$ on D_{train}
- 3 **Score** the calibration set: $s_i = |y_i - \hat{f}(x_i)|$
- 4 **Threshold:** $\hat{q} = s_{(k)}$, $k = \lceil (n + 1)(1 - \alpha) \rceil$
- 5 **Predict:** $C(x) = [\hat{f}(x) - \hat{q}, \hat{f}(x) + \hat{q}]$



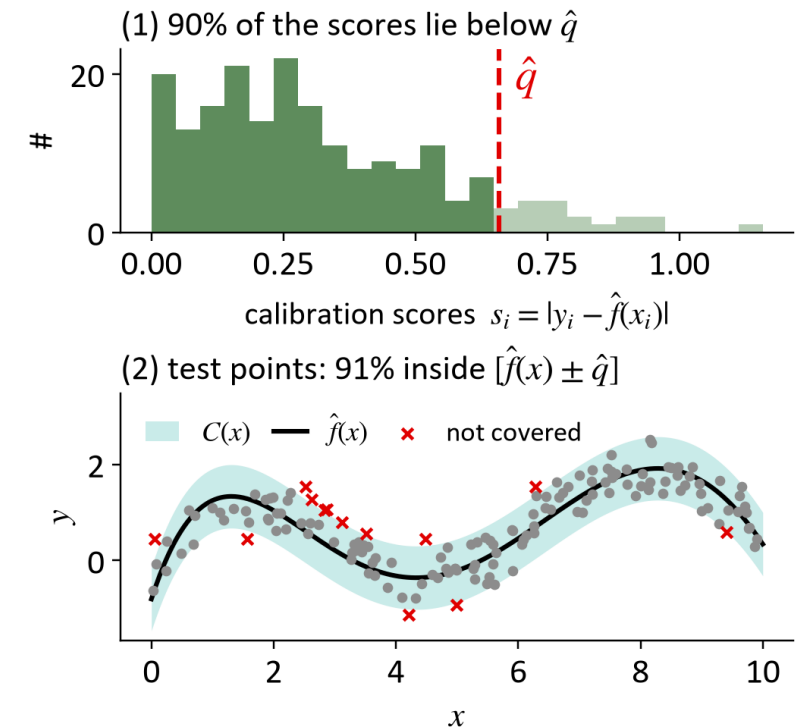
Steps 1–4 run once; step 5 costs one addition per test input

Conformal Prediction for Regression

From classification to regression, only the non-conformity score changes

	Classification	Regression
Model output	$f(y x)$ (softmax)	$\hat{f}(x)$ (point estimate)
Score	$s(x, y) = 1 - f(y x)$	$s(x, y) = y - \hat{f}(x) $
Prediction	$\tilde{f}(x) = \{y: s(x, y) \leq \hat{q}\}$ a set of labels	$C(x) = \{y: s(x, y) \leq \hat{q}\}$ $= [\hat{f}(x) - \hat{q}, \hat{f}(x) + \hat{q}]$ an interval

- Blog example (rent index): $\hat{q} = 1.97 \Rightarrow$ every interval has width $2\hat{q} = 3.94$
- Same width for every $x \Rightarrow$ not adaptive. Fix: $s(x, y) = |y - \hat{f}(x)|/\hat{\sigma}(x)$ (adaptive conformal regression)



Synthetic data, $\alpha = 0.1$, $n = 200$ calibration points

Adaptive Conformal Regression

Adaptive conformal regression: Instead of using residuals, we use standardized residuals. This approach requires a second model to estimate the variance and scale the residuals:

$$s(y, x) = \frac{|y - \hat{f}(x)|}{\hat{\sigma}(x)}$$

Theoretical Guarantee for Conformal Regression

Theorem (coverage of split conformal regression; Vovk et al., 2005; Lei et al., 2018).

Let $\hat{f}$ be trained on D_{train} , and let $(X_1, Y_1), \dots, (X_n, Y_n), (X_{test}, Y_{test})$ be exchangeable (e.g., i.i.d.) and independent of D_{train} . With $s_i = |Y_i - \hat{f}(X_i)|$, $\hat{q} = s_{(k)}$, $k = \lceil (n+1)(1-\alpha) \rceil$ and $C(X_{test}) = [\hat{f}(X_{test}) - \hat{q}, \hat{f}(X_{test}) + \hat{q}]$:

$$P(Y_{test} \in C(X_{test})) \geq 1 - \alpha$$

What it guarantees

- Any model $\hat{f}$, any data distribution (distribution-free)
- Finite-sample: holds for every n , no asymptotics
- Same proof works for any score (classification, APS, ...)

What it does not guarantee

- Coverage is **marginal**: averaged over X_{test} and the calibration set, not for each x
- A poor $\hat{f}$ still gives valid, but **wide**, intervals
- Breaks under **distribution shift**

Why Does the Guarantee Hold?

- 1 Coverage is an event about scores:

$$Y_{test} \in C(X_{test}) \Leftrightarrow s_{test} \leq \hat{q}$$

where $s_{test} = |Y_{test} - \hat{f}(X_{test})|$

- 2 $\hat{f}$ never saw the calibration or test points $\Rightarrow$
 $s_1, \dots, s_n, s_{test}$ are exchangeable $\Rightarrow s_{test}$ is equally
likely to take any rank among the $n + 1$ scores

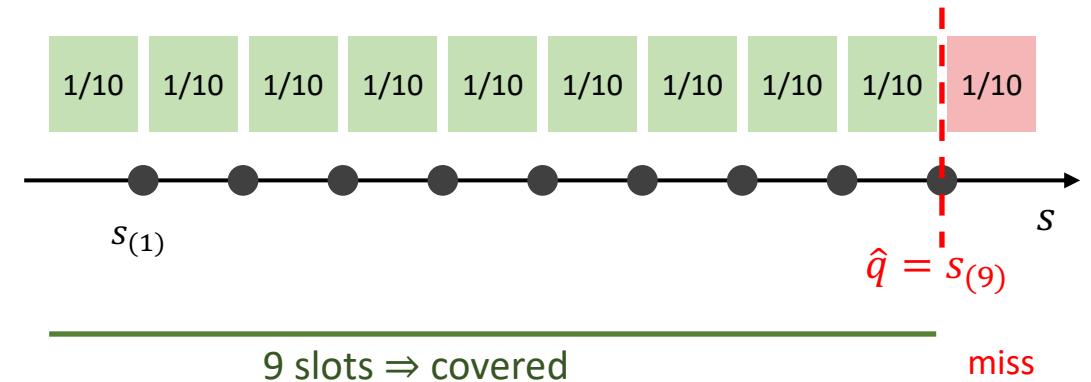
- 3 $\hat{q} = s_{(k)}$, so $s_{test} \leq \hat{q} \Leftrightarrow s_{test}$ lands in one of the k
lowest of the $n + 1$ slots

$$P(Y_{test} \in C(X_{test})) \geq \frac{k}{n+1} \geq 1 - \alpha$$

since $k = \lceil (n+1)(1-\alpha) \rceil \geq (n+1)(1-\alpha)$; without ties it equals $k/(n+1) \Rightarrow$ upper bound

Example: $n = 9, \alpha = 0.1 \Rightarrow k = \lceil 10 \times 0.9 \rceil = 9$

possible positions of s_{test} , each with probability $1/10$



$$P(\text{covered}) = \frac{9}{10} = 0.9 \geq 1 - \alpha$$

Larger n only makes the slots finer: $k/(n+1)$ always stays
within $1/(n+1)$ of $1 - \alpha$

Material: <https://arxiv.org/pdf/2107.07511> (proof of Theorem 1)