Towards Self-Certified Learning: Probabilistic Neural Networks trained by PAC-Bayes with Backprop

Omar Rivasplata

University College London

6 October 2022

Paper in JMLR (2021)

Tighter Risk Certificates for Neural Networks

María Pérez-Ortiz MARIA.PEREZ@UCL.AC.UK
AI Centre, University College London (UK)

Omar Rivasplata O.RIVASPLATA@CS.UCL.AC.UK
AI Centre, University College London (UK)

John Shawe-Taylor J.SHAWE-TAYLOR@UCL.AC.UK
AI Centre, University College London (UK)

Csaba Szepesvári SZEPI@GOOGLE.COM
DeepMind Edmonton (Canada)

Using PAC-Bayes bounds:

- For training NN classifiers.
- For certifying the risk of these classifiers.

Overview of this presentation

- Statistical Learning
  - PAC-Bayes bounds
    - Bayesian vs. PAC-Bayesian
      - NN classifiers: Learning and Certification
        - Highlight of results (tight certificates)

Classical Statistical Learning

Statistical Learning Framework

$\boxed{\text{ALG} : \mathcal{Z}^n \to \mathcal{W}}$ $\qquad$ $\boxed{\mathcal{W} \to \mathcal{H}}$

- $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$
  $\mathcal{X}$ = set of inputs
  $\mathcal{Y}$ = set of labels
- $\mathcal{W} \subset \mathbb{R}^p$
  weight space
  $\hat{w} = \text{ALG}(\text{data})$
- $\mathcal{H}$ function class
  predictors
  $h_{\hat{w}} : \mathcal{X} \to \mathcal{Y}$

data set: $S = (Z_1, \ldots, Z_n) \in \mathcal{Z}^n$ (e.g. training set)
an i.i.d sequence of input-label examples $Z_i = (X_i, Y_i)$.

Empirical risk: $\hat{L}_S(w) = \dfrac{1}{n} \sum_{i=1}^{n} \ell(w, Z_i)$
(in-sample error)

The Risk: $\boxed{L(w) = \mathbb{E}[\ell(w, Z)]} = \int_{\mathcal{Z}} \ell(w, z)\, dP(z)$
(out-of-sample)

Empirical Risk Minimization

$S_{\text{trn}} = (Z_1, \ldots, Z_{n\_\text{trn}})$ training set

Training loss: $L_{\text{trn}}(w) = \frac{1}{n\_\text{trn}} \sum_{Z_i \in S_{\text{trn}}} \ell(w, Z_i)$

($L_{\text{trn}} = \hat{L}_{S_{\text{trn}}}$ w/ some loss function ℓ)

ERM: $\hat{w} \in \arg\min_w L_{\text{trn}}(w)$

Penalized ERM: $\hat{w} \in \arg\min_w L_{\text{trn}}(w) + \text{Reg}(w)$

- Tied to the choice of a loss function $\ell(w, z)$

  e.g. 0-1 loss (classification), square loss (regression)
  cross-entropy loss (NN classification) ▹ surrogate loss

Test set error

$S_{\text{tst}} = (Z_1, \ldots, Z_{n\_\text{tst}})$ test set

Test set error: (with $\ell = \ell^{01}$)

$$L_{\text{tst}}(\hat{w}) = \frac{1}{n\_\text{tst}} \sum_{Z_i \in S_{\text{tst}}} \ell(\hat{w}, Z_i)$$

- $\hat{w}$ obtained from the training set
- test set not used for training
- $L_{\text{tst}}(\hat{w})$ serves as point estimate of $L(\hat{w})$
- Note: $L_{\text{tst}}(\hat{w}) \neq L(\hat{w})$ (test set error $\neq$ risk)
- Fact: the risk $L(\hat{w})$ remains unknown!

Confidence bounds

Risk upper bound: For any given $\delta \in (0, 1)$,

with probability at least $1 - \delta$
(over random samples S of size n)

simultaneously for all w : $L(w) \leq \hat{L}_S(w) + \epsilon(n, \delta)$

- **Risk certification on a test set:** Use your favourite ALG to find $\hat{w} = \text{ALG}(\text{train\_set})$, and use the confidence bound and the test set to certify $\hat{w}$'s risk: $L(\hat{w}) \leq L_{\text{tst}}(\hat{w}) + \epsilon(n\_\text{tst}, \delta)$
- **Bound minimization:** Use the upper bound as an objective by searching a $\hat{w}$ that minimizes $L_{\text{trn}}(w) + \epsilon(n\_\text{trn}, \delta)$.

PAC-Bayes bounds

Distributions over predictors

- Based on data, learn a distribution over weights:

$$Q_S = \mathsf{ALG}(\text{data set } S)$$

- Prediction rule (randomized predictions):
  - draw $w \sim Q_S$ and predict with the chosen w.
  - each prediction with a fresh random draw.

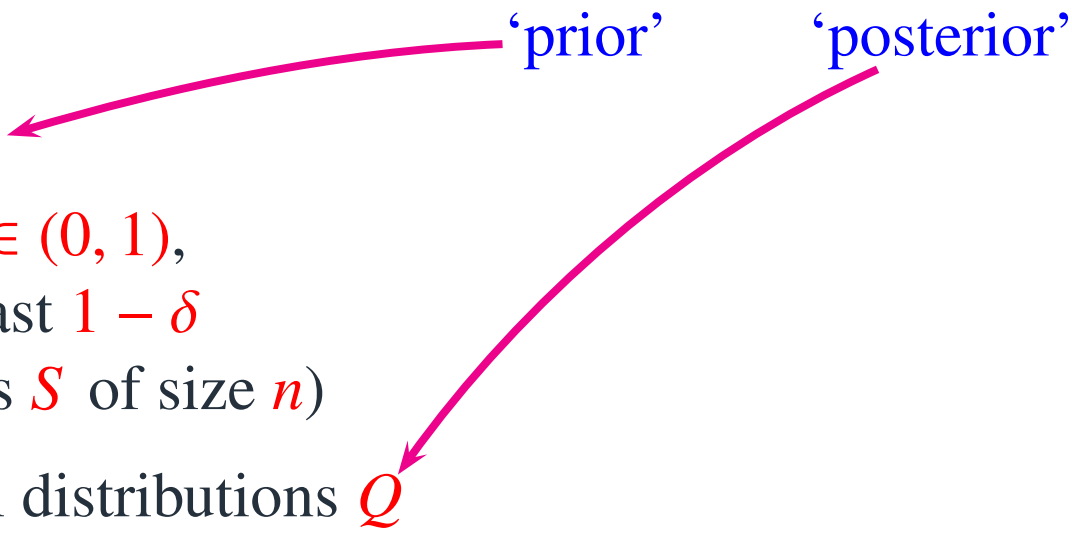
The risk measures $L(w)$ and $\hat{L}_S(w)$ are extended to Q by averaging:

$$L(Q) \equiv \int_W L(w)\, dQ(w) = \mathbb{E}_{w\sim Q}[L(w)]$$

$$\hat{L}_S(Q) \equiv \int_W \hat{L}_S(w)\, dQ(w) = \mathbb{E}_{w\sim Q}[\hat{L}_S(w)]$$

- Other prediction rules exist.

Classic PAC-Bayes bound (losses in [0,1])



Fix a distribution Q^0

for any confidence $\delta \in (0,1)$,
with probability at least $1-\delta$
(over random samples S of size n)

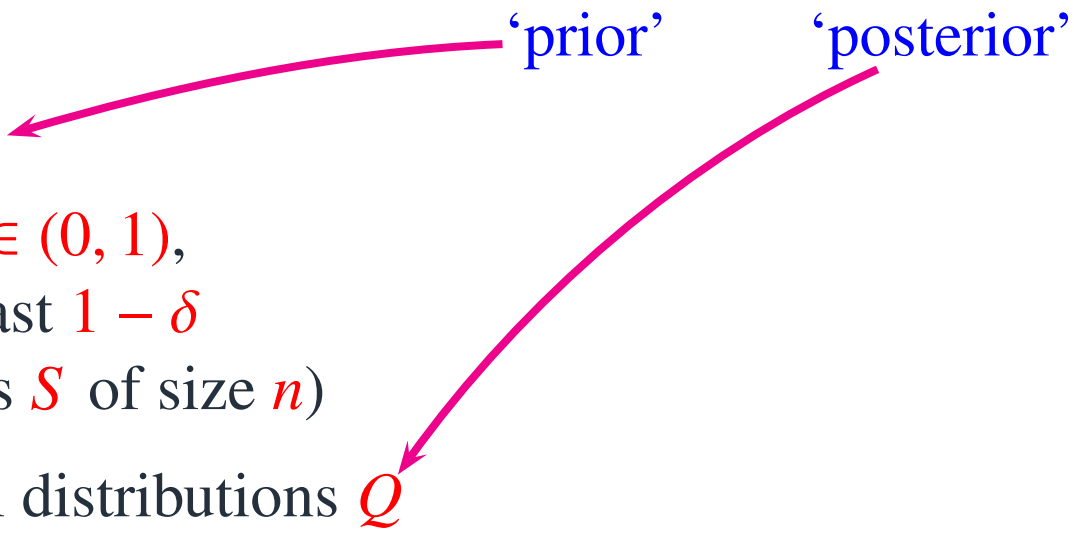
simultaneously for all distributions Q

$$L(Q) \le \hat{L}_S(Q) + \sqrt{\frac{\mathrm{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} \qquad \text{(PB-classic)}$$

Remarks:

- Original bound of McAllester [1999], [2003] had a slightly worse dependence on n
- Optimal dependence on n was given by Maurer [2004]

PAC-Bayes-kl bound (losses in [0,1])



Fix a distribution Q^0

for any confidence $\delta \in (0,1)$,
with probability at least $1-\delta$
(over random samples S of size n)

simultaneously for all distributions Q

$$\mathrm{kl}(\hat{L}_S(Q) \| L(Q)) \leq \frac{\mathrm{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{n} \qquad \text{(PB-kl)}$$

Remarks:

- Very tight bound
- Other PAC-Bayes bounds are relaxations of PB-kl
- Langford & Seeger [2001], Seeger [2002]

Two more PAC-Bayes bounds (losses in [0,1])

Fix a distribution Q_0. For any size n, for any confidence $\delta \in (0,1)$, with probability at least $1-\delta$ over random samples (of size n)

PB-quad: simultaneously for all distributions Q

$$L(Q) \le \left(\sqrt{\hat{L}_S(Q) + \frac{\mathrm{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} + \sqrt{\frac{\mathrm{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} \right)^2$$

PB-lambda: simultaneously for all distributions Q and $\lambda \in (0,2)$

$$L(Q) \le \frac{\hat{L}_S(Q)}{1-\lambda/2} + \frac{\mathrm{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{n\lambda(1-\lambda/2)}$$

Uses of a PAC-Bayes bound

- **Certification:** Use any ALG to find Q_S = ALG(data set S), plug Q_S into the PAC-Bayes bound to certify its risk:

$$L(Q_S) \leq \hat{L}_S(Q_S) + \sqrt{\frac{\mathrm{KL}(Q_S \| Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}}$$

- **Learning:** PAC-Bayes bound as a training objective:

$$Q_S \in \arg\min_Q \hat{L}_S(Q) + \sqrt{\frac{\mathrm{KL}(Q \| Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}}$$

Note: both uses illustrated here with McAllester's bound but the same can be done with other PAC-Bayes bounds.

Bayesian vs. PAC-Bayesian

Bayesian Learning

posterior Q_D, density $q_D(w)$ prior Q_0, density $q_0(w)$

$$q_D(w) = \mathcal{L}(D|w)\, q_0(w)\, /C$$

- Bayes rule update on prior to form posterior
  - likelihood factor $\mathcal{L}(D|w)$
- principled approach to deriving learning algorithms
  - e.g. MAP learning
- balance empirical loss and a regularization term
  - 'fit to data' versus 'fit to prior'

Generalized Bayes

A bit more general:
"temperature" $\lambda > 0$

$$q_D(w) = \mathcal{L}(D|w)^{\lambda}\, q_0(w)\ /C$$

Even more general:
data-dependent factor $\mathcal{F}$

$$q_D(w) = \mathcal{F}(D, w)\, q_0(w)$$

- P.G. Bissiri, C.C. Holmes, S.G. Walker (2016)
  A general framework for updating belief distributions

PAC-Bayes

$q_D(w)$ no update factor $q_0(w)$

- more general than generalized Bayes
- increased flexibility in choice of distributions
- balance $q_D[\hat{L}_D]$ and $\mathrm{KL}(q_D \| q_0)$
  - ‘fit to data’ versus ‘fit to prior’

Learning and Certification Strategies for Neural Network Classifiers

Paper in JMLR (2021)

Tighter Risk Certificates for Neural Networks

María Pérez-Ortiz MARIA.PEREZ@UCL.AC.UK
AI Centre, University College London (UK)

Omar Rivasplata O.RIVASPLATA@CS.UCL.AC.UK
AI Centre, University College London (UK)

John Shawe-Taylor J.SHAWE-TAYLOR@UCL.AC.UK
AI Centre, University College London (UK)

Csaba Szepesvári SZEPI@GOOGLE.COM
DeepMind Edmonton (Canada)

Some motivations:

- Blundel et al. [2015] ▹ 'Bayes by Backprop' (BBB), Variational Bayes
- Thiemann et al. [2017] ▹ PAC-Bayes-lambda, applied to SVMs (not NNs)
- Dziugaite & Roy [2017] ▹ non-vacuous bounds for NNs

Classic Neural Nets

Use the available data to:

(1) learn a weight vector $\hat{w}$
(2) certify $\hat{w}$'s performance

- split the data, part for (1) and part for (2)?
  - usual approach
- the whole of the data for (1) and (2) simultaneously?
  - **self-certified learning**!

Probabilistic Neural Nets

Use the available data to:

(1) learn a probability distribution over weights

(2) certify the predictions based on this distribution

Variational Bayes: $\min_\theta \mathrm{KL}(q_\theta(w)||p(w|D))$

- Objective : $f(\theta) = \mathbb{E}_{q_\theta(w)}[-\log(p(D|w))] + \mathrm{KL}(q_\theta(w)||p(w))$
- Algorithm : 'Bayes by Backprop' (BBB)

PAC-Bayes:

- Objective : A PAC-Bayes bound
- Algorithm : 'PAC-Bayes with Backprop' (PBB)
- Learning + certification

Learning and certification via PAC-Bayes

- Data-splitting protocol: $S_{\text{train}} = S_{\text{prior}} \uplus S_{\text{cert}}$ S_{test}
- Learning strategy:
  - PAC-Bayes bounds turned into training objectives
  - (PAC-Bayes) Prior learned on S_{prior}
  - (PAC-Bayes) Posterior learned on the whole S_{train}
- Certification strategy:
  - PAC-Bayes-kl bound (very tight!) evaluated on S_{cert}
- Highlights:
  - Competitive accuracy w.r.t. Bayesian and ERM
    - accuracy evaluated on S_{test}
  - Tight risk certificates!

Training objectives

$$f_{\text{classic}}(Q) = \hat{L}_S^{\text{ce}}(Q) + \sqrt{\frac{\text{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}}$$

$$f_{\text{quad}}(Q) = \left(\sqrt{\hat{L}_S^{\text{ce}}(Q) + \frac{\text{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} + \sqrt{\frac{\text{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{2n}} \right)^2$$

$$f_{\text{lambda}}(Q, \lambda) = \frac{\hat{L}_S^{\text{ce}}(Q)}{1 - \lambda/2} + \frac{\text{KL}(Q\|Q^0) + \log(\frac{2\sqrt{n}}{\delta})}{n\lambda(1 - \lambda/2)}$$

Optimization: PAC-Bayes with Backprop

Algorithm 1 PAC-Bayes with Backprop (PBB)

Require:

μ_0 ▷ Prior center parameters

ρ_0 ▷ Prior scale hyper-parameter

$Z_{1:n}$ ▷ Training examples (inputs + labels)

$\delta \in (0, 1)$ ▷ Confidence parameter

$\alpha \in (0, 1),\ T$ ▷ Learning rate; Number of iterations

Ensure: Optimal μ, ρ ▷ Centers, scales

procedure PB_QUAD_GAUSS

  $\mu \leftarrow \mu_0$ ▷ Set init. posterior center to prior center

  $\rho \leftarrow \rho_0$ ▷ Set init. posterior scale to prior scale

  for $t \leftarrow 1 : T$ **do** ▷ Run SGD for T iterations.

    Sample $V \sim \mathcal{N}(0, I)$

    $W = \mu + \log(1 + \exp(\rho)) \odot V$

    $f(\mu, \rho) = f_{\text{quad}}(Z_{1:n}, W, \mu, \rho, \mu_0, \rho_0, \delta)$

    SGD gradient step using $\begin{bmatrix} \nabla_\mu f \\ \nabla_\rho f \end{bmatrix}$

  end for

  return μ, ρ

end procedure

Certification via the PAC-Bayes-kl bound

Binary KL (right) inversion:
For $x \in [0,1]$ and $b \in [0,\infty)$.

$$f^\star(x,b) = \sup\{y \in (x,1] \ : \ \mathrm{kl}(x\|y) \le b\}$$

Invert the PB-kl bound:
(w.p. $\ge 1-\delta$ over data samples)

$$L^{01}(Q) \le f^\star\Big(\hat{L}_S^{01}(Q), \frac{\mathrm{KL}(Q\|Q^0)+\log(\frac{2\sqrt{n}}{\delta})}{n}\Big)$$

Use a Chernoff bound:
(w.p. $\ge 1-\delta'$ over MC weight samples)

$$\hat{L}_S^{01}(Q) \le f^\star\Big(\hat{L}_S^{01}(\hat{Q}_m), \tfrac{1}{m}\log(\tfrac{2}{\delta'})\Big)$$

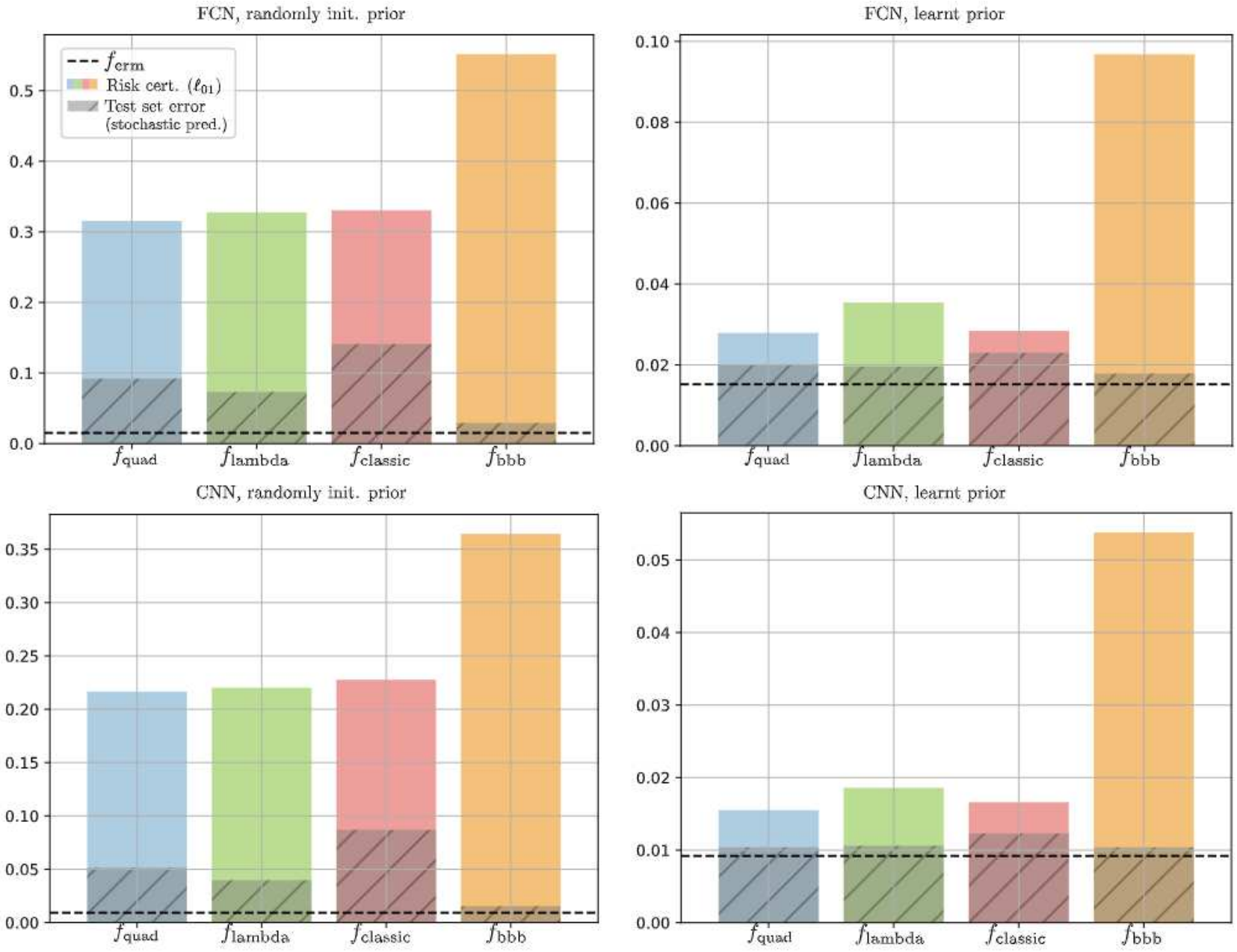
Altogether, with probability of at least $1-\delta-\delta'$:

$$L^{01}(Q) \le f^\star\left(f^\star\Big(\hat{L}_S^{01}(\hat{Q}_m), \frac{1}{m}\log(\frac{2}{\delta'})\Big), \frac{\mathrm{KL}(Q\|Q^0)+\log(\frac{2\sqrt{n}}{\delta})}{n}\right)$$

Summary of experiments

- Choice of distributions:
  - Gaussian vs Laplace.
  - data-free priors vs data-dependent priors (prior mean learned learnt by ERM + dropout on a split of the data).
  - posterior distribution learned on the whole training set (posterior always the same kind as the prior).
- Training objectives: f_{classic}, f_{quad}, f_{lambda} (PAC-Bayesian), f_{bbb} (variational Bayes), and f_{erm} (plain ERM).
- Optimization: PAC-Bayes with Backprop.
- Predictors: stochastic, deterministic, and ensemble.
- Certification: PAC-Bayes-kl (stochastic predictor)
- Datasets: MNIST and CIFAR-10.
- Architectures: FCN and CNN.
- Hyperparameter tuning: 6 different hyperparameters chosen using the risk upper bound.

Tight certificates on MNIST



Model Selection on MNIST

Pearson correlation between 0-1 error and risk certificate is 0.977.

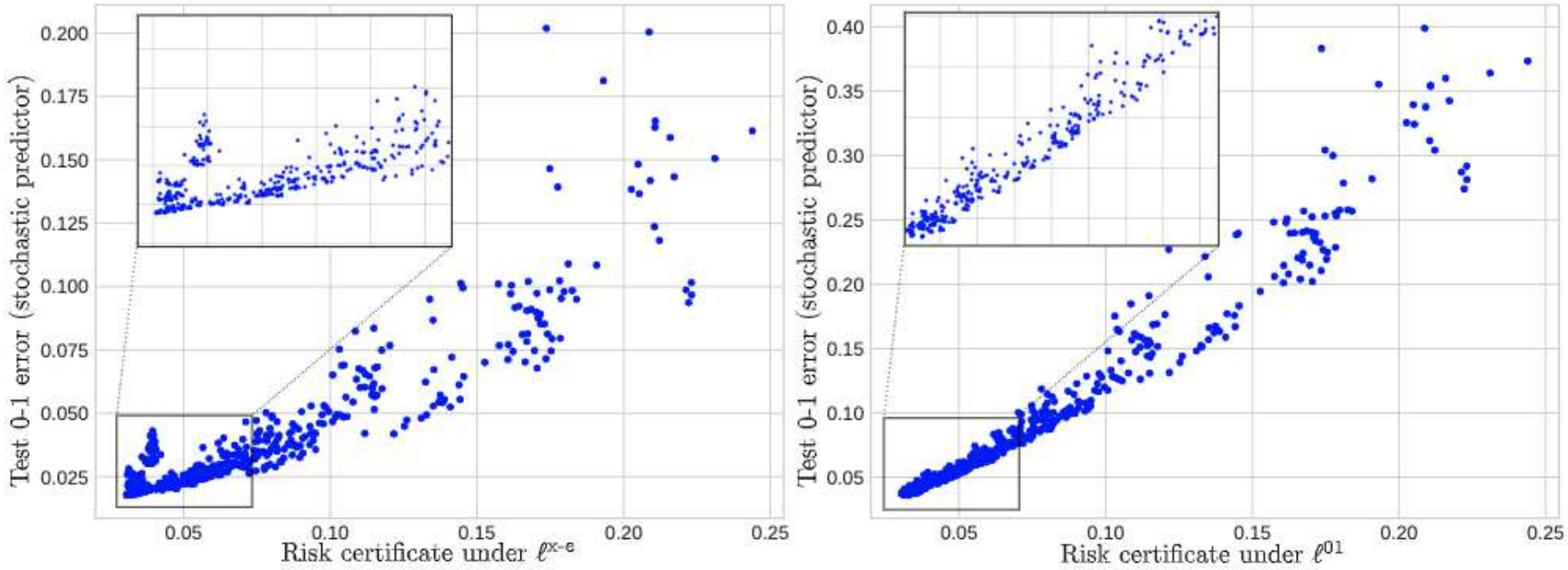


Figure 1: Model selection results from more than 600 runs with different hyper-parameters. The architecture used is a CNN with Gaussian data-dependent priors. We use a reduced subset of MNIST for these experiments (10% of training data).

Thank you!