Google AI
Brain Team

Hyperparameter Ensembles for Robustness and Uncertainty Quantification

Florian Wenzel florianwenzel@google.com
Jasper Snoek jsnoek@google.com
Dustin Tran trandustin@google.com
Rodolphe Jenatton rjenatton@google.com

AutoML Seminars, October 8th, 2020

Google AI

Collaborators

Google AI
Brain Team

Florian Wenzel

Jasper Snoek

Dustin Tran

Context and motivation

Google AI
Brain Team

ML systems are being deployed in many applications, e.g.,

- Conversational dialog systems, voice assistants, navigation systems
- Automatic content moderation [Link et al., 2016; Jhaver et al., 2019]

Context and motivation

Google AI
Brain Team

ML systems are being deployed in many applications, e.g.,

- Conversational dialog systems, voice assistants, navigation systems
- Automatic content moderation [Link et al., 2016; Jhaver et al., 2019]

Those applications can be in safety-critical areas :

- Health applications [Miotto et al., 2016; Rajkomar et al., 2018; Liu et al., 2020; Mckinney et al., 2020;...]
- Self-driving cars [Levinson et al., 2011; Sun et al., 2018]
- Risk assessments [Green et al., 2019]
- Automate decisions about benefit claims & welfare issues (e.g., the Guardian, 2019)

Context and motivation

Google AI
Brain Team

In those critical applications, we need robust uncertainty estimation

Context and motivation

Google AI
Brain Team

In those critical applications, we need robust uncertainty estimation

Why?

- Knowing when to trust model's predictions, e.g., under dataset shift

Context and motivation

Google AI
Brain Team

In those critical applications, we need robust uncertainty estimation

Why?

- Knowing when to trust model's predictions, e.g., under dataset shift
- Better decision making, e.g., with asymmetric costs
- Active learning: Getting more data in regions where the model is uncertain
- Open set recognition
- Lifelong learning
- Exploration in reinforcement learning, Bayesian optimization, ...

Application example (1/3)

Google AI
Brain Team

Dataset shift across:

- Time
- Countries

Image source: Hendrycks et al. 2020 “The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization”

Credits: slide from B. Lakshminarayanan, D. Tran, J. Snoek “Uncertainty and Out-of-Distribution Robustness in Deep Learning”

Application example (2/3)

Google AI
Brain Team

Conversational dialog systems

- Detecting out-of-scope utterances

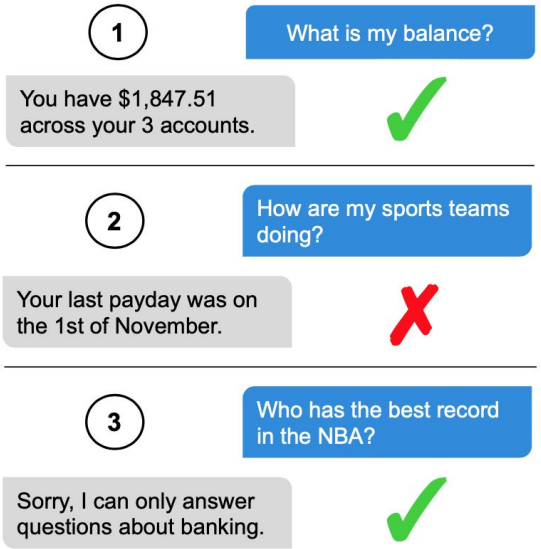


Figure 1: Example exchanges between a user (blue, right side) and a task-driven dialog system for personal finance (grey, left side). The system correctly identifies the user's query in (1), but in (2) the user's query is mis-identified as in-scope, and the system gives an unrelated response. In (3) the user's query is correctly identified as out-of-scope and the system gives a fallback response.

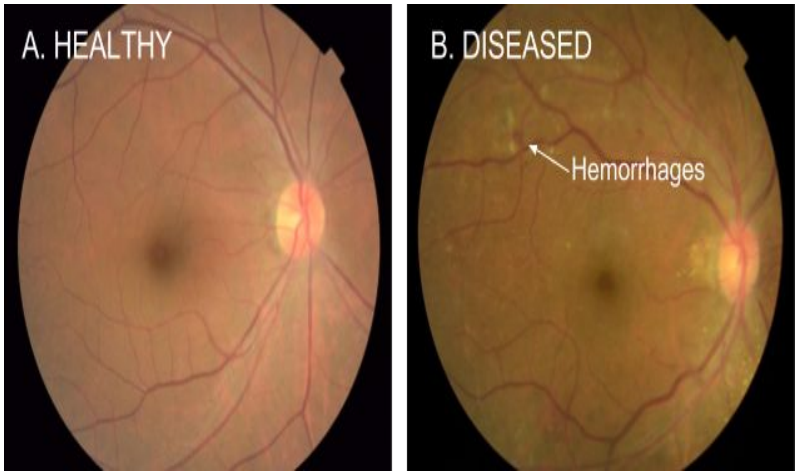
Image source: Larson et al. 2019 "An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction"

Credits: slide from B. Lakshminarayanan, D. Tran, J. Snoek "Uncertainty and Out-of-Distribution Robustness in Deep Learning"

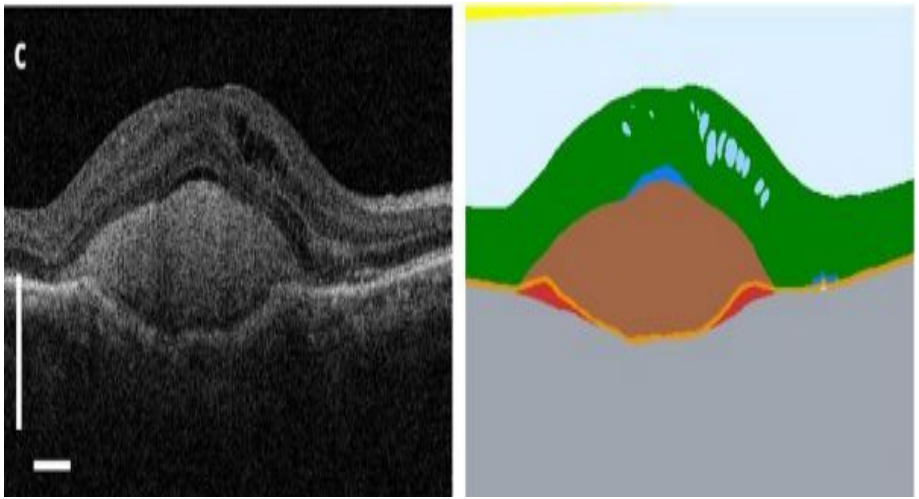
Application example (3/3)

Google AI
Brain Team

- Model uncertainty to decide when to trust model vs. when to defer to human



Diabetic retinopathy detection from fundus images
Gulshan et al, 2016



Eye disease classification from 3D OCT images
de Fauw et al, 2018

Credits: slide from B. Lakshminarayanan, D. Tran, J. Snoek "Uncertainty and Out-of-Distribution Robustness in Deep Learning"

But what about the behavior of modern neural networks?

Models accuracy degrades under dataset shift

Google AI Brain Team

- **Accuracy drops** with increasing shift on Imagenet-C

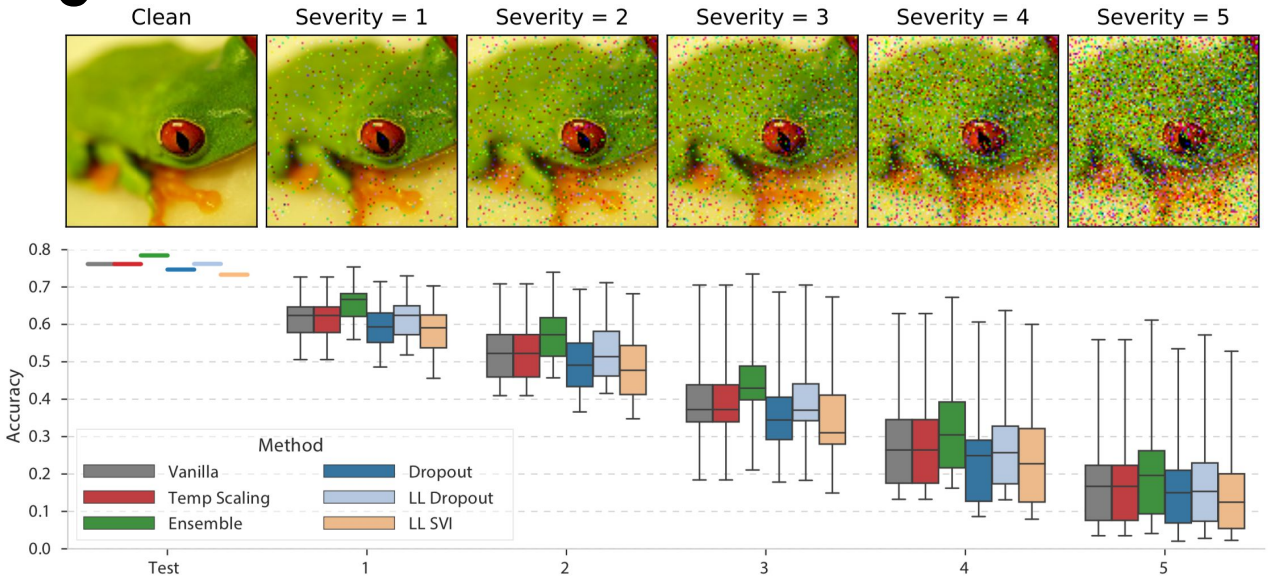
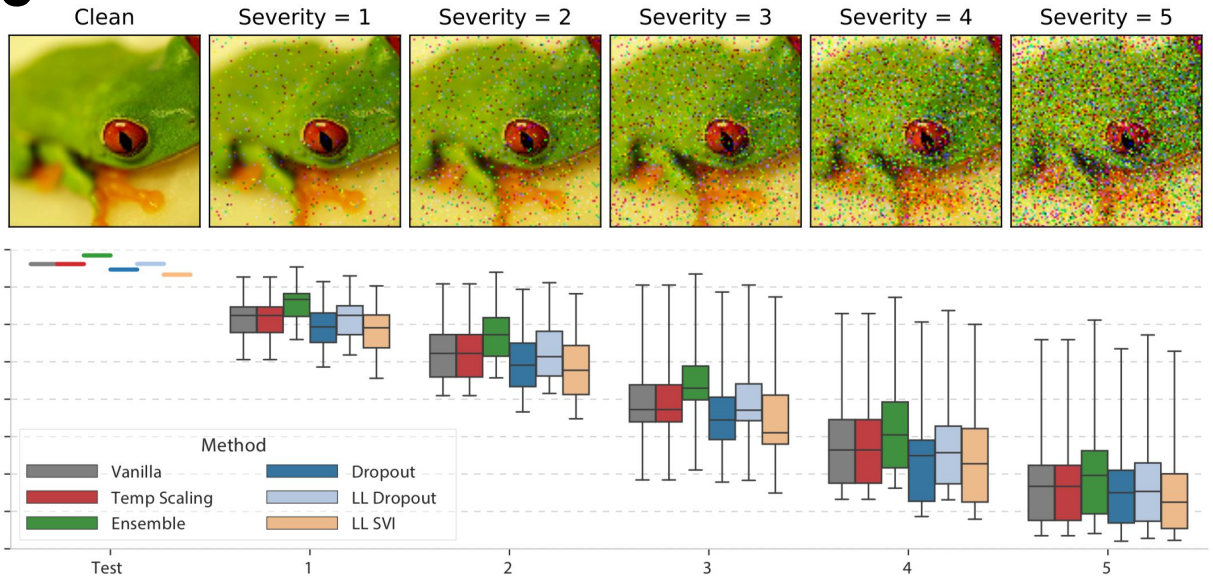


Image source: Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift?, Ovadia et al. 2019

Credits: slide from B. Lakshminarayanan, D. Tran, J. Snoek "Uncertainty and Out-of-Distribution Robustness in Deep Learning"

Models accuracy degrades under dataset shift

Google AI Brain Team



- **Accuracy drops** with increasing shift on Imagenet-C

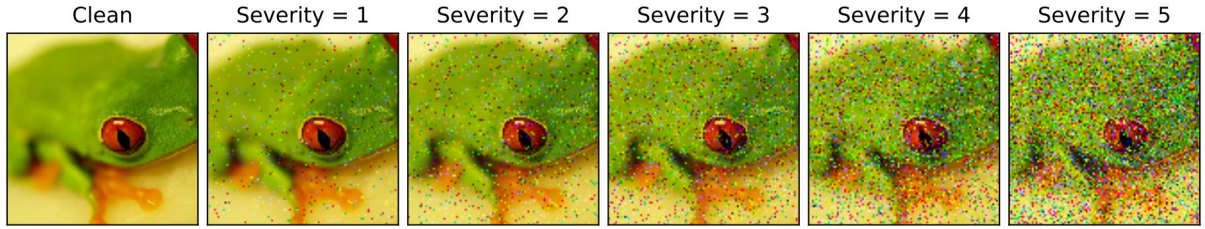
- But do the models know that they are less accurate?

Image source: Can You Trust Your Model's Uncertainty? Evaluating Predictive Uncertainty Under Dataset Shift?, Ovadia et al. 2019

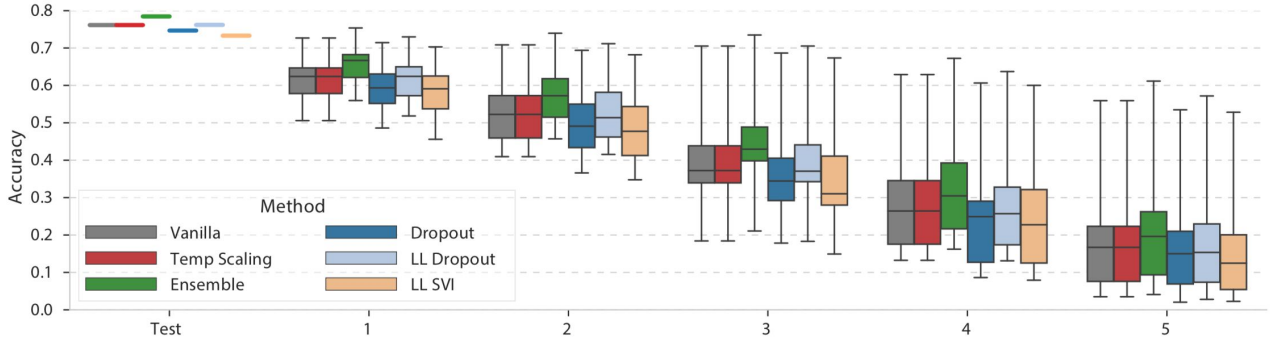
Credits: slide from B. Lakshminarayanan, D. Tran, J. Snoek "Uncertainty and Out-of-Distribution Robustness in Deep Learning"

Models are not calibrated under dataset shift

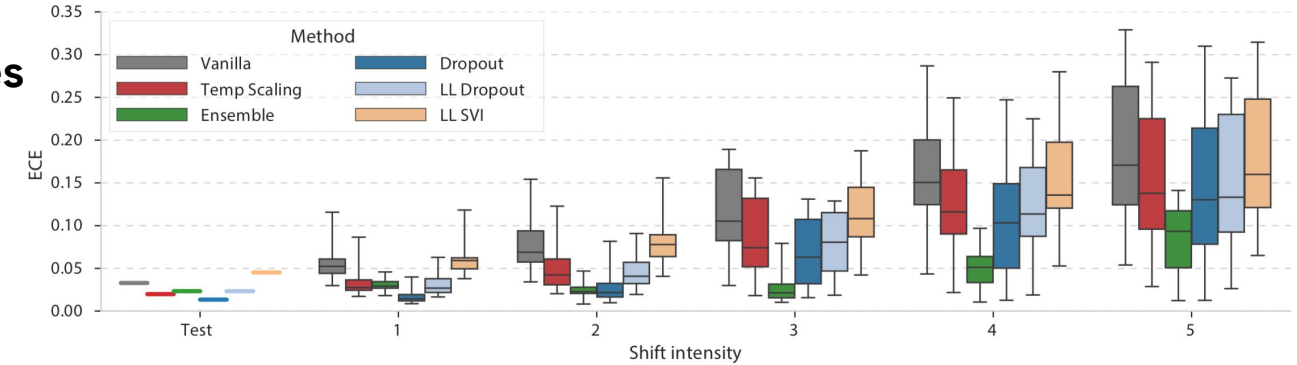
Google AI Brain Team



- **Accuracy drops** with increasing shift on Imagenet-C



- **Calibration degrades** with shift: "*overconfident mistakes*"



Credits: slide from B. Lakshminarayanan, D. Tran, J. Snoek "Uncertainty and Out-of-Distribution Robustness in Deep Learning"

Outline of the talk

Google AI
Brain Team

Hyper-deep ensemble

- From deep ensemble to hyper-deep ensemble
- Construction
- Evaluation

Hyper-batch ensemble

- Two key ingredients:
  - Batch ensemble
  - Self-tuning networks
- Layer structure & end-to-end ensemble training
- Evaluation

Outline of the talk

Google AI
Brain Team

Hyper-deep ensemble

- From deep ensemble to hyper-deep ensemble
- Construction
- Evaluation

Hyper-batch ensemble

- Two key ingredients:
  - Batch ensemble
  - Self-tuning networks
- Layer structure & end-to-end ensemble training
- Evaluation

Many approaches to capture uncertainty

Google AI
Brain Team

Various mechanisms:

- Data augmentation [Hendrycks et al., 2019; Yin et al., 2019; Thulasidasan et al., 2019]
- Architectures [Nado et al., 2020]
- Loss functions [Muller et al., 2019; Meinke et al., 2019]
- Generative models [Nalisnick et al., 2019; Ren et al., 2019]

Many approaches to capture uncertainty

Various mechanisms:

- Data augmentation [Hendrycks et al., 2019; Yin et al., 2019; Thulasidasan et al., 2019]
- Architectures [Nado et al., 2020]
- Loss functions [Muller et al., 2019; Meinke et al., 2019]
- Generative models [Nalisnick et al., 2019; Ren et al., 2019]

“Ensembles”: Generate a set of diverse members

- Bayesian NNs [Hinton et al., 1993; Neal, 1995; MacKay et al., 1995; Barber et al., 1998; ...; Wenzel et al. 2020]
- Long history [Hansen et al., 1990; Levin et al., 1990; Geman et al., 1992;...; Dietterich, 2000; ...]
  - Initialization, bootstraps [Lee et al., 2015; Lakshminarayanan et al., 2017]
  - Optimization [Huang et al., 2017; Loshchilov et al., 2017; Zhang et al., 2019]
  - Hyperparameters [Caruana et al., 2004; ...; Feurer et al., 2015; Lévesque et al., 2016; Saikia et al. 2020]
  - Architectures [Zaidi+Zela et al., 2020; Antorán et al., 2020]

Many approaches to capture uncertainty

Various mechanisms:

- Data augmentation [Hendrycks et al., 2019; Yin et al., 2019; Thulasidasan et al., 2019]
- Architectures [Nado et al., 2020]
- Loss functions [Muller et al., 2019; Meinke et al., 2019]
- Generative models [Nalisnick et al., 2019; Ren et al., 2019]

“Ensembles”: Generate a set of diverse members

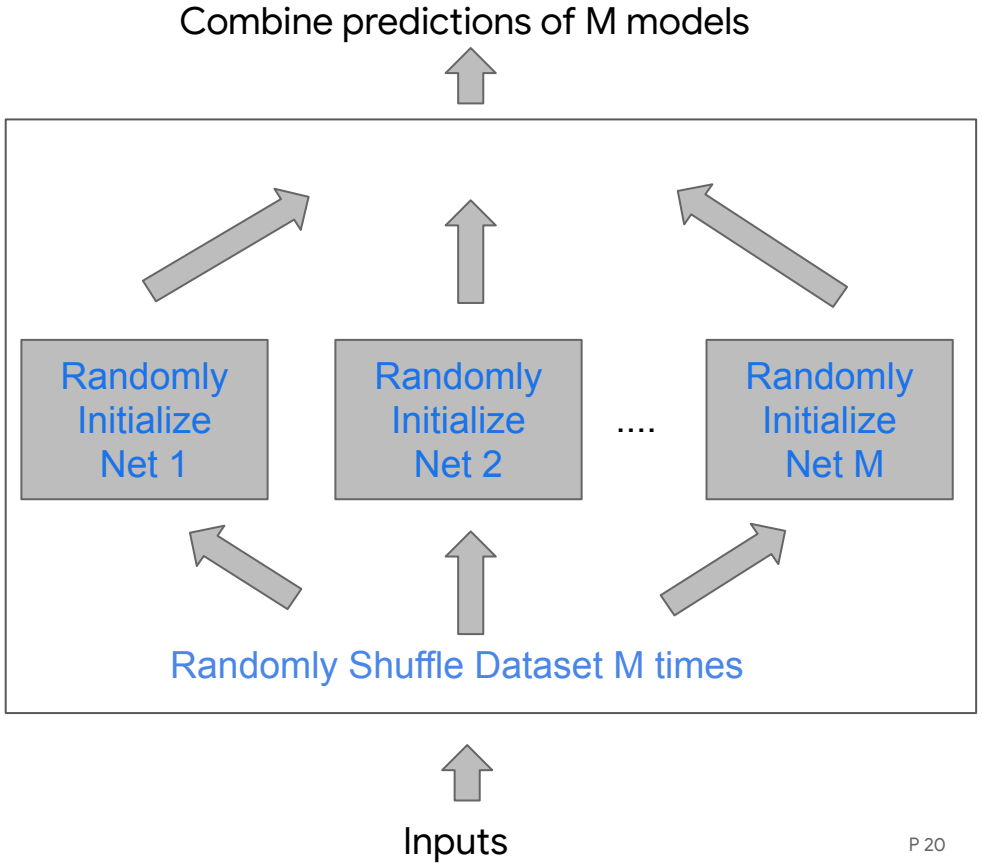
- Bayesian NNs [Hinton et al., 1993; Neal, 1995; MacKay et al., 1995; Barber et al., 1998; ...; Wenzel et al. 2020]
- Long history [Hansen et al., 1990; Levin et al., 1990; Geman et al., 1992;...; Dietterich, 2000; ...]
  - Initialization, bootstraps [Lee et al., 2015; Lakshminarayanan et al., 2017]
  - Optimization [Huang et al., 2017; Loshchilov et al., 2017; Zhang et al., 2019]
  - Hyperparameters [Caruana et al., 2004; ...; Feurer et al., 2015; Lévesque et al., 2016; Saikia et al. 2020]
  - Architectures [Zaidi+Zela et al., 2020, Antorán et al., 2020]

Deep ensemble [Lakshminarayanan et al. 2017]

Google AI Brain Team

Idea:

- Multiple trainings from different seeds
- Average the predictions
- Rely on
  - Non-convexity of the objective
  - Stochasticity of initialization
  - Stochasticity of optimization
- SOTA performance [Ovadia et al., 2019; Gustafsson et al., 2019]

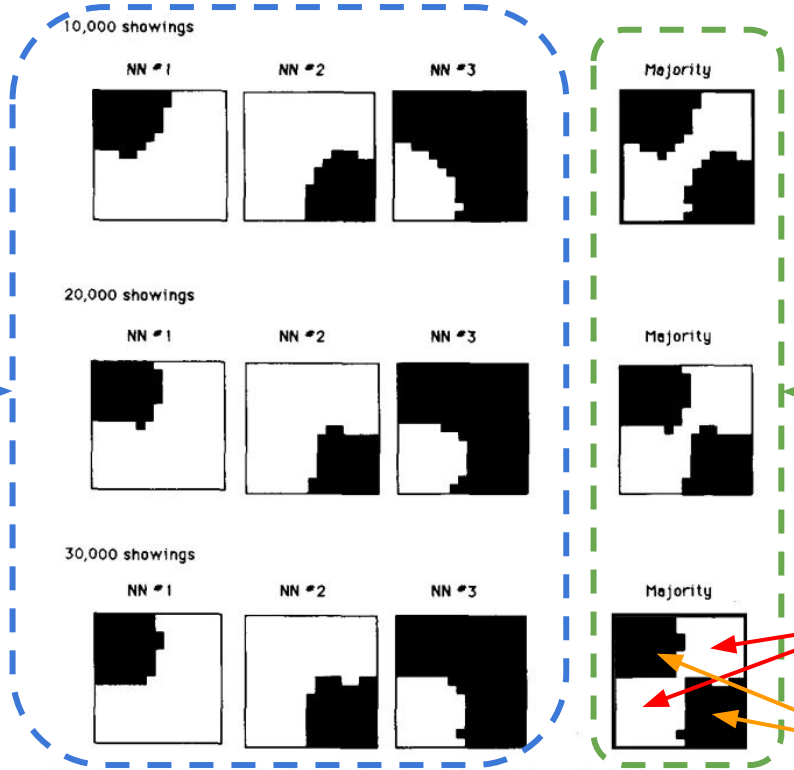


Credits: slide from B. Lakshminarayanan, D. Tran, J. Snoek "Uncertainty and Out-of-Distribution Robustness in Deep Learning"

Google AI
Brain Team

Neural Network Ensembles

LARS KAI HANSEN AND PETER SALAMON



Single model predictions

Ensemble predictions

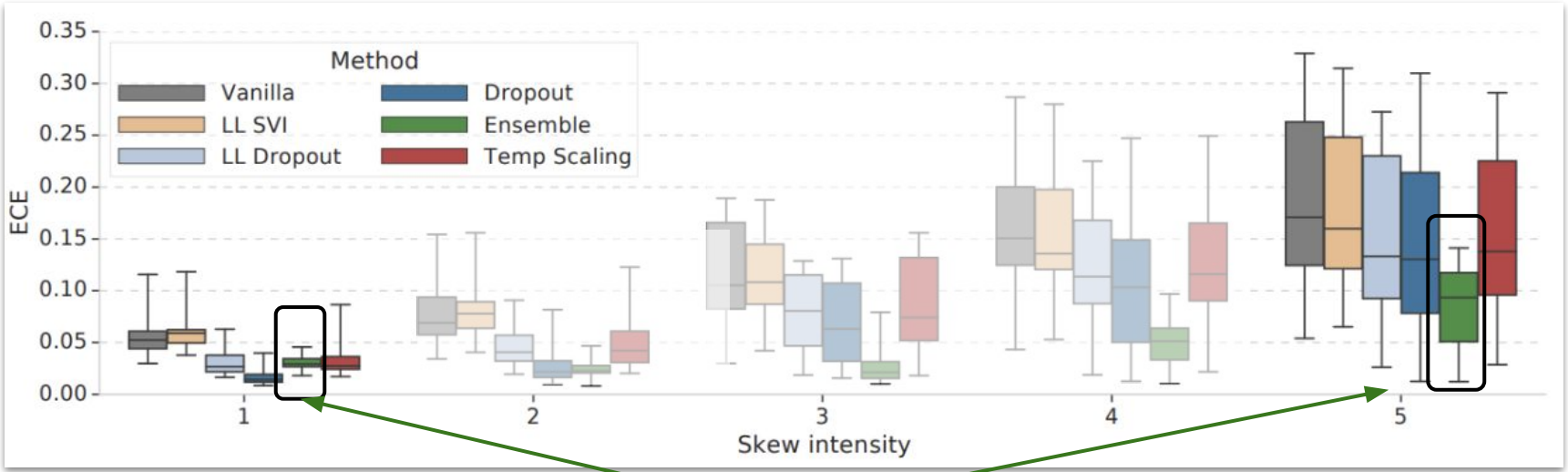
Classe 1

Classe 2

Fig. 1. Three networks trained to identify inputs (x, y) as belonging to the first and third quadrants (black) or the second and fourth quadrants (white). The figure shows the classification by each of the networks and their majority decisions.

Deep ensembles work surprisingly well in practice

Google AI
Brain Team



Deep ensembles are consistently among the best performing methods, especially under dataset shift

Credits: slide from B. Lakshminarayanan, D. Tran, J. Snoek "Uncertainty and Out-of-Distribution Robustness in Deep Learning"

Our contribution (part 1)

Google AI
Brain Team

- Show the importance of ensembling HPs for uncertainty modelling
- Comparison: deep ensemble vs. **hyper-deep ensemble**
- Simple algorithmic scheme (& little compute overhead)

Goal: Leverage diversity from both initializations *and* hyperparameters

Outline of the talk

Google AI
Brain Team

Hyper-deep ensemble

- From deep ensemble to hyper-deep ensemble
- Construction
- Evaluations

Hyper-batch ensemble

- Two key ingredients:
  - Batch ensemble
  - Self-tuning networks
- Layer structure & end-to-end ensemble training
- Evaluations

Hyper-deep ensembles

Google AI
Brain Team

Two simple ingredients

- **Random search:** Set of models with different hyperparameters
- **Greedy selection:** Sequentially add model minimizing ens. loss

Algorithm 1: `greedy`$(\mathcal{M},\ K)$ # Caruana et al. 2004

ensemble $\mathcal{E} = \{\ \}$, score $\mathcal{S}(\cdot)$, $\mathcal{S}_{\text{best}} = +\infty$;
while $|\mathcal{E}.\text{unique}()| \leq K$ **do**
  $f_{\boldsymbol{\theta}^\star} = \arg\min_{f_{\boldsymbol{\theta}} \in \mathcal{M}} \mathcal{S}(\mathcal{E} \cup \{f_{\boldsymbol{\theta}}\})$;
  if $\mathcal{S}(\mathcal{E} \cup \{f_{\boldsymbol{\theta}^\star}\}) < \mathcal{S}_{\text{best}}$ **then**
    $\mathcal{E} = \mathcal{E} \cup \{f_{\boldsymbol{\theta}^\star}\},\ \mathcal{S}_{\text{best}} = \mathcal{S}(\mathcal{E})$;
  else
    return $\mathcal{E}$;
  end
end
return $\mathcal{E}$;

Hyper-deep ensembles

Google AI
Brain Team

Two simple ingredients

- **Random search:** Set of models with different hyperparameters
- **Greedy selection:** Sequentially add model minimizing ens. loss

Algorithm 1: greedy($\mathcal{M}$, K) # Caruana et al. 2004

ensemble $\mathcal{E} = \{\}$, score $\mathcal{S}(\cdot)$, $\mathcal{S}_{\text{best}} = +\infty$;
while $|\mathcal{E}.\text{unique}()| \leq K$ **do**
  $f_{\theta^\star} = \arg\min_{f_\theta \in \mathcal{M}} \mathcal{S}(\mathcal{E} \cup \{f_\theta\})$;
  if $\mathcal{S}(\mathcal{E} \cup \{f_{\theta^\star}\}) < \mathcal{S}_{\text{best}}$ **then**
    $\mathcal{E} = \mathcal{E} \cup \{f_{\theta^\star}\}$, $\mathcal{S}_{\text{best}} = \mathcal{S}(\mathcal{E})$;
  else
    return $\mathcal{E}$;
  end
end
return $\mathcal{E}$;

Hyper-deep ensembles

Google AI
Brain Team

Two simple ingredients

- **Random search:** Set of models with different hyperparameters
- **Greedy selection:** Sequentially add model minimizing ens. loss

Algorithm 1: greedy($\mathcal{M}$, K) # Caruana et al. 2004

ensemble $\mathcal{E} = \{\}$, score $\mathcal{S}(\cdot)$, $\mathcal{S}_{\text{best}} = +\infty$;
while $|\mathcal{E}.\text{unique}()| \leq K$ **do**
  $f_{\theta^\star} = \arg\min_{f_\theta \in \mathcal{M}} \mathcal{S}(\mathcal{E} \cup \{f_\theta\})$;
  if $\mathcal{S}(\mathcal{E} \cup \{f_{\theta^\star}\}) < \mathcal{S}_{\text{best}}$ **then**
    $\mathcal{E} = \mathcal{E} \cup \{f_{\theta^\star}\}$, $\mathcal{S}_{\text{best}} = \mathcal{S}(\mathcal{E})$;
  else
    return $\mathcal{E}$;
  end
end
return $\mathcal{E}$;

Hyper-deep ensembles

Google AI
Brain Team

Two simple ingredients

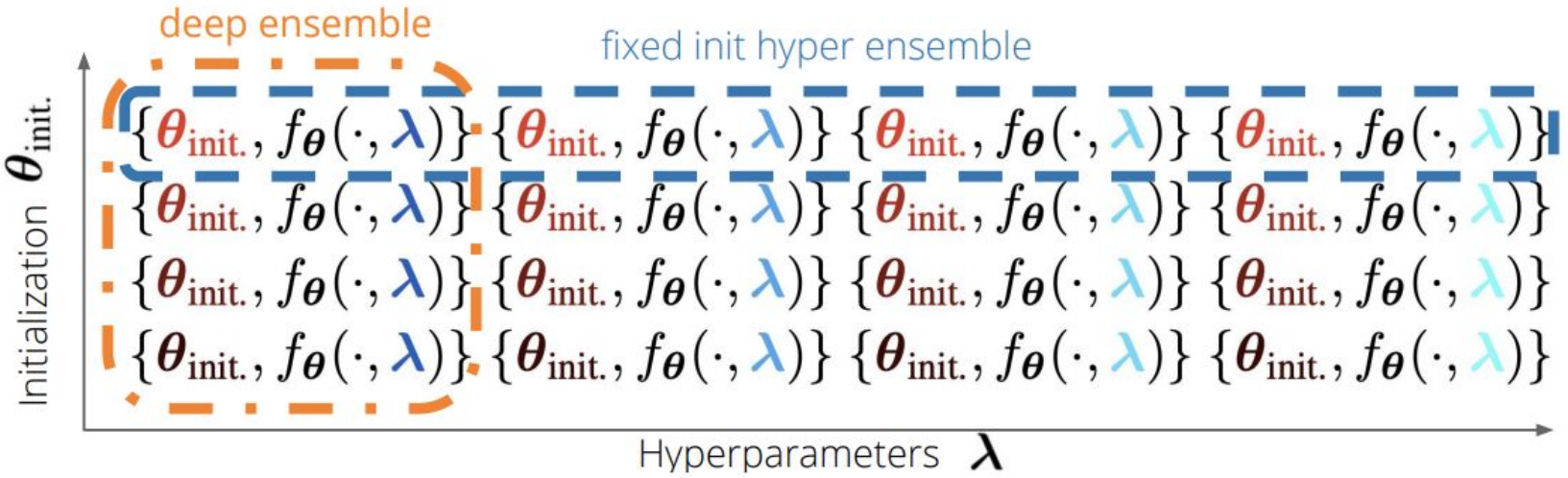
- **Random search:** Set of models with different hyperparameters
- **Greedy selection:** Sequentially add model minimizing ens. loss

Algorithm 1: greedy($\mathcal{M}$, K) # Caruana et al. 2004

ensemble $\mathcal{E} = \{\}$, score $\mathcal{S}(\cdot)$, $\mathcal{S}_{\text{best}} = +\infty$;
while $|\mathcal{E}.\text{unique}()| \leq K$ **do**
  $f_{\theta^\star} = \arg\min_{f_\theta \in \mathcal{M}} \mathcal{S}(\mathcal{E} \cup \{f_\theta\})$;
  if $\mathcal{S}(\mathcal{E} \cup \{f_{\theta^\star}\}) < \mathcal{S}_{\text{best}}$ **then**
    $\mathcal{E} = \mathcal{E} \cup \{f_{\theta^\star}\}$, $\mathcal{S}_{\text{best}} = \mathcal{S}(\mathcal{E})$;
  else
    return $\mathcal{E}$;
  end
end
return $\mathcal{E}$;

Construction of hyper-deep ensembles

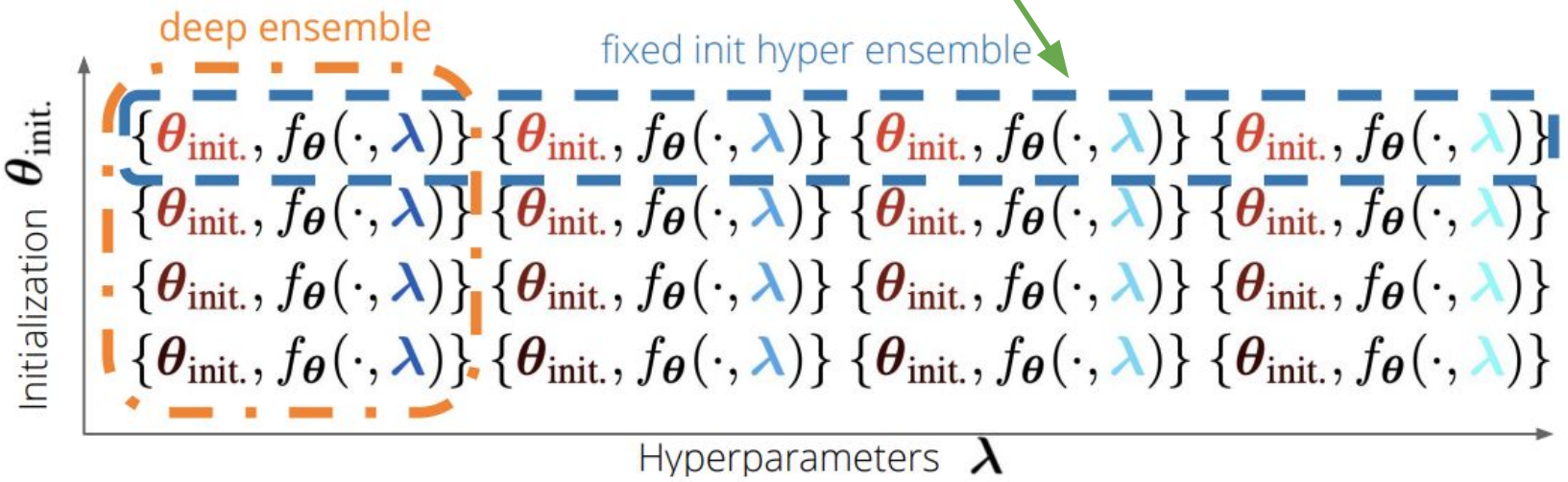
Google AI
Brain Team



Construction of hyper-deep ensembles

Google AI
Brain Team

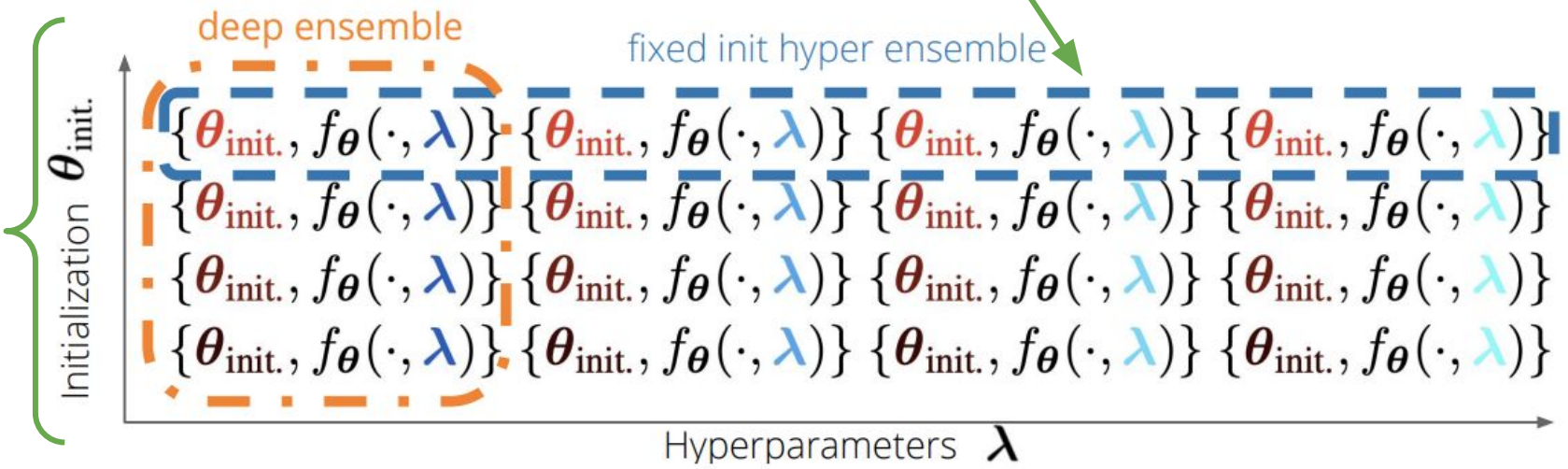
(A) Construct one “row” from random search + greedy



Construction of hyper-deep ensembles

Google AI
Brain Team

(A) Construct one “row” from random search + greedy

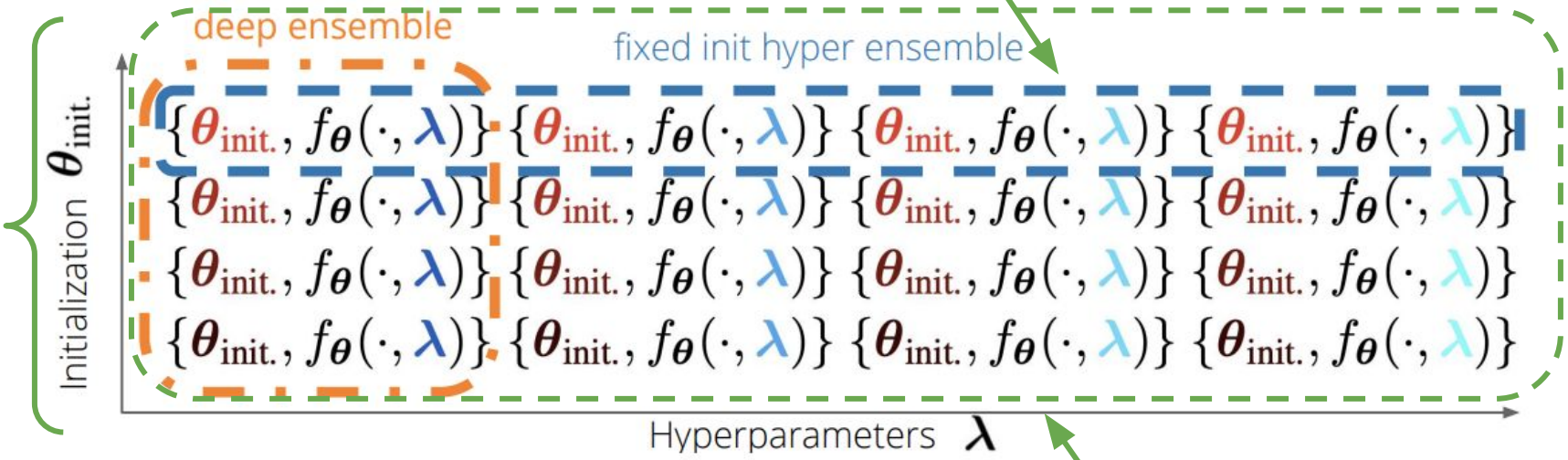


(B) Tile “row” for different initializations

Construction of hyper-deep ensembles

Google AI
Brain Team

(A) Construct one “row” from random search + greedy



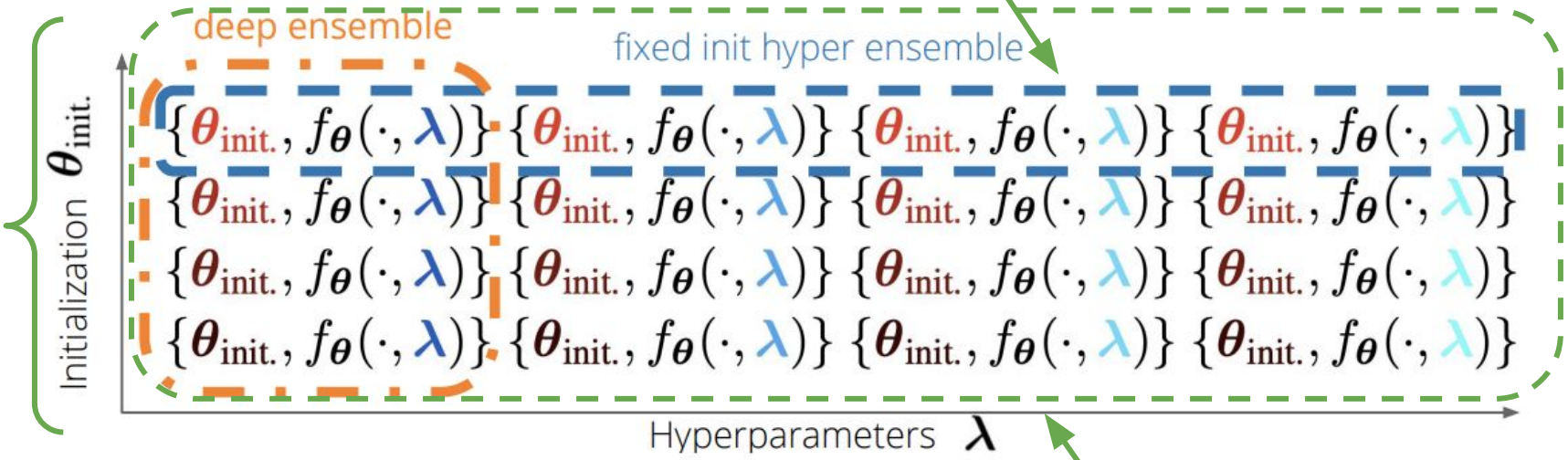
(B) Tile “row” for different initializations

(C) Extract final ensemble with greedy

Construction of hyper-deep ensembles

Google AI
Brain Team

(A) Construct one “row” from random search + greedy



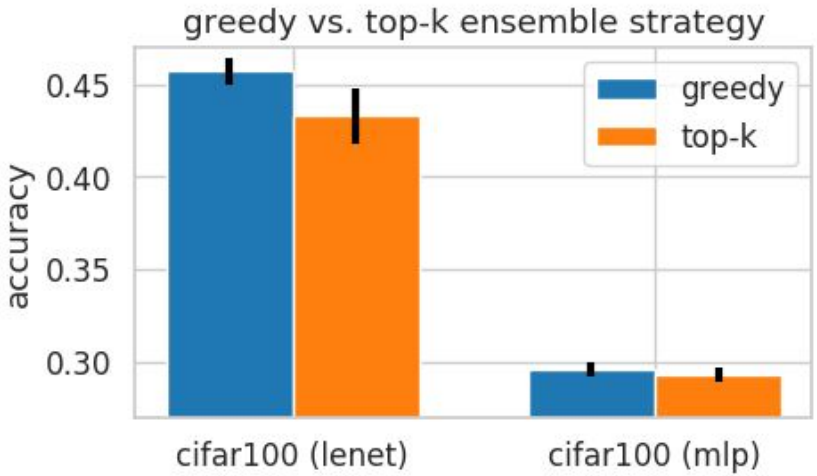
(B) Tile “row” for different initializations

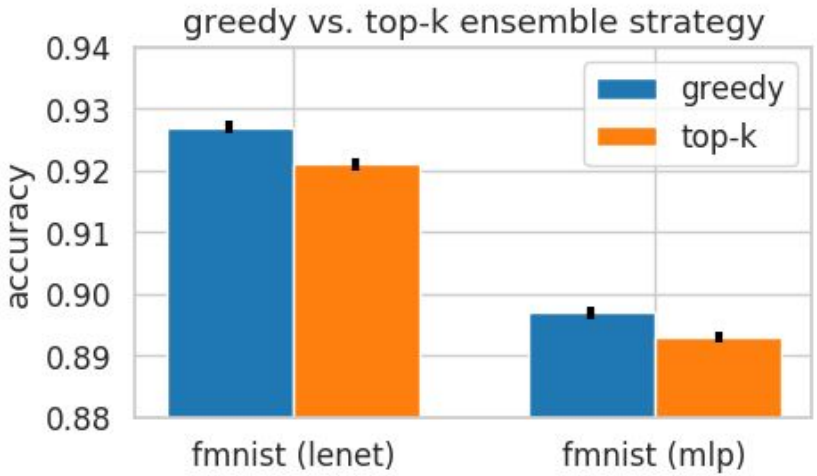
(C) Extract final ensemble with greedy

Concurrent work with similar construction [Zaidi+Zela et al., 2020]

Why not simply a top-K selection? [Saikia et al. 2020]

Google AI
Brain Team





Unlike greedy, top-K is not informed by the ensemble performance

Outline of the talk

Google AI
Brain Team

Hyper-deep ensemble

- From deep ensemble to hyper-deep ensemble
- Construction
- Evaluation

Hyper-batch ensemble

- Two key ingredients:
  - Batch ensemble
  - Self-tuning networks
- Layer structure & end-to-end ensemble training
- Evaluation

Settings

Google AI
Brain Team

Small-scale experiments:

- Hyperparameters: layer-wise L2, dropout
- MLP/LeNet over CIFAR-100, Fashion MNIST

Larg(er)-scale experiments:

- Hyperparameters: block-wise L2, label smoothing
- ResNet-20 and Wide ResNet-28-10 over CIFAR-10, CIFAR-100
- Initial random search with 100 models

Results CIFAR-10

Google AI Brain Team

		single (1)	deep ens (4)	hyper-deep ens (4)
cifar10	nll ↓	0.262 ± 0.006	**0.172** ± 0.003	**0.173** ± 0.003
	acc ↑	0.927 ± 0.001	0.946 ± 0.000	**0.947** ± 0.000
	ece ↓	0.035 ± 0.001	**0.011** ± 0.001	**0.011** ± 0.001

ResNet 20

		single (1)	deep ens (4)	hyper-deep ens (4)
cifar10	nll ↓	0.152 ± 0.009	**0.108** ± 0.005	**0.108** ± 0.004
	acc ↑	0.961 ± 0.001	0.968 ± 0.000	**0.969** ± 0.000
	ece ↓	0.023 ± 0.005	**0.007** ± 0.003	**0.008** ± 0.002

W. ResNet 28-10

Results CIFAR-10

Google AI Brain Team

		single (1)	deep ens (4)	hyper-deep ens (4)
cifar10	nll ↓	0.262 ± 0.006	**0.172** ± 0.003	**0.173** ± 0.003
	acc ↑	0.927 ± 0.001	0.946 ± 0.000	**0.947** ± 0.000
	ece ↓	0.035 ± 0.001	**0.011** ± 0.001	**0.011** ± 0.001
	div ↑	–	1.398 ± 0.005	**1.446** ± 0.004

ResNet 20

Diversity [Fort et al. 2019]

$$\frac{\text{mean prediction disagreement}}{\text{1 - accuracy}}$$

		single (1)	deep ens (4)	hyper-deep ens (4)
cifar10	nll ↓	0.152 ± 0.009	**0.108** ± 0.005	**0.108** ± 0.004
	acc ↑	0.961 ± 0.001	0.968 ± 0.000	**0.969** ± 0.000
	ece ↓	0.023 ± 0.005	**0.007** ± 0.003	**0.008** ± 0.002
	div ↑	–	0.982 ± 0.002	**1.087** ± 0.002

W. ResNet 28-10

Results CIFAR-100

Google AI
Brain Team

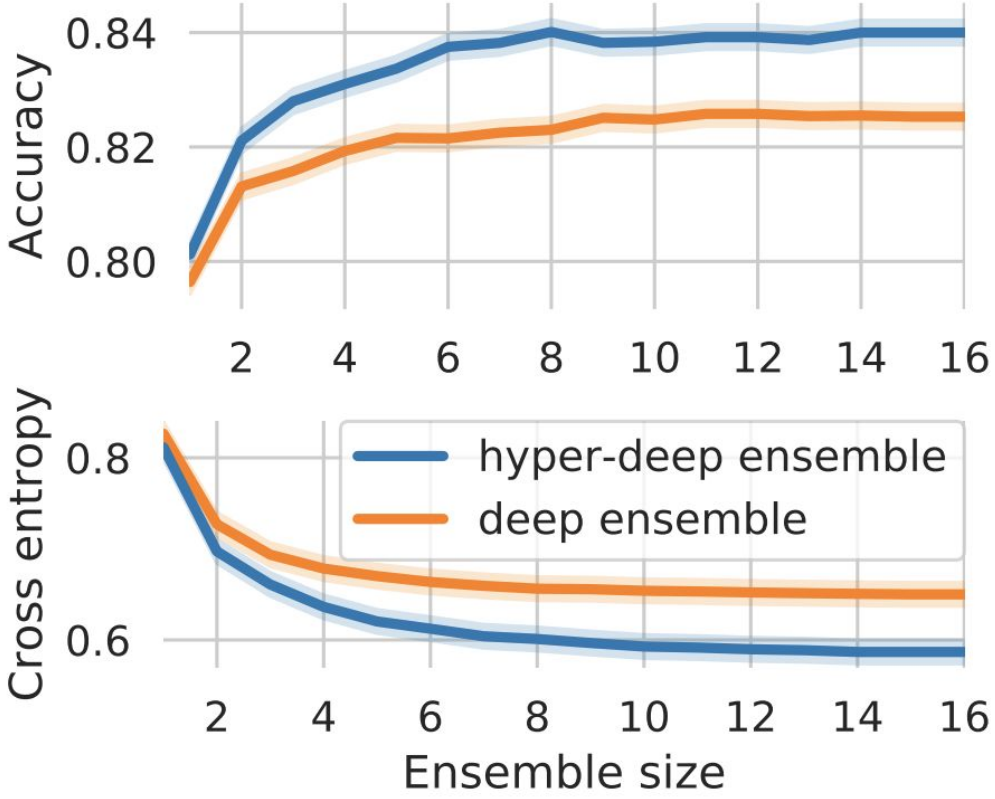
		single (1)	deep ens (4)	hyper-deep ens (4)
cifar100	nll ↓	1.178 ± 0.020	0.908 ± 0.003	**0.896** ± 0.003
	acc ↑	0.682 ± 0.005	0.751 ± 0.002	**0.754** ± 0.002
	ece ↓	0.064 ± 0.005	0.070 ± 0.005	**0.050** ± 0.004
	div ↑	–	**1.332** ± 0.051	**1.356** ± 0.049

ResNet 20

		single (1)	deep ens (4)	hyper-deep ens (4)
cifar100	nll ↓	0.811 ± 0.026	0.678 ± 0.013	**0.636** ± 0.013
	acc ↑	0.801 ± 0.004	0.819 ± 0.001	**0.831** ± 0.001
	ece ↓	0.062 ± 0.001	**0.021** ± 0.002	**0.021** ± 0.002
	div ↑	–	0.954 ± 0.002	**1.142** ± 0.001

W. ResNet 28-10

Results CIFAR-100



Overall cost comparison

Google AI
Brain Team

	Deep ensembles	Hyper-deep ensembles
Training cost	$O(K)$	$O(K^2) + \text{Cost}_{\text{random search}}$
Prediction cost	Same	
Memory cost	Same	

Overall cost comparison

Google AI
Brain Team

	Deep ensembles	Hyper-deep ensembles
Training cost	$O(K)$	$O(K^2) + \text{Cost}_{\text{random search}}$
Prediction cost	Same	
Memory cost	Same	
“Tuning cost”	$\text{Cost'}_{\text{random search}}$	0

Overall cost comparison

Google AI
Brain Team

	Deep ensembles	Hyper-deep ensembles
Training cost	$O(K)$	$O(K^2) + \text{Cost}_{\text{random search}}$
Prediction cost	Same	
Memory cost	Same	
“Tuning cost”	$\text{Cost'}_{\text{random search}}$	0
Total cost	$\approx$	

Outline of the talk

Google AI
Brain Team

Hyper-deep ensemble

- From deep ensemble to hyper-deep ensemble
- Construction
- Evaluation

Hyper-batch ensemble

- Two key ingredients:
  - Batch ensemble
  - Self-tuning networks
- Layer structure & end-to-end ensemble training
- Evaluation

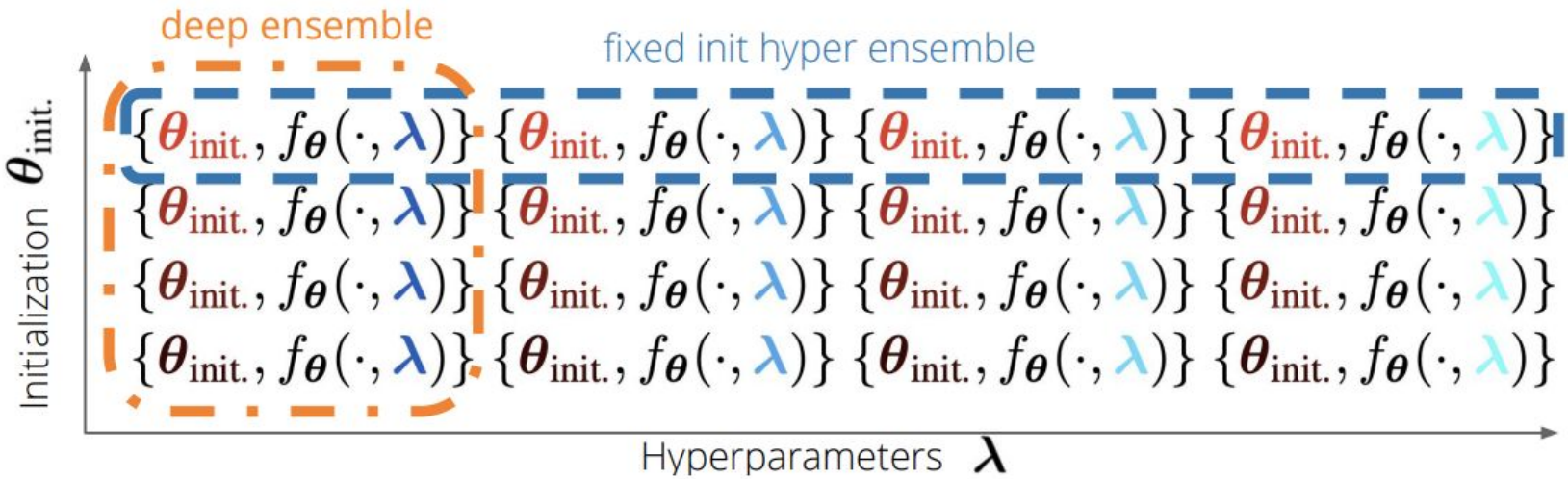
Limitations of (hyper-)deep ensembles

Google AI
Brain Team

For (hyper-)deep ensembles:

- Training scales linearly with number of members
- Storage scales linearly with number of members
- Prediction scales linearly with number of members

Limitations of (hyper-)deep ensembles

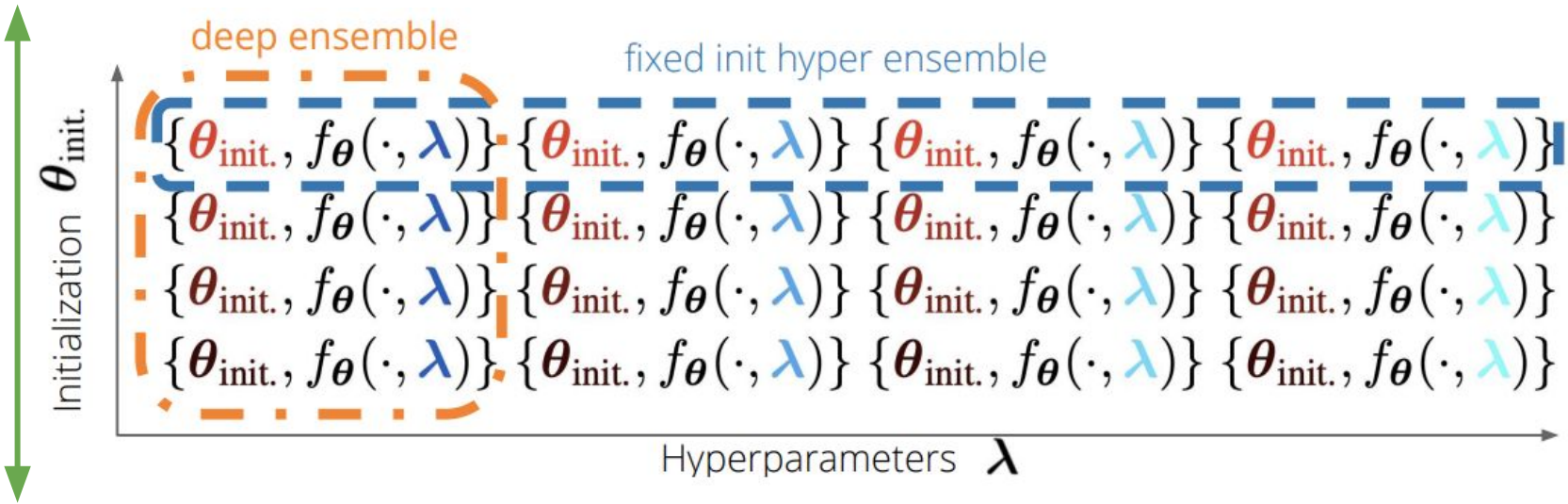


Limitations of (hyper-)deep ensembles

Google AI
Brain Team

Efficient extensions of deep ensembles:

- Batch ensemble [Wen et al., 2019]
- rank-1 BNN [Dusenberry et al., 2020]

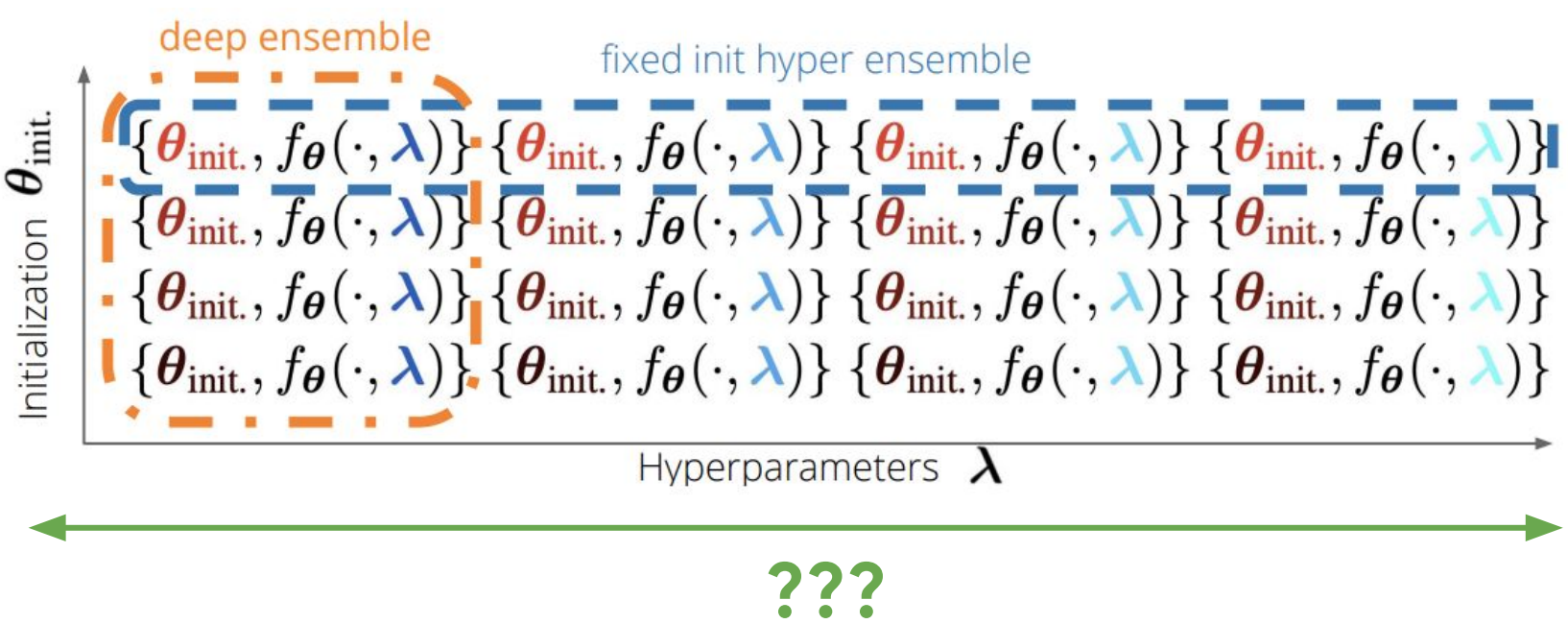


Limitations of (hyper-)deep ensembles

Google AI
Brain Team

Efficient extensions of deep ensembles:

- Batch ensemble [Wen et al., 2019]
- rank-1 BNN [Dusenberry et al., 2020]
- • •



Our contribution (part 2)

Google AI
Brain Team

- Efficient ensemble defined over different hyperparameters
- Introduce layer structure composing
  - Batch ensemble
  - Self-tuning network
- Ensemble members **&** their HPs are learned **end-to-end** in a **single** training

Goal:

- Exploit insights of hyper-deep ensemble...
- ...But make it more efficient!

Outline of the talk

Google AI
Brain Team

Hyper-deep ensemble

- From deep ensemble to hyper-deep ensemble
- Construction
- Evaluation

Hyper-batch ensemble

- Two key ingredients:
  - Batch ensemble
  - Self-tuning networks
- Layer structure & end-to-end ensemble training
- Evaluation

1-slide summary about batch ensemble

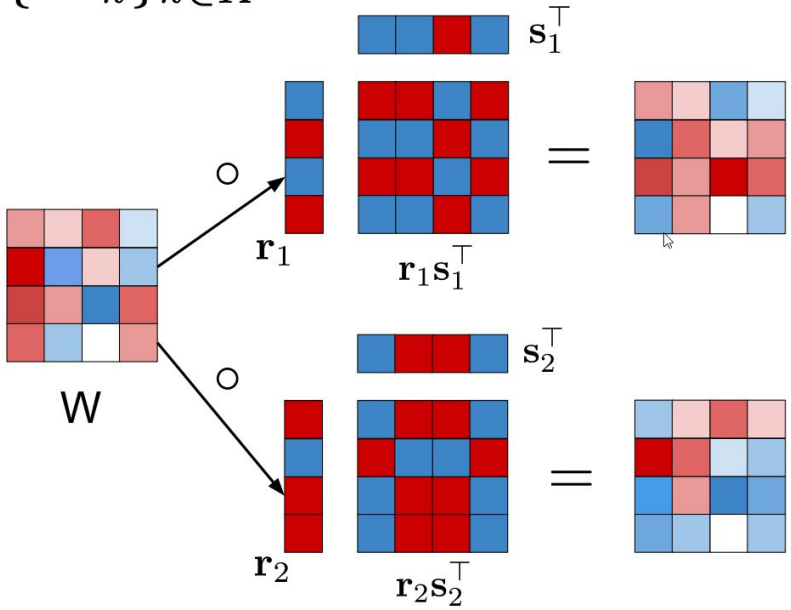
Google AI
Brain Team

Batch ensemble [Wen et al., 2019]:

- Tie parameters across members: Instead of $\{\mathbf{W}_k\}_{k \in K}$

$$\mathbf{W} \circ (\mathbf{r}_k \mathbf{s}_k^\top) \text{ for each } k \in \{1, \ldots, K\}$$

shared per member



1-slide summary about batch ensemble

Google AI
Brain Team

Batch ensemble [Wen et al., 2019]:

- Tie parameters across members: Instead of $\{\mathbf{W}_k\}_{k \in K}$

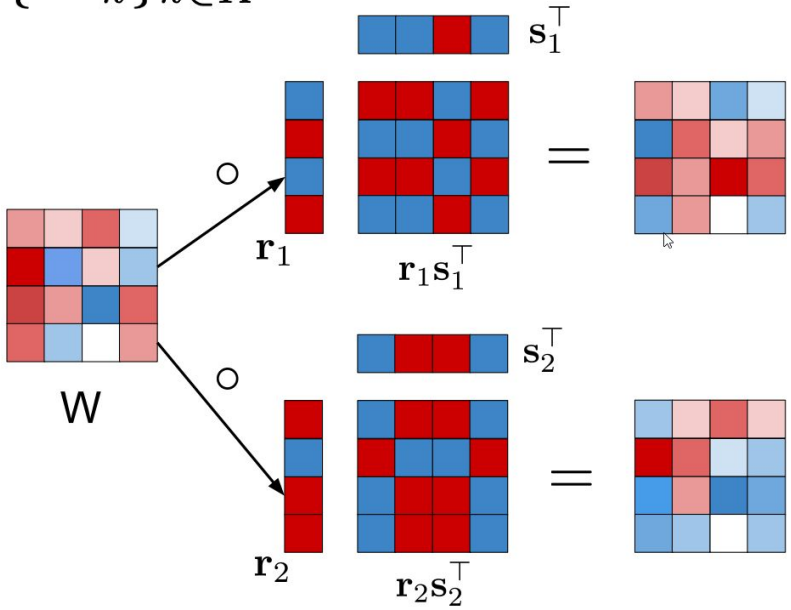
$$\mathbf{W} \circ (\mathbf{r}_k \mathbf{s}_k^\top) \text{ for each } k \in \{1, \ldots, K\}$$

shared per member

- Save parameters
- Fast prediction thanks to vectorization:

$$\mathbf{X}[\mathbf{W} \circ (\mathbf{r}_k \mathbf{s}_k^\top)] = [(\mathbf{X} \circ \mathbf{r}_k^\top)\mathbf{W}] \circ \mathbf{s}_k^\top$$

- (similar for conv. layers)



Outline of the talk

Google AI
Brain Team

Hyper-deep ensemble

- From deep ensemble to hyper-deep ensemble
- Construction
- Evaluation

Hyper-batch ensemble

- Two key ingredients:
  - Batch ensemble
  - Self-tuning networks
- Layer structure & end-to-end ensemble training
- Evaluation

2-slide summary about self-tuning networks [MacKay et al. 2019]

Google AI Brain Team

Main idea:

$$\min_{\boldsymbol{\theta}} \left\{ \frac{1}{2} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 + \lambda_1 \|\boldsymbol{\theta}\|^2 \right\} \quad \min_{\boldsymbol{\theta}} \left\{ \frac{1}{2} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 + \lambda_2 \|\boldsymbol{\theta}\|^2 \right\}$$

solution

$$\boldsymbol{\theta}^\star(\lambda_1)$$

solution

$$\boldsymbol{\theta}^\star(\lambda_2)$$

- Can you track how the parameters of your model change w.r.t. HPs?

2-slide summary about self-tuning networks [MacKay et al. 2019]

Google AI Brain Team

Main idea:

$$\min_{\boldsymbol{\theta}} \left\{ \frac{1}{2} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 + \lambda_1 \|\boldsymbol{\theta}\|^2 \right\} \quad \min_{\boldsymbol{\theta}} \left\{ \frac{1}{2} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 + \lambda_2 \|\boldsymbol{\theta}\|^2 \right\}$$

solution solution

$$\boldsymbol{\theta}^\star(\lambda_1) \qquad \boldsymbol{\theta}^\star(\lambda_2)$$

- Can you track how the parameters of your model change w.r.t. HPs?
- Assume you can approximate the mapping from HPs to the solution

$$\boldsymbol{\theta}^\star(\lambda) \approx \boldsymbol{\theta}^{\text{approx}}_{\boldsymbol{\beta}}(\lambda)$$

2-slide summary about self-tuning networks [MacKay et al. 2019]

Main idea:

$$\min_{\boldsymbol{\theta}} \left\{ \frac{1}{2} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 + \lambda_1 \|\boldsymbol{\theta}\|^2 \right\} \quad \min_{\boldsymbol{\theta}} \left\{ \frac{1}{2} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 + \lambda_2 \|\boldsymbol{\theta}\|^2 \right\}$$

solution $\boldsymbol{\theta}^\star(\lambda_1)$ solution $\boldsymbol{\theta}^\star(\lambda_2)$

- Can you track how the parameters of your model change w.r.t. HPs?
- Assume you can approximate the mapping from HPs to the solution

$$\min_{\boldsymbol{\beta}} \mathbb{E}_{\lambda \sim p(\lambda)} \left\{ \frac{1}{2} \|\mathbf{X}\boldsymbol{\theta}^{\text{approx}}_{\boldsymbol{\beta}}(\lambda) - \mathbf{y}\|^2 + \lambda \|\boldsymbol{\theta}^{\text{approx}}_{\boldsymbol{\beta}}(\lambda)\|^2 \right\} \text{ with } \boldsymbol{\theta}^\star(\lambda) \approx \boldsymbol{\theta}^{\text{approx}}_{\boldsymbol{\beta}}(\lambda)$$

(to be defined)

2-slide summary about self-tuning networks [MacKay et al. 2019]

Main idea:

$$\min_{\boldsymbol{\theta}} \left\{ \frac{1}{2} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 + \lambda_1 \|\boldsymbol{\theta}\|^2 \right\} \quad \min_{\boldsymbol{\theta}} \left\{ \frac{1}{2} \|\mathbf{X}\boldsymbol{\theta} - \mathbf{y}\|^2 + \lambda_2 \|\boldsymbol{\theta}\|^2 \right\}$$

solution $\boldsymbol{\theta}^\star(\lambda_1)$ solution $\boldsymbol{\theta}^\star(\lambda_2)$

- Can you track how the parameters of your model change w.r.t. HPs?
- Assume you can approximate the mapping from HPs to the solution

$$\min_{\boldsymbol{\beta}} \mathbb{E}_{\lambda \sim p(\lambda)} \left\{ \frac{1}{2} \|\mathbf{X}\boldsymbol{\theta}^{\text{approx}}_{\boldsymbol{\beta}}(\lambda) - \mathbf{y}\|^2 + \lambda \|\boldsymbol{\theta}^{\text{approx}}_{\boldsymbol{\beta}}(\lambda)\|^2 \right\} \text{ with } \boldsymbol{\theta}^\star(\lambda) \approx \boldsymbol{\theta}^{\text{approx}}_{\boldsymbol{\beta}}(\lambda)$$

(to be defined)

- With a **single** training, we could recover the **entire set of solutions**!

2-slide summary about self-tuning networks [MacKay et al. 2019]

Google AI Brain Team

Inner workings of self-tuning networks:

- Changes captured **layer-wise**, by shifting & scaling the units:

$$\mathbf{W}(\boldsymbol{\lambda}) = \boxed{\mathbf{W}} + \Delta \circ \mathbf{e}(\boldsymbol{\lambda})^\top \quad \text{and} \quad \mathbf{b}(\boldsymbol{\lambda}) = \boxed{\mathbf{b}} + \boldsymbol{\delta} \circ \mathbf{e}'(\boldsymbol{\lambda})$$

Standard parameters

2-slide summary about self-tuning networks [MacKay et al. 2019]

Inner workings of self-tuning networks:

- Changes captured **layer-wise**, by shifting & scaling the units:

$$\mathbf{W}(\boldsymbol{\lambda}) = \mathbf{W} + \Delta \circ \mathbf{e}(\boldsymbol{\lambda})^\top \quad \text{and} \quad \mathbf{b}(\boldsymbol{\lambda}) = \mathbf{b} + \boldsymbol{\delta} \circ \mathbf{e}'(\boldsymbol{\lambda})$$

Standard parameters

Embedding of HPs

2-slide summary about self-tuning networks [MacKay et al. 2019]

Inner workings of self-tuning networks:

- Changes captured **layer-wise**, by shifting & scaling the units:

$$\mathbf{W}(\boldsymbol{\lambda}) = \mathbf{W} + \Delta \circ \mathbf{e}(\boldsymbol{\lambda})^\top \quad \text{and} \quad \mathbf{b}(\boldsymbol{\lambda}) = \mathbf{b} + \boldsymbol{\delta} \circ \mathbf{e}'(\boldsymbol{\lambda})$$

Standard parameters

Embedding of HPs

- Alternating optimization:

$$\min_{\boldsymbol{\Theta}} \mathbb{E}_{\boldsymbol{\lambda} \sim p(\boldsymbol{\lambda}), (\mathbf{x}, y) \in \mathcal{D}} \left[\mathcal{L}(\mathbf{x}, y, \boldsymbol{\Theta}, \boldsymbol{\lambda}) \right]$$

2-slide summary about self-tuning networks [MacKay et al. 2019]

Inner workings of self-tuning networks:

- Changes captured **layer-wise**, by shifting & scaling the units:

$$\mathbf{W}(\boldsymbol{\lambda}) = \mathbf{W} + \Delta \circ \mathbf{e}(\boldsymbol{\lambda})^\top \quad \text{and} \quad \mathbf{b}(\boldsymbol{\lambda}) = \mathbf{b} + \boldsymbol{\delta} \circ \mathbf{e}'(\boldsymbol{\lambda})$$

Standard parameters

Embedding of HPs

- Alternating optimization:

$$\min_{\boldsymbol{\Theta}} \mathbb{E}_{\boldsymbol{\lambda} \sim p(\boldsymbol{\lambda}|\boldsymbol{\xi}_t), (\mathbf{x}, y) \in \mathcal{D}} \left[\mathcal{L}(\mathbf{x}, y, \boldsymbol{\Theta}, \boldsymbol{\lambda}) \right]$$

  - **Training step**
  - Update $\boldsymbol{\Theta}$
  - Keep $\boldsymbol{\xi}_t$ fixed

2-slide summary about self-tuning networks [MacKay et al. 2019]

Google AI Brain Team

Inner workings of self-tuning networks:

- Changes captured **layer-wise**, by shifting & scaling the units:

$$\mathbf{W}(\boldsymbol{\lambda}) = \mathbf{W} + \Delta \circ \mathbf{e}(\boldsymbol{\lambda})^\top \quad \text{and} \quad \mathbf{b}(\boldsymbol{\lambda}) = \mathbf{b} + \boldsymbol{\delta} \circ \mathbf{e}'(\boldsymbol{\lambda})$$

Standard parameters

Embedding of HPs

- Alternating optimization:

$$\min_{\boldsymbol{\Theta}} \mathbb{E}_{\boldsymbol{\lambda} \sim p(\boldsymbol{\lambda}|\boldsymbol{\xi}_t), (\mathbf{x}, y) \in \mathcal{D}} \left[\mathcal{L}(\mathbf{x}, y, \boldsymbol{\Theta}, \boldsymbol{\lambda}) \right]$$

- **Training step**
- Update $\boldsymbol{\Theta}$
- Keep $\boldsymbol{\xi}_t$ fixed

$$\min_{\boldsymbol{\xi}_t} \mathbb{E}_{\boldsymbol{\lambda} \sim p(\boldsymbol{\lambda}|\boldsymbol{\xi}_t), (\mathbf{x}, y) \in \mathcal{D}_{\text{val}}} \left[\ell_{\text{val}}(f_{\boldsymbol{\Theta}}(\mathbf{x}, \boldsymbol{\lambda}), y) \right]$$

- **Validation step**
- Keep $\boldsymbol{\Theta}$ fixed
- Update $\boldsymbol{\xi}_t$

Outline of the talk

Google AI
Brain Team

Hyper-deep ensemble

- From deep ensemble to hyper-deep ensemble
- Construction
- Evaluation

Hyper-batch ensemble

- Two key ingredients:
  - Batch ensemble
  - Self-tuning networks
- Layer structure & end-to-end ensemble training
- Evaluation

Hyper-batch ensemble (1/3)

Google AI
Brain Team

Composite layer structure:

- Batch ensemble

$$\mathbf{W} \circ (\mathbf{r}_k \mathbf{s}_k^\top) \text{ for each } k \in \{1, \ldots, K\}$$

Hyper-batch ensemble (1/3)

Google AI
Brain Team

Composite layer structure:

- Batch ensemble

$$\mathbf{W} \circ (\mathbf{r}_k \mathbf{s}_k^\top) \text{ for each } k \in \{1, \ldots, K\}$$

- Self-tuning network

$$\mathbf{W}(\boldsymbol{\lambda}) = \mathbf{W} + [\Delta] \circ \mathbf{e}(\boldsymbol{\lambda})^\top$$

Model vicinity of single HP

Hyper-batch ensemble (1/3)

Google AI
Brain Team

Composite layer structure:

- Batch ensemble

$$\mathbf{W} \circ (\mathbf{r}_k \mathbf{s}_k^\top) \text{ for each } k \in \{1, \ldots, K\}$$

- Self-tuning network

$$\mathbf{W}(\boldsymbol{\lambda}) = \mathbf{W} + [\Delta] \circ \mathbf{e}(\boldsymbol{\lambda})^\top$$

Model vicinity of single HP

- Hyper-batch ensemble

$$\mathbf{W}_k(\boldsymbol{\lambda}_k) = \mathbf{W} \circ (\mathbf{r}_k \mathbf{s}_k^\top) + [\Delta \circ (\mathbf{u}_k \mathbf{v}_k^\top)] \circ \mathbf{e}(\boldsymbol{\lambda}_k)^\top$$

Model vicinity of K HPs

→ Preserves memory compactness & efficient vectorization

Hyper-batch ensemble (2/3)

Google AI
Brain Team

One HP distribution per ensemble member:

$$\min_{\boldsymbol{\Theta}} \mathbb{E}_{\boldsymbol{\Lambda}_K \sim q_t, (\mathbf{x}, y) \in \mathcal{D}} \Big[\mathcal{L}(\mathbf{x}, y, \boldsymbol{\Theta}, \boldsymbol{\Lambda}_K) \Big] \quad \text{with} \begin{cases} \boldsymbol{\Lambda}_K = \{\boldsymbol{\lambda}_k\}_{k=1}^K \\ q_t\big(\boldsymbol{\Lambda}_K\big) = q(\boldsymbol{\Lambda}_K | \{\boldsymbol{\xi}_{k,t}\}_{k=1}^K) = \prod_{k=1}^K p(\boldsymbol{\lambda}_k | \boldsymbol{\xi}_{k,t}) \end{cases}$$

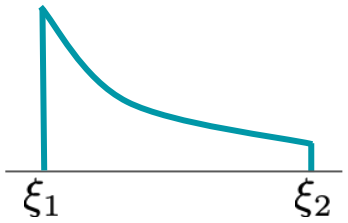
Hyper-batch ensemble (2/3)

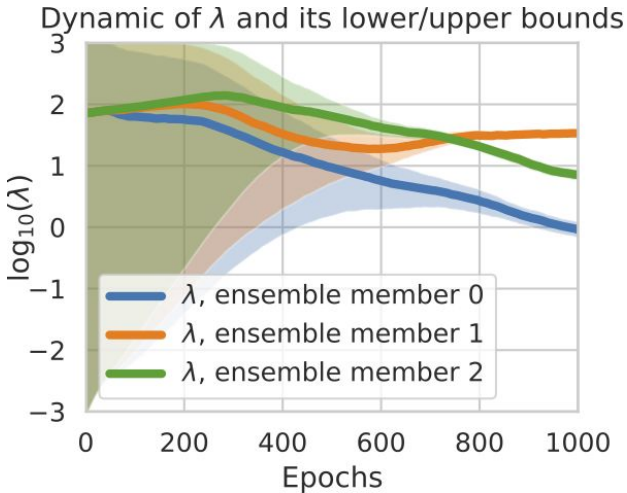
Google AI Brain Team

One HP distribution per ensemble member:

$$\min_{\Theta} \mathbb{E}_{\mathbf{\Lambda}_K \sim q_t, (\mathbf{x}, y) \in \mathcal{D}} \Big[\mathcal{L}(\mathbf{x}, y, \mathbf{\Theta}, \mathbf{\Lambda}_K) \Big] \quad \text{with} \begin{cases} \mathbf{\Lambda}_K = \{\boldsymbol{\lambda}_k\}_{k=1}^K \\ q_t\big(\mathbf{\Lambda}_K\big) = q(\mathbf{\Lambda}_K | \{\boldsymbol{\xi}_{k,t}\}_{k=1}^K) = \prod_{k=1}^K p(\boldsymbol{\lambda}_k | \boldsymbol{\xi}_{k,t}) \end{cases}$$

- Which choice for $p(\boldsymbol{\lambda}_k | \boldsymbol{\xi}_{k,t})$?
  - Log-uniform parametrized by bounds
  - Popular choice for positive HPs [Bergstra et al., 2011-12]





Hyper-batch ensemble (3/3)

Google AI
Brain Team

Ensemble predictions:

Individual member logits

Ensemble output

$$\begin{matrix} f_{\Theta_1}(\mathbf{x}, \lambda_1) \\ \vdots \\ f_{\Theta_K}(\mathbf{x}, \lambda_K) \end{matrix} \Bigg\} \; f_{\Theta}(\mathbf{x}, \lambda) = \frac{1}{K} \sum_k \texttt{softmax}\left(f_{\Theta_k}(\mathbf{x}, \lambda_k) \right)$$

$$\mathbf{W} \circ (\mathbf{r}_k \mathbf{s}_k^\top) + [\Delta \circ (\mathbf{u}_k \mathbf{v}_k^\top)] \circ \mathbf{e}(\boldsymbol{\lambda}_k)^\top$$

Outline of the talk

Google AI
Brain Team

Hyper-deep ensemble

- From deep ensemble to hyper-deep ensemble
- Construction
- Evaluation

Hyper-batch ensemble

- Two key ingredients:
  - Batch ensemble
  - Self-tuning networks
- Layer structure & end-to-end ensemble training
- Evaluation

Results CIFAR-10

Google AI
Brain Team

		batch ens (4)	hyper-batch ens (4)
cifar10	nll ↓	0.278 ± 0.004	**0.235** ± 0.004
	acc ↑	**0.929** ± 0.000	**0.929** ± 0.001
	ece ↓	0.039 ± 0.001	**0.017** ± 0.000
	div ↑	0.789 ± 0.010	**0.821** ± 0.013

ResNet 20

		batch ens (4)	hyper-batch ens (4)
cifar10	nll ↓	0.136 ± 0.001	**0.126** ± 0.001
	acc ↑	**0.963** ± 0.001	**0.963** ± 0.000
	ece ↓	0.017 ± 0.001	**0.009** ± 0.001
	div ↑	0.444 ± 0.003	**0.874** ± 0.026

W. ResNet 28-10

Results CIFAR-100

Google AI
Brain Team

		batch ens (4)	hyper-batch ens (4)
cifar100	nll ↓	1.235 ± 0.007	**1.152** ± 0.015
	acc ↑	0.697 ± 0.000	**0.699** ± 0.002
	ece ↓	0.119 ± 0.001	**0.095** ± 0.002
	div ↑	**0.154** ± 0.006	**0.159** ± 0.007

ResNet 20

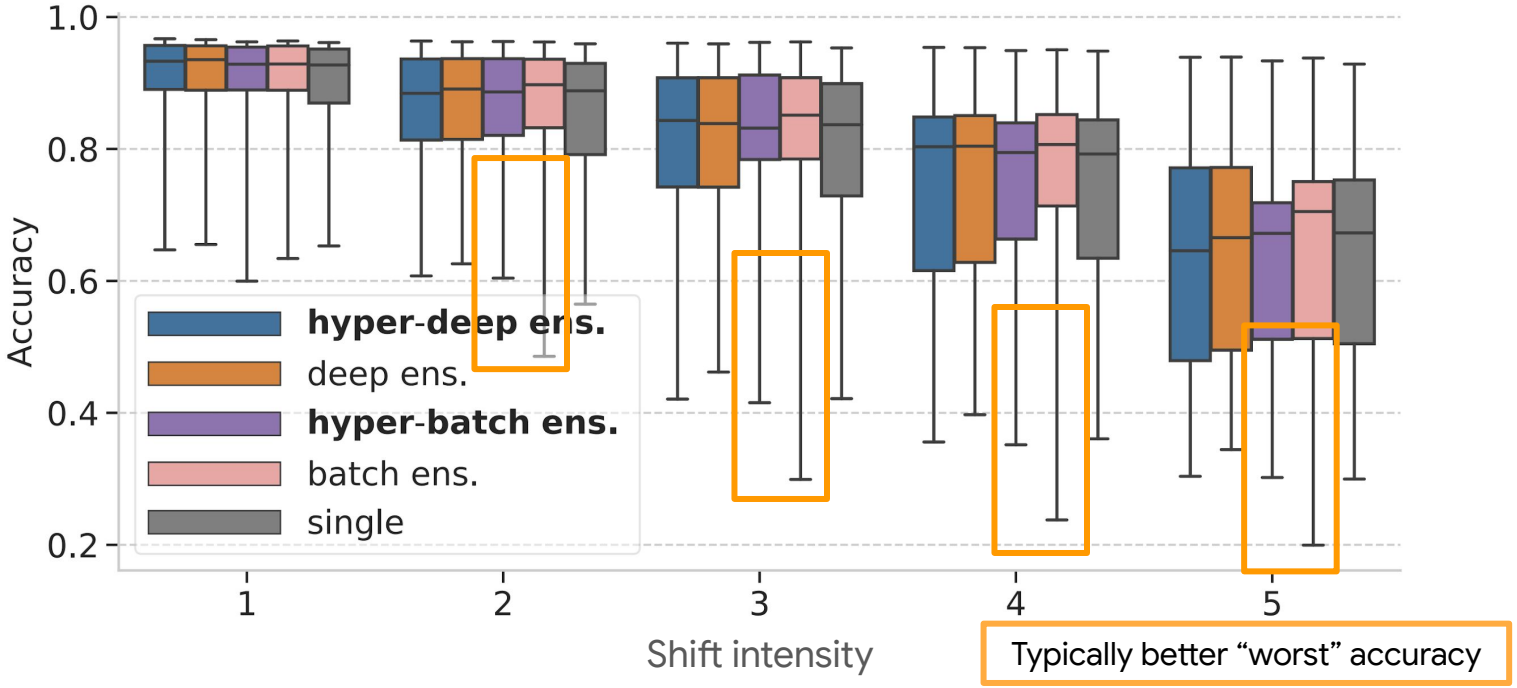
		batch ens (4)	hyper-batch ens (4)
cifar100	nll ↓	0.690 ± 0.005	**0.678** ± 0.005
	acc ↑	**0.819** ± 0.001	**0.820** ± 0.000
	ece ↓	0.026 ± 0.002	**0.022** ± 0.001
	div ↑	0.761 ± 0.014	**0.996** ± 0.015

W. ResNet 28-10

Results on corrupted data

Google AI
Brain Team

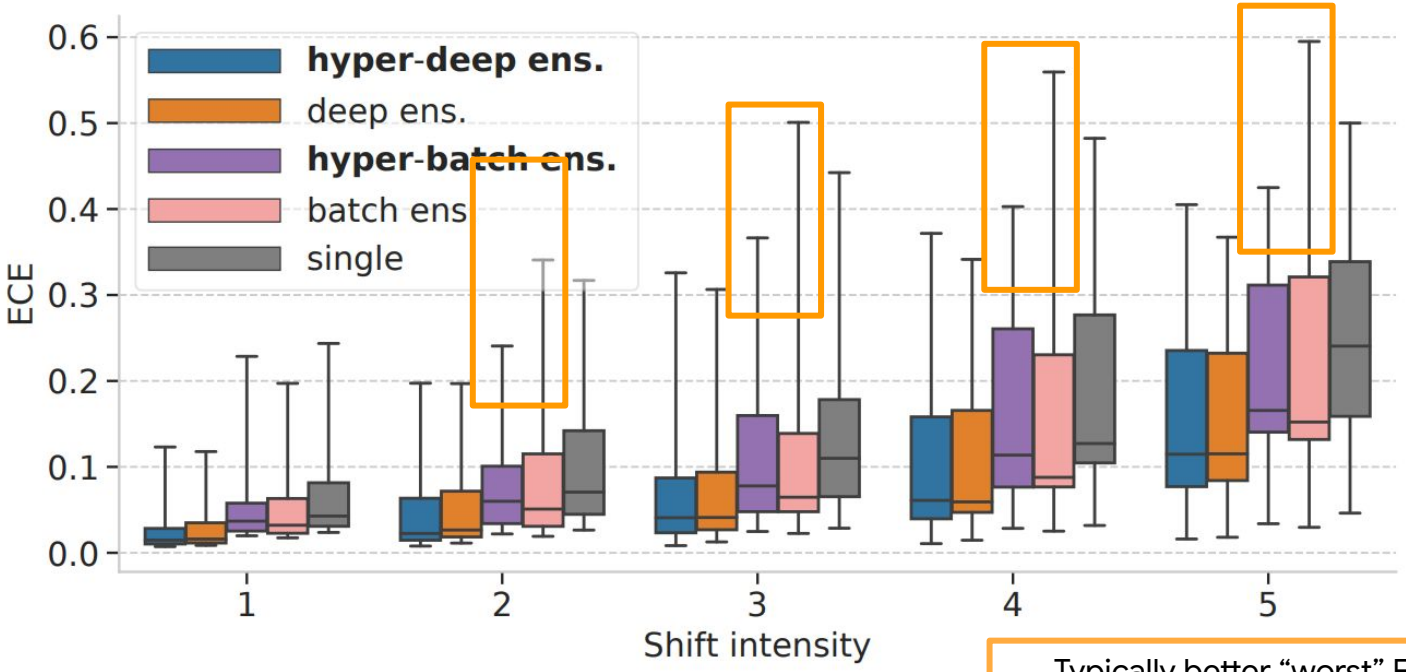
W. ResNet 28-10 on CIFAR-10 Corruptions [Hendrycks et al., 2019]



Results on corrupted data

Google AI
Brain Team

W. ResNet 28-10 on CIFAR-10 Corruptions [Hendrycks et al., 2019]



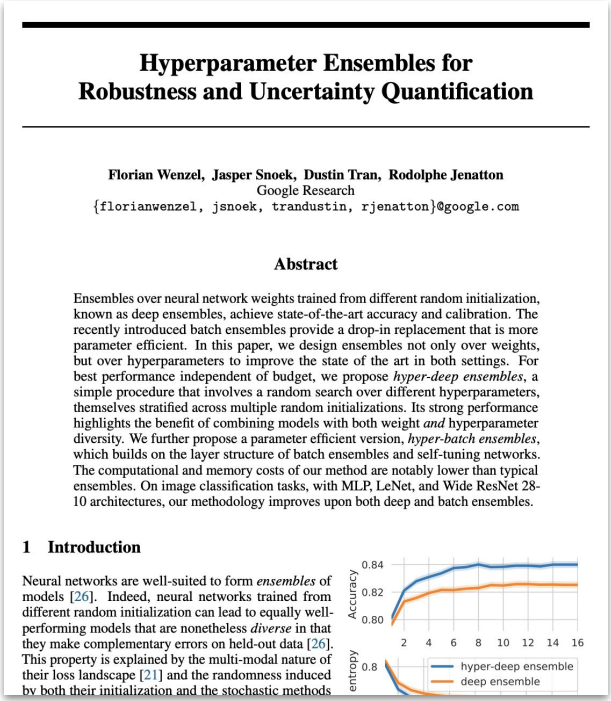
Typically better “worst” ECE

Thank you!

Google AI
Brain Team

- Beyond continuous hyperparameters?
- Make the approach more “turnkey”
- What other types of diversity matter?
- Diversity ⟷ performance [Masegosa, 2019]

- Open source code:
  - Experiments in uncertainty baselines
  - Generic layer code in edward2
- Paper on Arxiv



Hyperparameter Ensembles for Robustness and Uncertainty Quantification

Florian Wenzel, Jasper Snoek, Dustin Tran, Rodolphe Jenatton
Google Research
{florianwenzel, jsnoek, trandustin, rjenatton}@google.com

Abstract

Ensembles over neural network weights trained from different random initialization, known as deep ensembles, achieve state-of-the-art accuracy and calibration. The recently introduced batch ensembles provide a drop-in replacement that is more parameter efficient. In this paper, we design ensembles not only over weights, but over hyperparameters to improve the state of the art in both settings. For best performance independent of budget, we propose *hyper-deep ensembles*, a simple procedure that involves a random search over different hyperparameters, themselves stratified across multiple random initializations. Its strong performance highlights the benefit of combining models with both weight *and* hyperparameter diversity. We further propose a parameter efficient version, *hyper-batch ensembles*, which builds on the layer structure of batch ensembles and self-tuning networks. The computational and memory costs of our method are notably lower than typical ensembles. On image classification tasks, with MLP, LeNet, and Wide ResNet 28-10 architectures, our methodology improves upon both deep and batch ensembles.

1 Introduction

Neural networks are well-suited to form *ensembles* of models [26]. Indeed, neural networks trained from different random initialization can lead to equally well-performing models that are nonetheless *diverse* in that they make complementary errors on held-out data [26]. This property is explained by the multi-modal nature of their loss landscape [21] and the randomness induced by both their initialization and the stochastic methods

Additional results

Google AI
Brain Team

		hyper-batch ens (3)	hyper-batch ens (5)	batch ens (3)	batch ens (5)	STN (1)
cifar100 (mlp)	nll ↓	**2.979** ± 0.004	**2.983** ± 0.001	3.015 ± 0.003	3.056 ± 0.004	3.029 ± 0.006
	acc ↑	**0.281** ± 0.002	**0.282** ± 0.001	0.275 ± 0.001	0.265 ± 0.001	0.268 ± 0.002
	brier ↓	**-0.157** ± 0.000	**-0.157** ± 0.000	-0.153 ± 0.001	-0.141 ± 0.000	-0.145 ± 0.001
	ece ↓	0.030 ± 0.002	0.034 ± 0.001	**0.022** ± 0.002	0.033 ± 0.002	0.033 ± 0.004
cifar100 (lenet)	nll ↓	**2.283** ± 0.016	**2.297** ± 0.009	**2.350** ± 0.024	**2.239** ± 0.027	2.329 ± 0.017
	acc ↑	0.428 ± 0.003	0.425 ± 0.002	**0.438** ± 0.003	**0.437** ± 0.006	0.415 ± 0.003
	brier ↓	**-0.288** ± 0.003	-0.282 ± 0.002	**-0.295** ± 0.003	**-0.296** ± 0.008	-0.280 ± 0.002
	ece ↓	**0.058** ± 0.004	0.069 ± 0.006	**0.058** ± 0.015	**0.038** ± 0.018	**0.024** ± 0.007
fmnist (mlp)	nll ↓	0.308 ± 0.002	**0.304** ± 0.001	0.351 ± 0.004	0.320 ± 0.002	0.316 ± 0.003
	acc ↑	**0.892** ± 0.001	**0.892** ± 0.001	0.884 ± 0.001	**0.892** ± 0.000	**0.890** ± 0.001
	brier ↓	**-0.844** ± 0.001	**-0.845** ± 0.001	-0.830 ± 0.001	**-0.844** ± 0.001	-0.840 ± 0.002
	ece ↓	0.016 ± 0.001	**0.013** ± 0.001	0.020 ± 0.001	0.024 ± 0.001	**0.016** ± 0.001
fmnist (lenet)	nll ↓	**0.212** ± 0.001	**0.209** ± 0.002	0.230 ± 0.005	0.221 ± 0.002	0.224 ± 0.003
	acc ↑	**0.924** ± 0.001	**0.925** ± 0.001	0.920 ± 0.001	0.922 ± 0.001	0.920 ± 0.001
	brier ↓	**-0.889** ± 0.001	**-0.891** ± 0.001	-0.883 ± 0.001	-0.886 ± 0.001	-0.884 ± 0.001
	ece ↓	**0.009** ± 0.001	**0.008** ± 0.001	0.017 ± 0.002	0.015 ± 0.001	0.015 ± 0.001