Ensemble in deep neural network and Uncertainty quantification

Superb AI, Machine Learning Engineer

Kyeong Ryeol, Go

Basic concept

- **What is Ensemble**
  - Combine multiple hypothesis into one

- **Why does it work**
  - Suppose 5 independent classifiers for majority voting

    - The accuracy of majority voting
      - 0.8369 ≈ (0.7)^5 + 5 * 0.3 * (0.7)^4 + 10 * (0.3)^2 * (0.7)^3
  - Predictive variance
    - $Var((y_1 + y_2 + \cdots + y_K)/K) = \sigma^2/K$ where $Var(y_1) = \sigma^2$

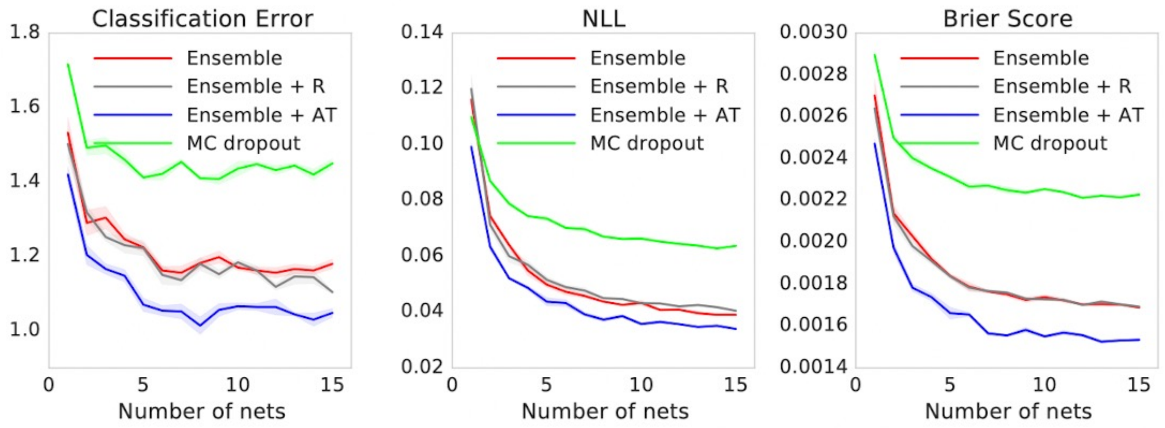
- **What matters**
  - Combine <u>accurate</u>(low bias) and <u>de-correlated</u>(low variance) solutions

Deep ensemble

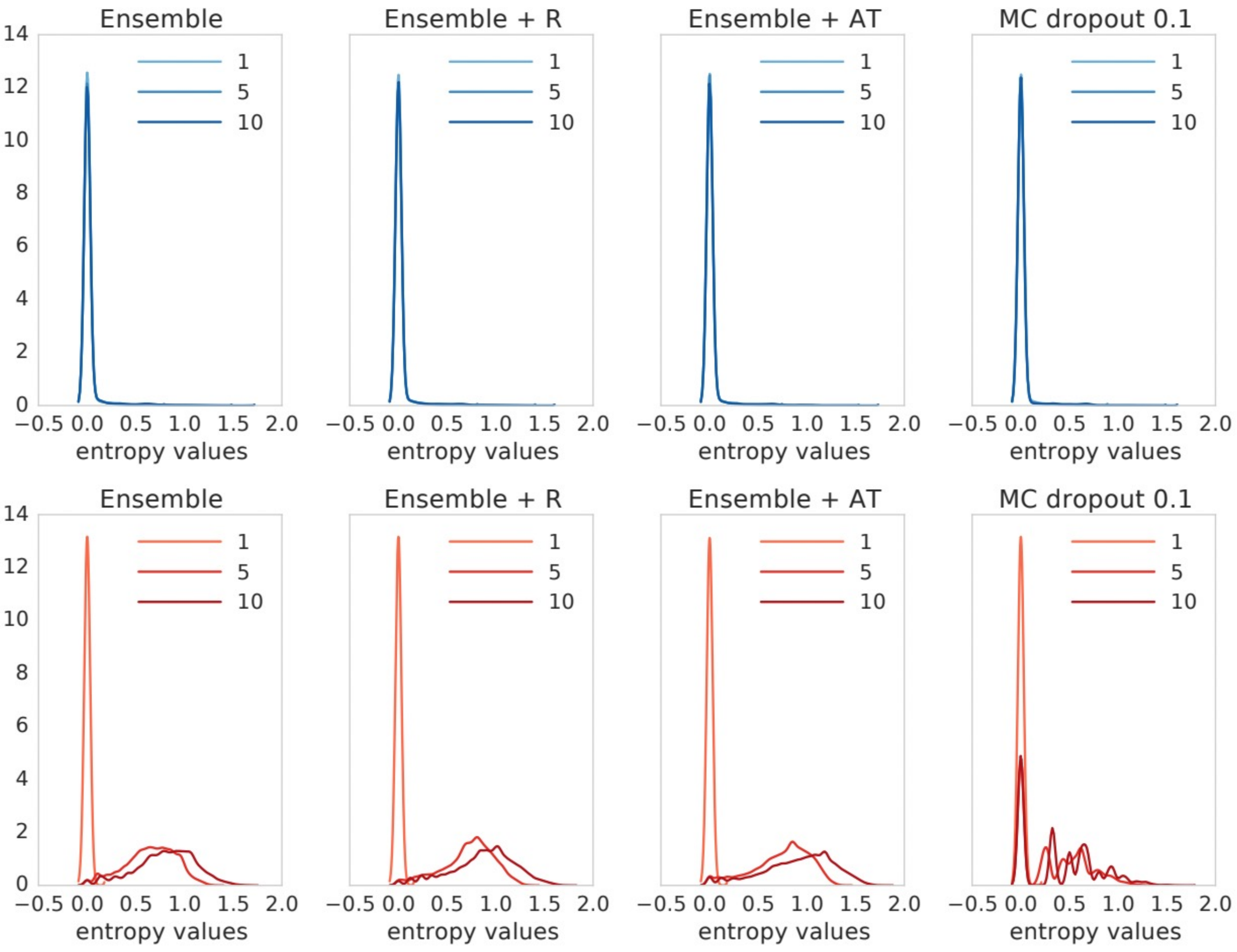
- **Core idea**
  - Average predictions of M(=5) randomly initialized NNs
    - Predictive probability : $\mathrm{p}(y|x) = M^{-1} \sum_m p_{\theta_m}(y|x, \theta_m)$

  - (Optional) Smooth the predictive distribution to improve the robustness on OOD
    - Adversarial example with ϵ(=0.01) : $x' = x + \epsilon \cdot sign\big(\nabla_x l(\theta, x, y)\big)$
    - Objective : $\min_{\theta_m} l(\theta_m, x, y) + l(\theta_m, x', y) \quad \forall m \in \{1, \ldots, M\}$

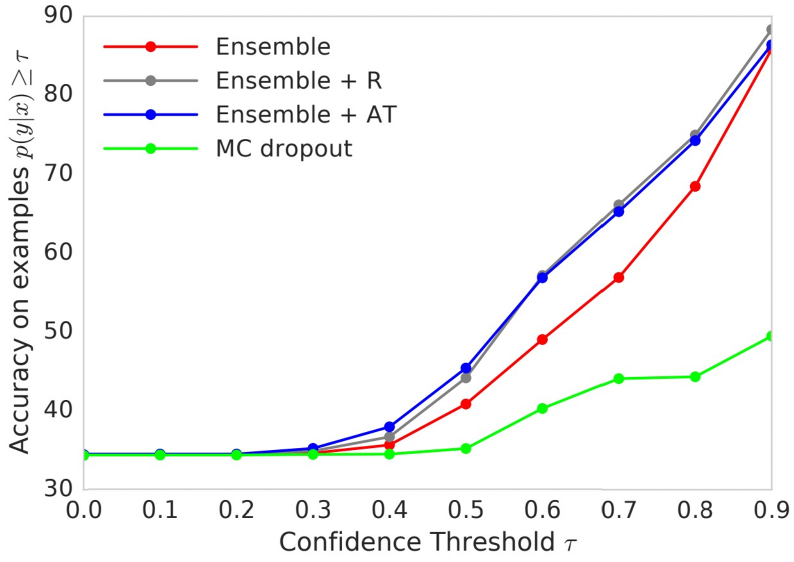
- **Experiment**
  - Compared to MC-dropout,
    - lower error, NLL, Brier score
    - Avoid overconfidence



Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles." *Advances in neural information processing systems* 30 (2017).



(**Top row**) known class, (**Bottom row**) unknown class
As ensemble size increases, overconfidence is relieved



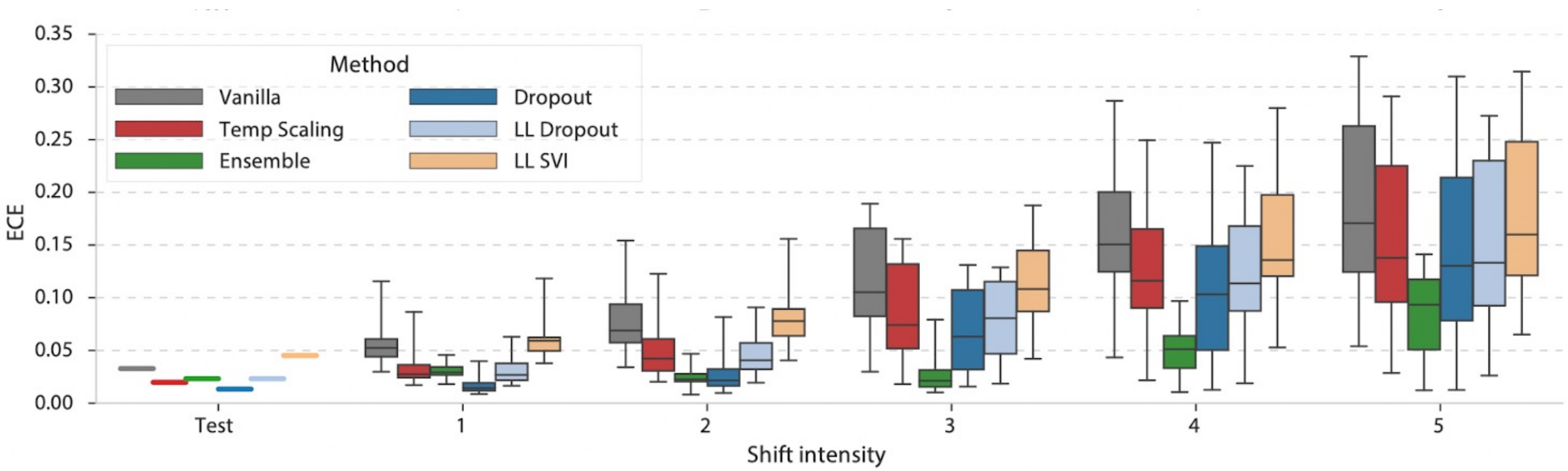
Confidence : $\max_k p(y = k|x)$
MC-dropout is overconfident to wrong prediction

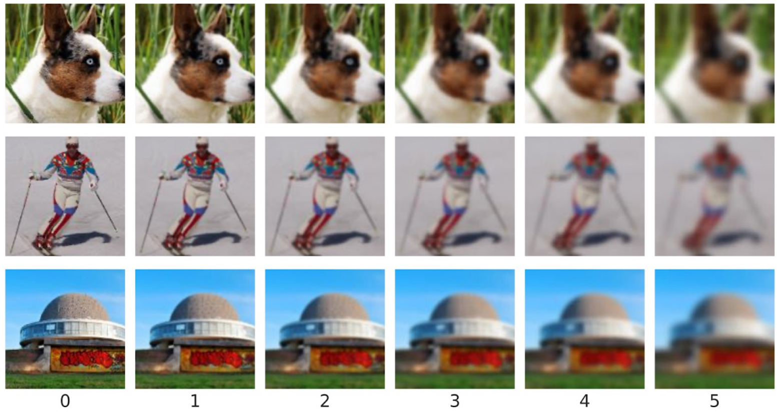
Better than others, too?

- Yes, both for accuracy and calibration under dataset shift
- Calibration metric : $ECE = \sum_{s=1}^{S} \frac{|B_s|}{N} |acc(B_s) - conf(B_s)|$
  - $\mathrm{acc}(B_s) = \frac{1}{|B_s|} \sum_{i \in \mathrm{B}_s} 1(\hat{y}_i = y_i)$
  - $\mathrm{conf}(\mathrm{B}_s) = \frac{1}{|B_s|} \sum_{i \in \mathrm{B}_s} p(\hat{y}_i | x_i)$

  where $\{\rho_s : s \in 1, \dots, S\}$ are percentiles of held-out predicted prob.

  $B_s = \{n \in 1, \dots, N : p(\hat{y}_n | x_n) \in (\rho_s, \rho_{s+1}]\}$

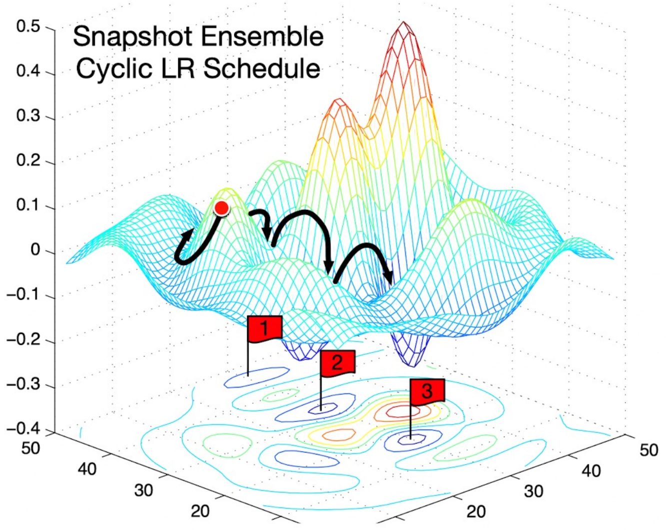




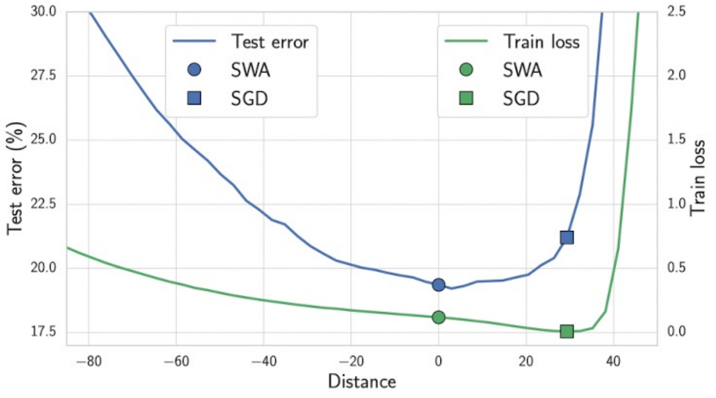
Shift w/ Gaussian blur

Ovadia, Yaniv, et al. "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift." *Advances in neural information processing systems* 32 (2019).

Scalability 1 : Loss landscape perspective



Cyclic learning rate



SGD (narrow) vs SWA (wide)

Instead of training $\underline{M \text{ models}}$, get $\underline{M \text{ checkpoints}}$

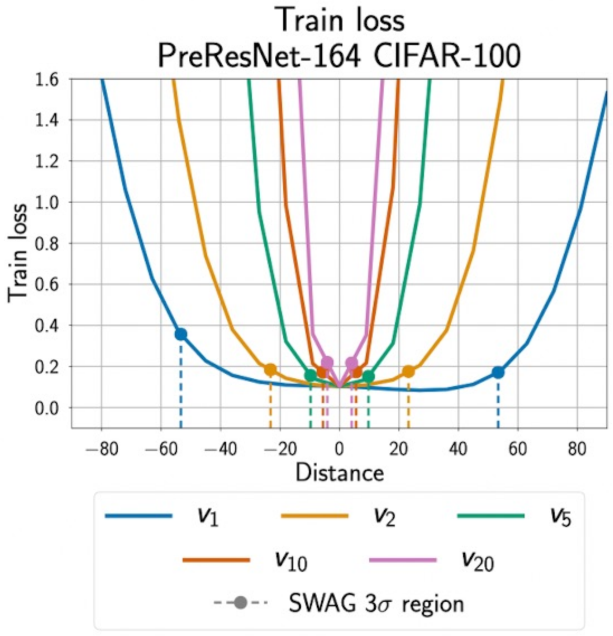
- **Snapshot Ensemble** (long interval)
  - Use cosine cyclic learning rate schedule
  - Repeat to i) converge to and ii) escape from local optima
- **Fast Geometric Ensemble** (pre-training + short interval)
  - Use linear cyclic learning rate schedule
  - Interpolate the optima with near-constant loss with small perturbation
- **Stochastic Weight Averaging** (average weights not predictions)
  - Use high constant or abrupt cyclic learning rate schedule
  - Require only a single forward process for the test-time prediction

Huang, Gao, et al. "Snapshot ensembles: Train 1, get m for free." *arXiv preprint arXiv:1704.00109* (2017).
Garipov, Timur, et al. "Loss surfaces, mode connectivity, and fast ensembling of dnns." *Advances in neural information processing systems* 31 (2018).
Izmailov, Pavel, et al. "Averaging weights leads to wider optima and better generalization." *arXiv preprint arXiv:1803.05407* (2018).

Estimating covariance

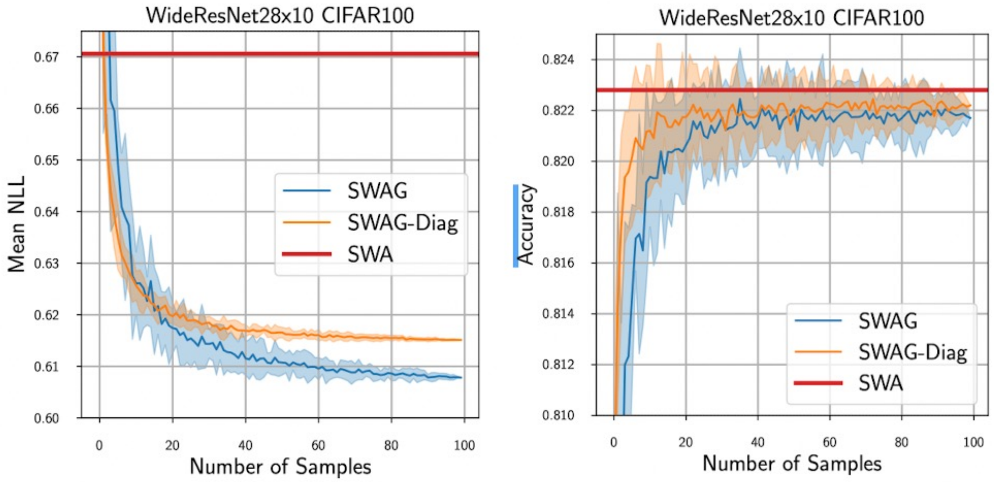
- Average predictions by multiple samples from $\mathcal{N}(\theta_{swa}, \Sigma)$
  - SWAG-Diagonal : $\Sigma = \Sigma_{\text{diag}} = diag\left(\overline{\theta^2} - \theta_{swa}^2\right)$
  - SWAG : $\Sigma = \frac{1}{2}\left(\Sigma_{diag} + \Sigma_{low-rank}\right)$

    where $\Sigma_{low-rank} = \frac{1}{K-1}\sum_{i=1}^{K}(\theta_i - \bar{\theta}_i)(\theta_i - \bar{\theta}_i)^T$

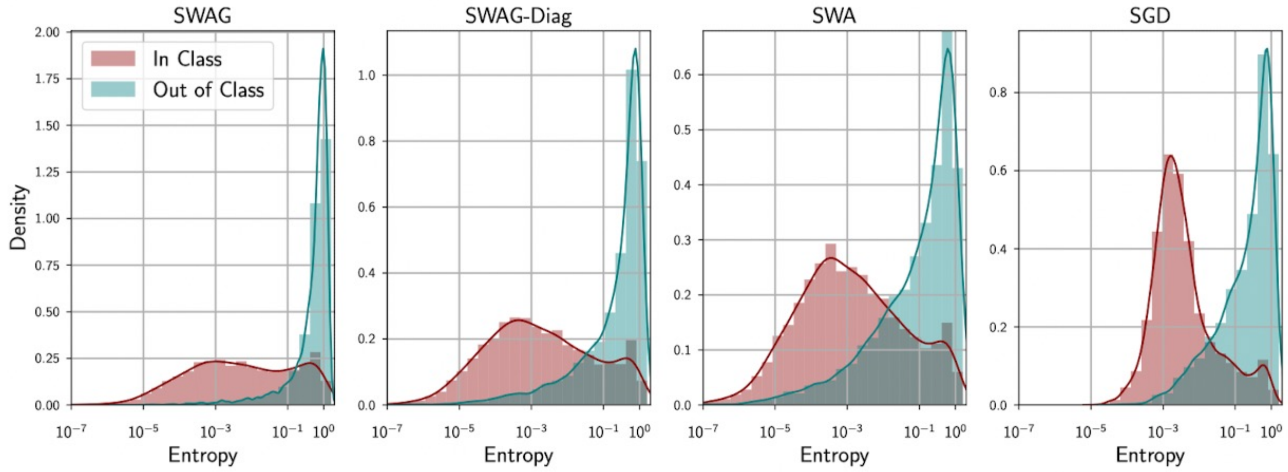


Loss along eigenvectors

- Deep ensemble can be viewed as Bayesian Model Averaging (BMA)
  - BMA : $p(y|x,D) = \int p(y|x,\theta)p(\theta|D)d\theta \; (\approx M^{-1}\sum_m p(y|x,\theta_m))$



Superior NLL, Inferior accuracy

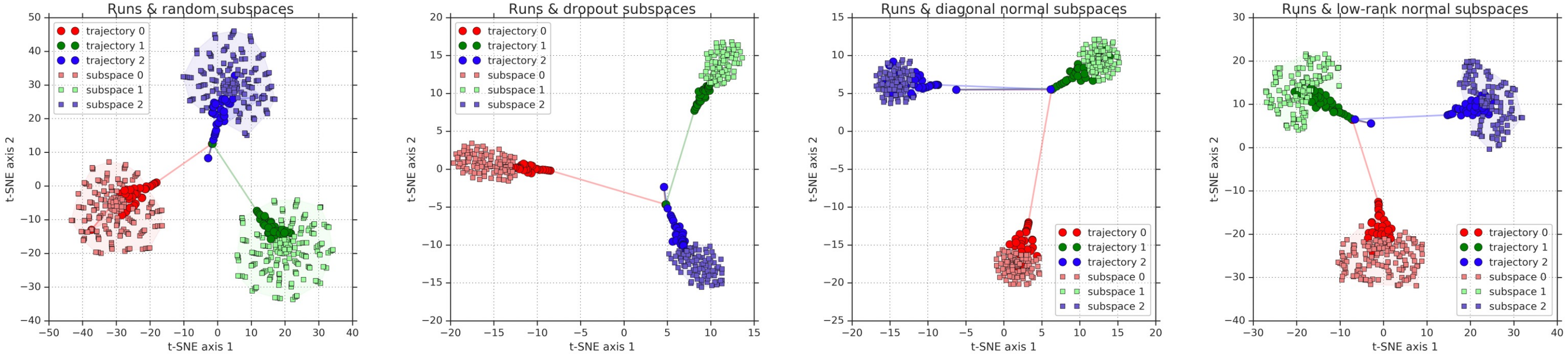


Better calibration to OOD

Maddox, Wesley J., et al. "A simple baseline for bayesian uncertainty in deep learning." *Advances in Neural Information Processing Systems* 32 (2019).

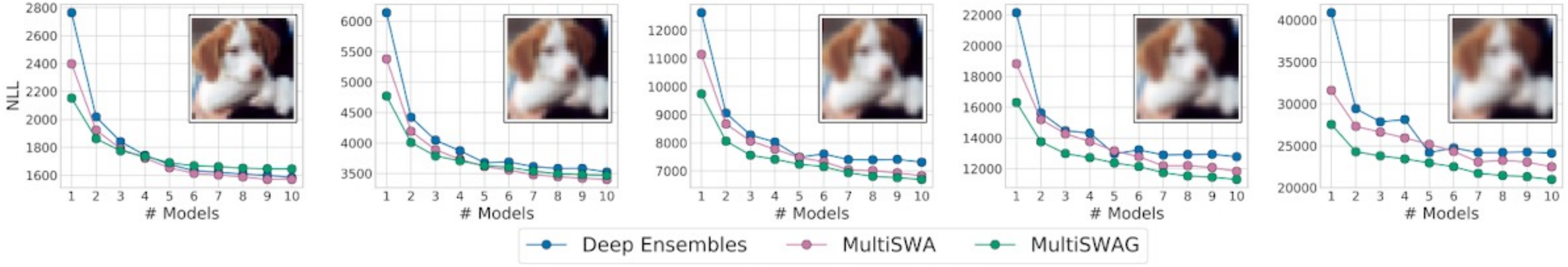
Prediction diversity

- Functions from the single training trajectory is <u>very similar in prediction</u>



Subspace sampling (by square), Initialization (by different color)

- Multi-modal approach to BMA leads to better generalization : $1/M \sum_i \mathcal{N}(\theta_{swa}^{(i)}, \Sigma^{(i)})$



Fort, Stanislav, Huiyi Hu, and Balaji Lakshminarayanan. "Deep ensembles: A loss landscape perspective." *arXiv preprint arXiv:1912.02757* (2019).
Wilson, Andrew G., and Pavel Izmailov. "Bayesian deep learning and a probabilistic perspective of generalization." *Advances in neural information processing systems* 33 (2020): 4697-4708.

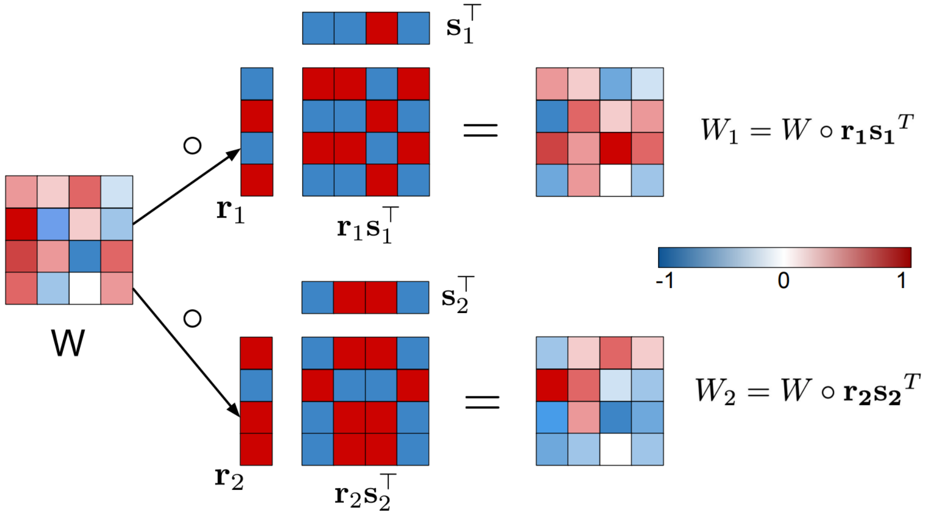
Scalability 2 : Parameter sharing

- **Core idea**
  - Sharing weights W, construct rank-1 matrix F_m for each ensemble member
    - <u>Low memory overhead</u>
    - $\overline{W}_m = W \otimes F_m$ where $F_m = r_m s_m^T$
  - Batch computation is available
    - <u>Low computation overhead</u>
    - $y_i = act\left((W \otimes r_m s_m^T)x_i\right)$ where $W \in \mathbb{R}^{P\times Q}$

      $\Rightarrow Y = act\left(\left((X \otimes R)W\right) \otimes S\right)$

      where $R = [r_1^T, \dots, r_M^T] \in \mathbb{R}^{M\times P}$

      $S = [s_1^T, \dots, s_M^T] \in \mathbb{R}^{M\times Q}$
- **Property**
  - Diversity is encouraged when:
    - Initializing $\{r_m, s_m\}_{m=1}^M$ to random sign vectors
    - Each ensemble is trained w/ different sub-batch



	Single	MC-drop	BatchE	NaiveE
C10	95.31	95.72	**95.94**	96.30
C100	78.32	78.89	**80.32**	81.02

Accuracy

	Single	MC-drop	BatchE	NaiveE
C10	3.27	2.89	**2.37**	2.32
C100	9.28	8.99	**8.89**	6.82

Expected Calibration Error

$$\mathcal{L} = -\frac{N}{B}\sum_{b=1}^{B} \mathbb{E}_{q(\mathbf{r})q(\mathbf{s})}[\log p(y_b \mid \mathbf{x}_b, \mathbf{W}, \mathbf{r}, \mathbf{s})]$$
$$\boxed{+\,\mathrm{KL}(q(\mathbf{r})\|p(\mathbf{r})) + \mathrm{KL}(q(\mathbf{s})\|p(\mathbf{s}))} - \log p(\mathbf{W})$$

Diversity

Wen, Yeming, Dustin Tran, and Jimmy Ba. "Batchensemble: an alternative approach to efficient ensemble and lifelong learning." *arXiv preprint arXiv:2002.06715* (2020).
Dusenberry, Michael, et al. "Efficient and scalable bayesian neural nets with rank-1 factors." *International conference on machine learning*. PMLR, 2020.

Scalability 3 : Distribution of function (GP)

$$p^*(y|x) = p^*(y|x, x \in X_{IND}) \cdot p^*(x \in X_{IND}) + p^*(y|x, x \notin X_{IND}) \cdot p^*(x \notin X_{IND})$$

- **Overview**
  - "Input distance-aware" is the key to reliable uncertainty estimation
    - For safety-critical application, one can manually choose conservative $p^*(y|x, x \notin X_{IND})$
      ex) Uniform distribution for classification task
    - Then, the only thing that matters is knowing $p^*(x \in X_{IND})$ = knowing (dis-)similarity b/t data

  - Acquiring the input distance-aware for DNN
    1. make the feature extractor **input distance-preserving**
       Spectral normalization to weights of feature extractor
    2. make the output layer **feature distance-aware**
       Laplace approximation with Random Fourier Feature

Liu, Jeremiah, et al. "Simple and principled uncertainty estimation with deterministic deep learning via distance awareness." *Advances in Neural Information Processing Systems* 33 (2020): 7498-7512.

Input distance-preserving

- Bi-Lipchitz condition for the feature extractor $h(\cdot)$
  - $L_1\|x - x'\|_{\mathrm{X}} \leq \|h(x) - h(x')\|_H \leq L_2\|x - x'\|_X$ where $0 < L_1 < 1 < L_2$
    ($\Leftrightarrow L_1 \leq \|h\|_{Lip} \leq L_2$)
    - Lower : <u>not to be loosely invariant</u> to semantically meaningful change
    - Upper : <u>not to be overly sensitive</u> to semantically meaningless perturbation

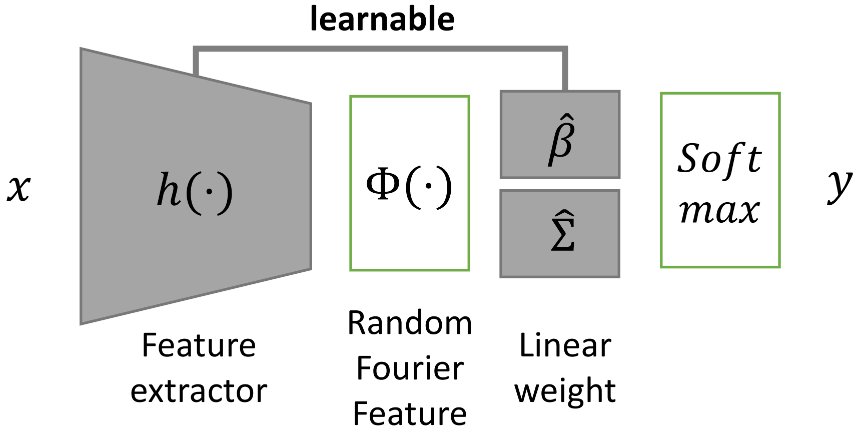
- For the residual block mapping:
  - $r_{l+1}(x) = x + r_l(x)$ where $r_l(x) = act(W_l x + b_l)$
  - Then, $\|r_l\|_{Lip} \leq \sigma(W_l)$ assuming $\|act\|_{Lip} = 1$ where $\sigma(\cdot)$ computes spectral norm
  - $h = r_{L-1} \circ \cdots \circ r_1 \quad \rightarrow \|h\|_{Lip} \leq \prod_{l=1}^{L-1} \sigma(W_l)$

  - Spectral Normalization : $\bar{W}_l \leftarrow c \, * \, W_l/\sigma(W_l)$ if $\sigma(W_l) > c$ so that $\sigma(\bar{W}_l) \leq c$

Miyato, Takeru, et al. "Spectral normalization for generative adversarial networks." *arXiv preprint arXiv:1802.05957* (2018).
Liu, Jeremiah, et al. "Simple and principled uncertainty estimation with deterministic deep learning via distance awareness." *Advances in Neural Information Processing Systems* 33 (2020): 7498-7512.

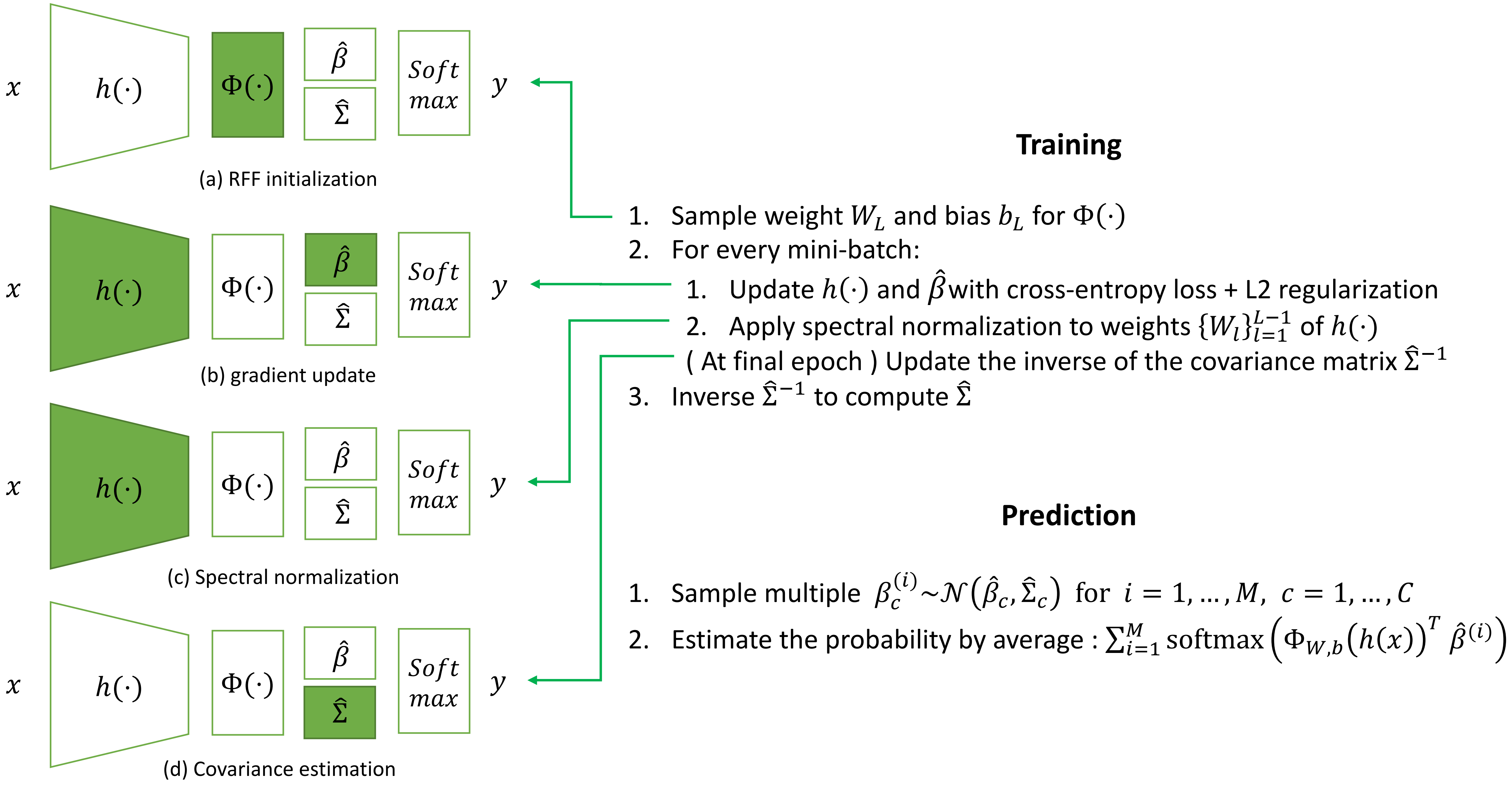
Feature distance-aware

- Random Fourier Features $\Phi(\cdot)$ construct kernel $k(\cdot,\cdot)$
  - $k(h,h') \approx \mathbb{E}_{W,b}\left[\Phi_{W,b}(h)\Phi_{W,b}(h')^T\right]$
    where $\Phi_{W,b}(h) = \sqrt{2/\mathrm{D}_h}\cos(Wh+b)$
    $W_L^{(j)} \sim \mathcal{N}(0,I), \quad b_L^{(j)} \sim U(0,2\pi)$
  - $p(y|x,\beta) = \mathbb{E}_\beta\left[\mathrm{softmax}\left(\Phi_{W,b}\left(h(x)\right)^T \beta\right)\right]$
    where $\beta = [\beta_1, \dots, \beta_C] \in \mathbb{R}^{D_h \times C}$

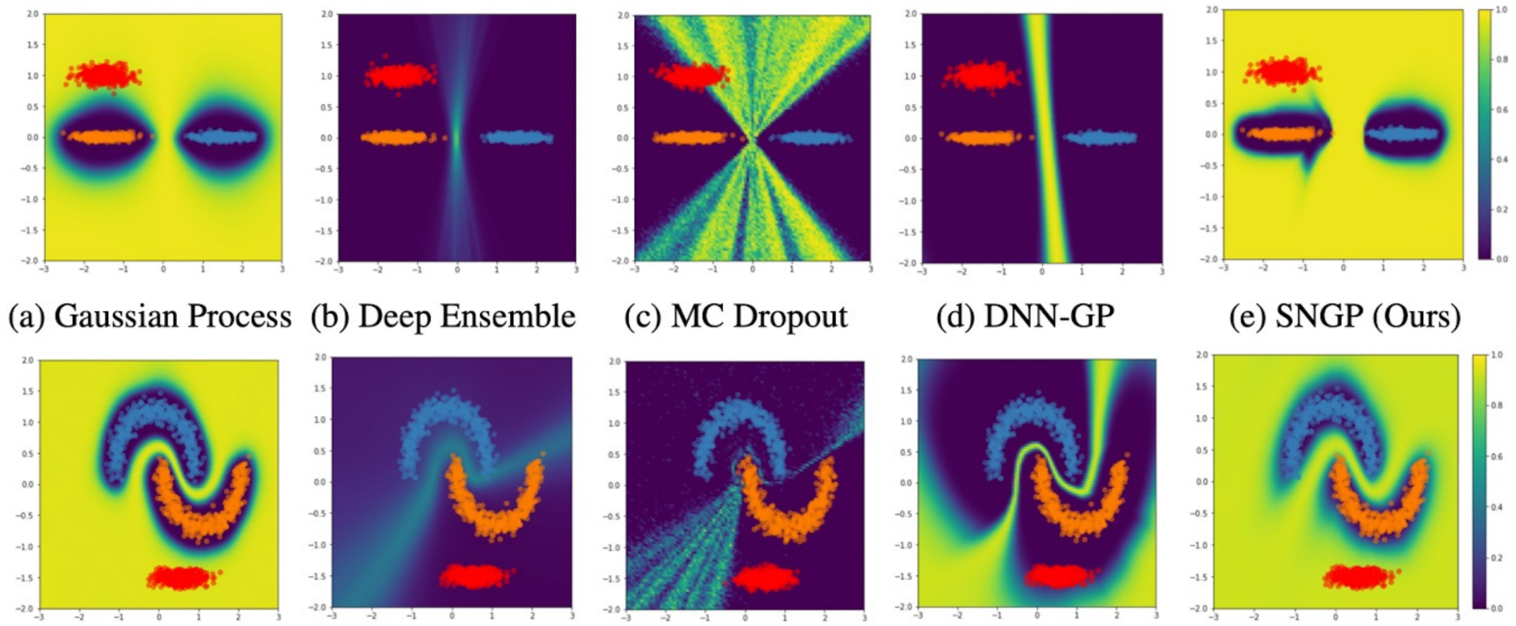


- Laplace approximation to estimate $p(\beta_c|D) \propto p(y|x,\beta_c)p(\beta_c)$
  - Prior : $\beta_c \sim \mathcal{N}(0,I)$
  - Posterior : $\beta_c|x,y \sim \mathcal{N}(\hat{\beta}_c, \hat{\Sigma}_c)$
    - $\hat{\beta}_c = \arg\max_{\beta_c} \log p(y|x,\beta_c)\, p(\beta_c) = \arg\max_{\beta_c} \log p(D|\beta_c) - \frac{1}{2}\|\beta_c\|_2^2$
    - $\hat{\Sigma}_c^{-1} = -\frac{d^2}{d\beta_c^2} log\, p(y|x,\beta_c)p(\beta_c)\,|_{\beta_c=\hat{\beta}_c} = -\frac{d}{d\beta_c}\left(1-p(D|\beta_c)\right)\Phi_{W,b}|_{\beta_c=\hat{\beta}_c} + I$
      $= p\left(y\middle|x,\hat{\beta}_c\right)\left(1-p\left(y\middle|x,\hat{\beta}_c\right)\right)\Phi_{W,b}\Phi_{W,b}^T + I$

Williams, Christopher KI, and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. No. 3. Cambridge, MA: MIT press, 2006.
Rahimi, Ali, and Benjamin Recht. "Random features for large-scale kernel machines." *Advances in neural information processing systems* 20 (2007).
Liu, Jeremiah, et al. "Simple and principled uncertainty estimation with deterministic deep learning via distance awareness." *Advances in Neural Information Processing Systems* 33 (2020): 7498-7512.



x
h(·)
Φ(·)
β̂
Σ̂
Soft max
y
(a) RFF initialization
(b) gradient update
(c) Spectral normalization
(d) Covariance estimation
Training
1. Sample weight W_L and bias b_L for $\Phi(\cdot)$
2. For every mini-batch:
1. Update $h(\cdot)$ and $\hat{\beta}$ with cross-entropy loss + L2 regularization
2. Apply spectral normalization to weights $\{W_l\}_{l=1}^{L-1}$ of $h(\cdot)$
(At final epoch) Update the inverse of the covariance matrix $\hat{\Sigma}^{-1}$
3. Inverse $\hat{\Sigma}^{-1}$ to compute $\hat{\Sigma}$
Prediction
1. Sample multiple $\beta_c^{(i)} \sim \mathcal{N}(\hat{\beta}_c, \hat{\Sigma}_c)$ for $i = 1, \dots, M,\ c = 1, \dots, C$
2. Estimate the probability by average : $\sum_{i=1}^{M} \mathrm{softmax}\left(\Phi_{W,b}(h(x))^T \hat{\beta}^{(i)}\right)$



(a) Gaussian Process (b) Deep Ensemble (c) MC Dropout (d) DNN-GP (e) SNGP (Ours)

(f) Gaussian Process (g) Deep Ensemble (h) MC Dropout (i) DNN-GP (j) SNGP (Ours)

(**Blue**) Positive, (**Orange**) Negative, (**Red**) OOD

Methods	Additional Regularization	Output Layer	Ensemble Training	Multi-pass Inference
Deterministic	-	Dense	-	-
MC Dropout	Dropout	Dense	-	Yes
Deep Ensemble	-	Dense	Yes	Yes
MCD-GP	Dropout	GP	-	Yes
DUQ	Gradient Penalty	RBF	-	-
DNN-SN	Spec Norm	Dense	-	-
DNN-GP	-	GP	-	-
SNGP	Spec Norm	GP	-	-

Model comparison

Method	Accuracy (↑) Clean	Accuracy (↑) Corrupted	ECE (↓) Clean	ECE (↓) Corrupted	NLL (↓) Clean	NLL (↓) Corrupted	OOD AUPR (↑) SVHN	OOD AUPR (↑) CIFAR-100	Latency (↓) (ms / example)
Deterministic	96.0 ± 0.01	72.9 ± 0.01	0.023 ± 0.002	0.153 ± 0.011	0.158 ± 0.01	1.059 ± 0.02	0.781 ± 0.01	0.835 ± 0.01	**3.91**
MC Dropout	96.0 ± 0.01	70.0 ± 0.02	0.021 ± 0.002	0.116 ± 0.009	0.173 ± 0.01	1.152 ± 0.01	0.971 ± 0.01	0.832 ± 0.01	27.10
Deep Ensembles	**96.6 ± 0.01**	**77.9 ± 0.01**	**0.010 ± 0.001**	**0.087 ± 0.004**	**0.114 ± 0.01**	**0.815 ± 0.01**	0.964 ± 0.01	0.888 ± 0.01	38.10
MCD-GP	95.5 ± 0.02	70.0 ± 0.01	0.024 ± 0.004	0.100 ± 0.007	0.172 ± 0.01	1.157 ± 0.01	0.960 ± 0.01	0.863 ± 0.01	29.53
DUQ	94.7 ± 0.02	71.6 ± 0.02	0.034 ± 0.002	0.183 ± 0.011	0.239 ± 0.02	1.348 ± 0.01	0.973 ± 0.01	0.854 ± 0.01	8.68
DNN-SN	96.0 ± 0.01	72.5 ± 0.01	0.025 ± 0.004	0.178 ± 0.013	0.171 ± 0.01	1.306 ± 0.01	0.974 ± 0.01	0.859 ± 0.01	5.20
DNN-GP	95.9 ± 0.01	71.7 ± 0.01	0.029 ± 0.002	0.175 ± 0.008	0.221 ± 0.02	1.380 ± 0.01	0.976 ± 0.01	0.887 ± 0.01	5.58
SNGP (Ours)	95.9 ± 0.01	74.6 ± 0.01	0.018 ± 0.001	0.090± 0.012	0.138 ± 0.01	0.935 ± 0.01	**0.990 ± 0.01**	**0.905 ± 0.01**	6.25

Table 2: Results for Wide ResNet-28-10 on CIFAR-10, averaged over 10 seeds.

Conclusion

- **Deep ensemble** is generally better in terms of Acc, ECE, NLL, etc. than typical BNNs (MC-dropout, VI)

- However, scalability is a main huddle for application:
    1. Loss landscape
        Gather checkpoints along the SGD trajectory (FGE, SWA, SWAG, Multi-SWA(G))
    2. Parameter sharing
        Element-wise product of rank-1 matrix to weight tensor enables efficiency
    3. Distribution of function
        Acquire input distance-preserving feature extractor
        Acquire feature distance-aware classifier