Uncertainty Quantification in Deep Learning

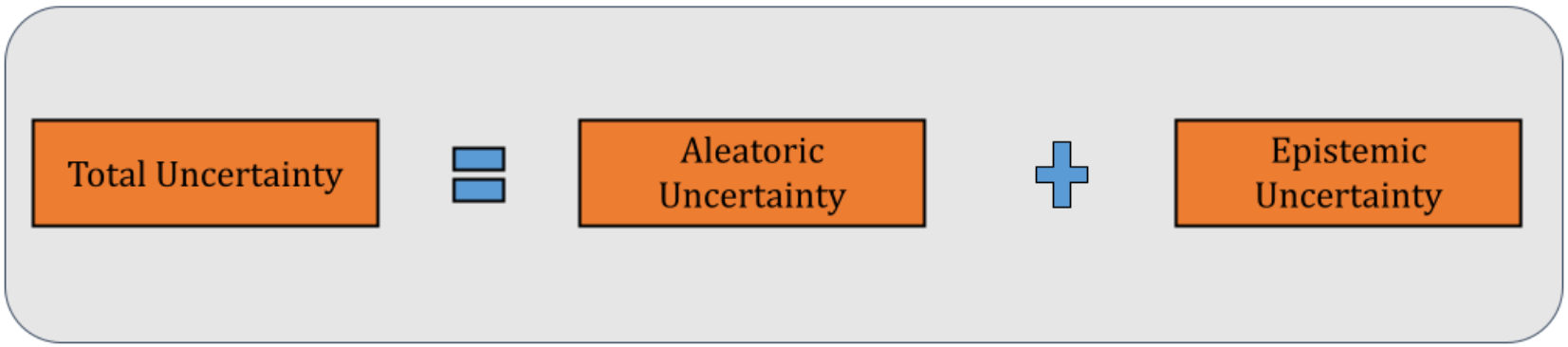
Gianni FRANCHI

U2IS, ENSTA Paris

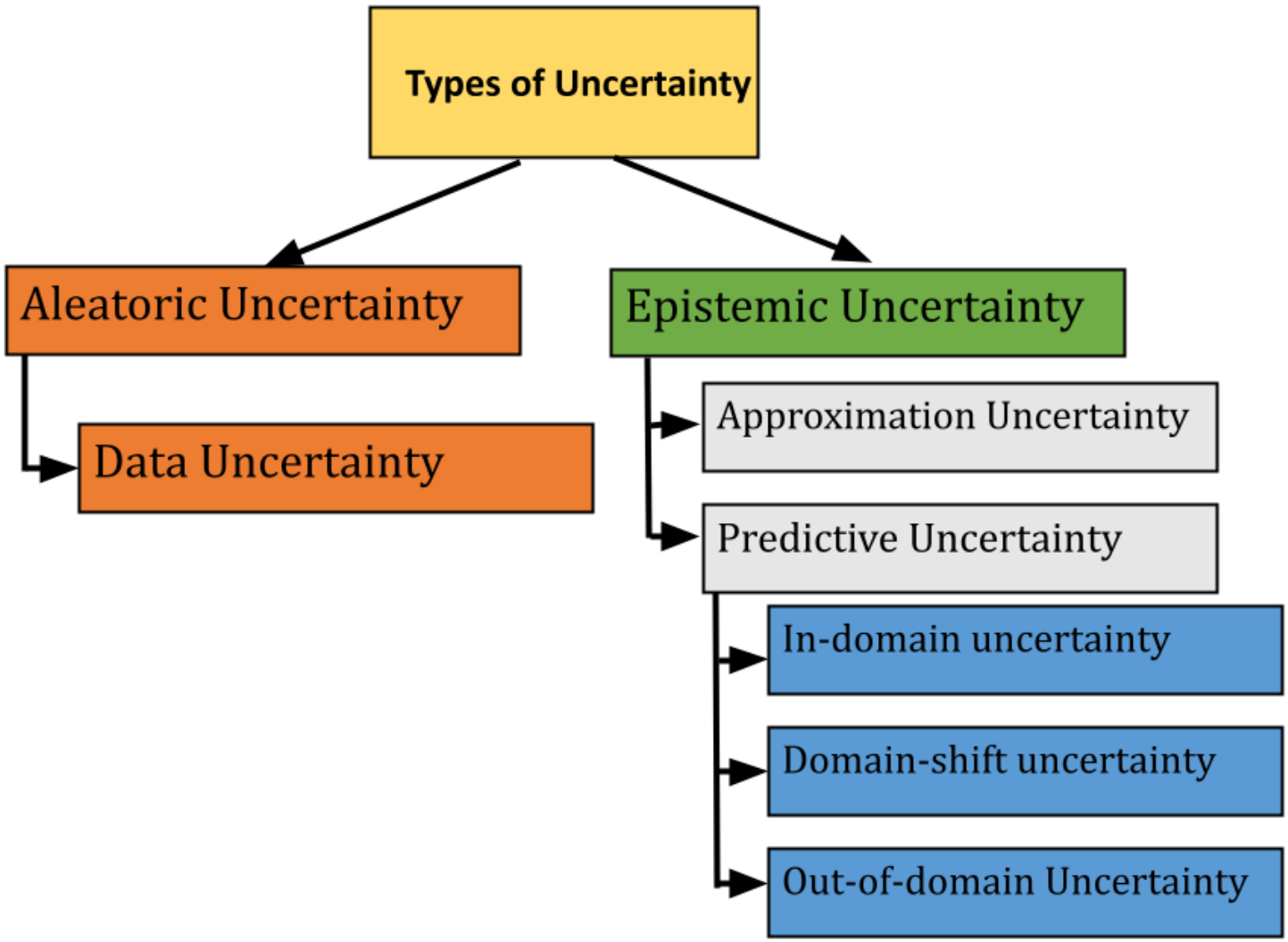
Presentation 2024

Plan

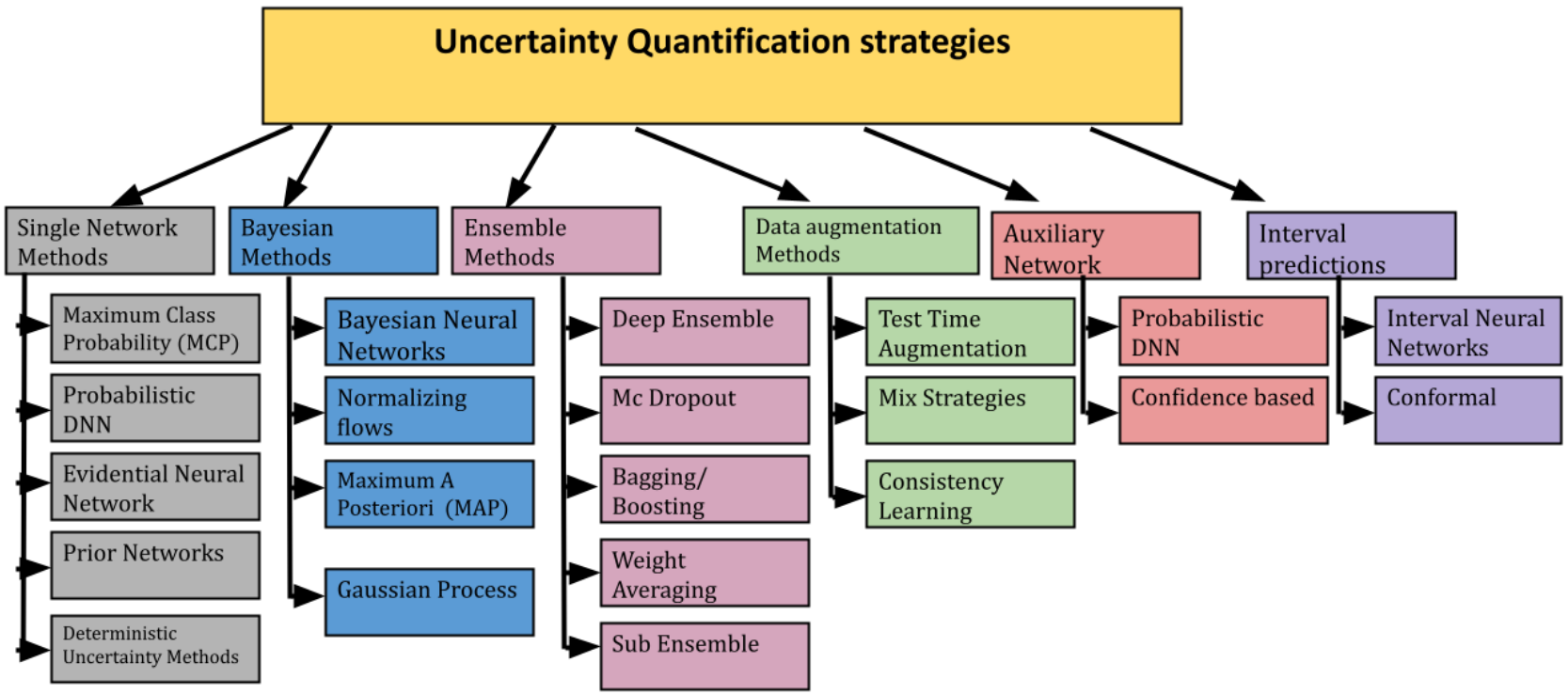
Types of Uncertainty in Machine Learning



Types of Uncertainty in Machine Learning

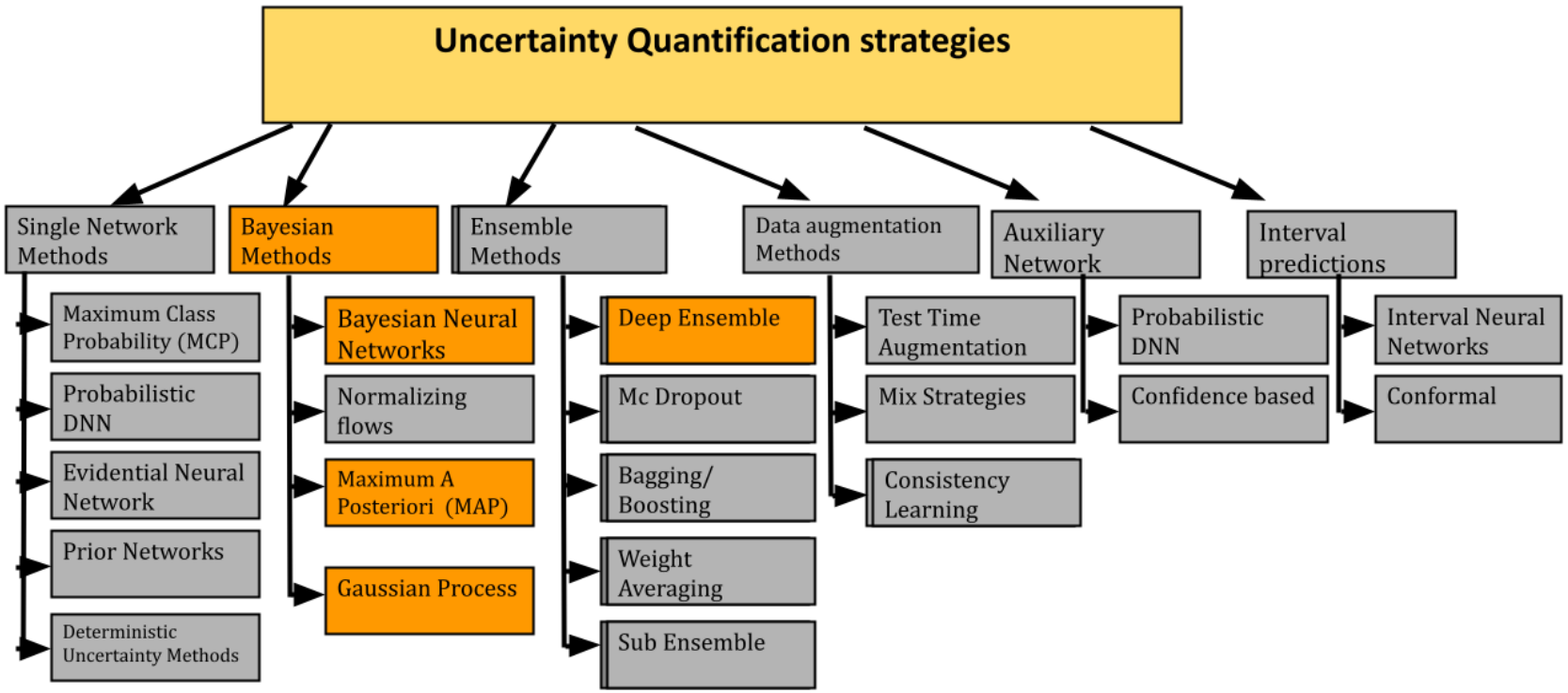


Uncertainty Quantification Strategies



Uncertainty Quantification Strategies

We will focus today just on the Orange blocks.



Computationally-efficient BNNs for Computer Vision

Bayesian approach and DNN

The Goal of DNN is to find $\mathcal{P}(Y|X,\boldsymbol{\omega})$. In the classic Bayesian approach we find $\boldsymbol{\omega}$ such that we have the maximum a posteriori (MAP).

$$\boldsymbol{\omega} = \arg\max_{\boldsymbol{\omega}} \log P(\boldsymbol{\omega}|\mathcal{D})$$
$$\boldsymbol{\omega} = \arg\max_{\boldsymbol{\omega}} \log P(\mathcal{D}|\boldsymbol{\omega}) + \log P(\boldsymbol{\omega})$$

This leads to l2 regularization.

Bayesian DNN [17]

Bayesian DNN is based on marginalization instead of MAP optimization.

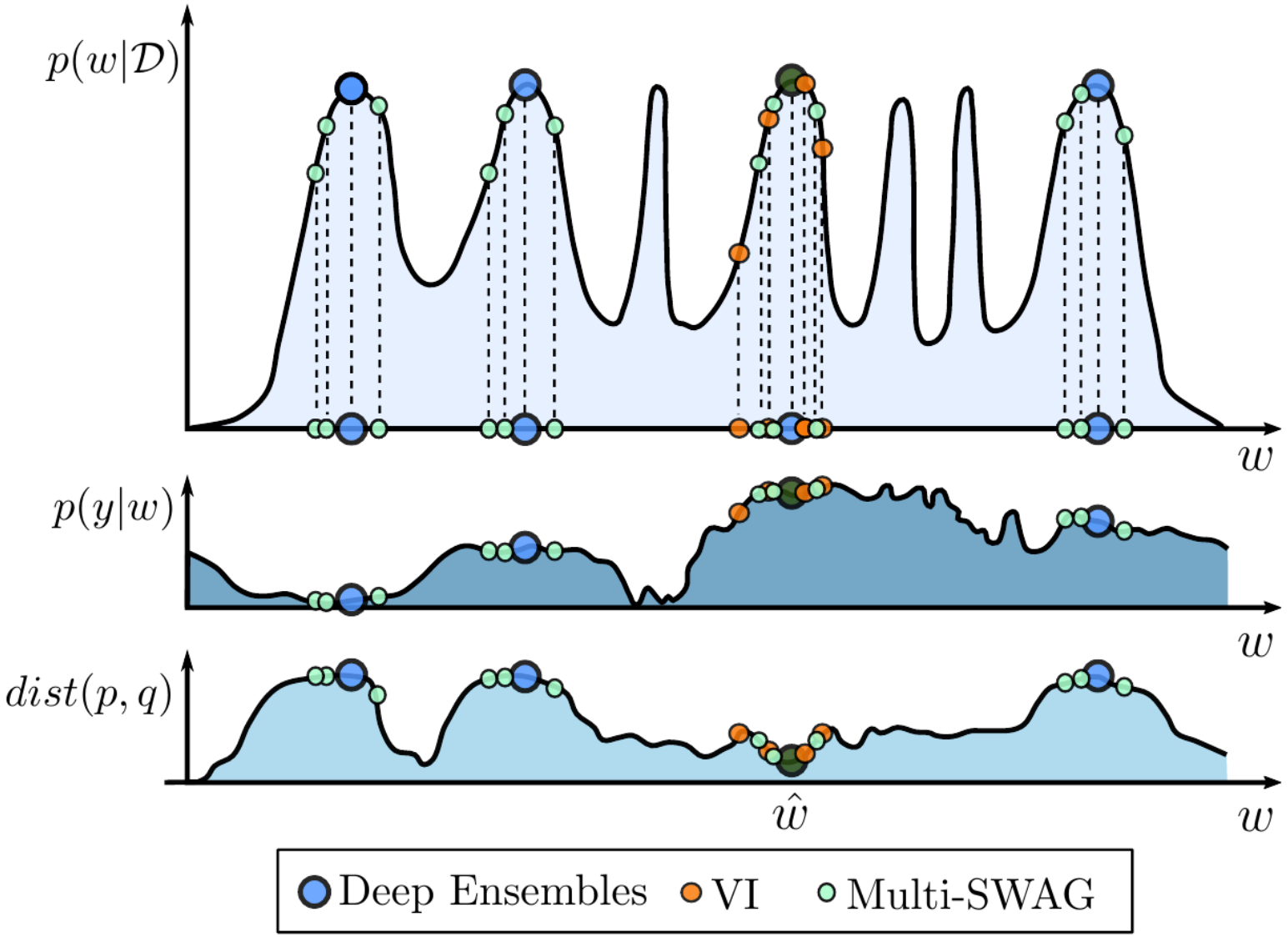
$$\mathcal{P}(Y|X) = \mathbb{E}_{\omega \sim \mathcal{P}(\omega|\mathcal{D})}\left(\mathcal{P}(Y|X,\omega)\right)$$
$$\mathcal{P}(Y|X) = \int \mathcal{P}(Y|X,\omega)\mathcal{P}(\omega|\mathcal{D})d\omega$$

In practice:

$$\mathcal{P}(Y|X) \simeq \sum_i \left(\mathcal{P}(Y|X,\omega_i)\right) \text{ with } \omega_i \sim \mathcal{P}(\omega|\mathcal{D})$$

Different techniques to estimate $\mathcal{P}(\omega|\mathcal{D})$.

Posterior and ensemble [21]



Symmetries impact posterior visualization

Most successful deep learning uncertainty quantification methods – Deep Ensembles, SWA(G), Laplace, Monte-Carlo Dropout, etc – seek to approximate the Bayesian posterior via marginalization over the weights:

$$p(y \mid \boldsymbol{x}, \mathcal{D}) = \int_{\boldsymbol{\omega} \in \Omega} p(y \mid \boldsymbol{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega} \mid \mathcal{D}) d\boldsymbol{\omega}$$

However, in modern deep neural networks, there exists a large or infinite number of corresponding weight configurations:
⇀Scaling the weights by sequences of coefficients and their inverse leaves the function unchanged
⇀ Reordering the weights using sequences of permutation matrices and their inverse does not change the function & others

⇀ Symmetries impact optimization, and generalization via the loss landscape.

⇀ What is the impact of symmetries on Bayesian posteriors and their estimation by uncertainty quantification methods?

Symmetries impact posterior visualization

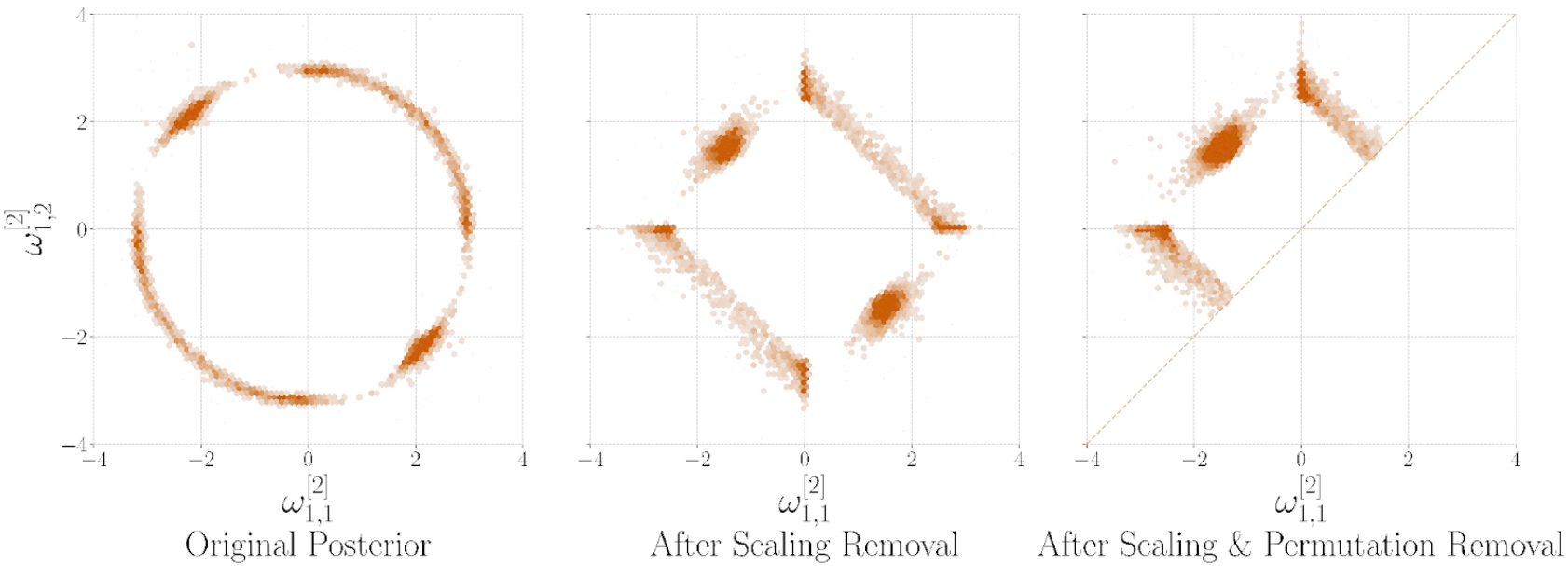


Figure: **Weight-space symmetries impact the estimated Bayesian posterior**. Permutation symmetries clearly increase the number of modes of the posterior distribution in the case of the last layer of a 2-hidden neuron perceptron.

Contributions

1. Express the theoretical impact of perm./scale symmetries on the posterior
2. Evaluate & discuss the impact of symmetries on UQ methods
3. The min. of the weight decay on scaling symmetries has a unique solution.
4. "Checkpoints": dataset of medium-sized independently trained models

Comparing the estimations of the posterior

	Method	MMD / D	MMD / H	NS / D	NS / H	Acc ↑	ECE ↓	ACE ↓	Brier ↓	AUPR ↑	FPR95 ↓	**ID**MI ↓	**OOD**MI ↑
One Mode	Dropout	5.7	5.5	5.7	5.4	89.9	2.0	7.3	14.7	64.9	81.6	7.8	5.0
	BNN	8.4	6.7	8.5	6.6	88.0	20.0	10.3	19.9	78.1	38.6	**0.1**	0.3
	SWAG	6.0	5.7	3.9	4.5	89.7	**0.8**	5.5	14.7	82.7	34.2	2.0	7.5
	Laplace	5.5	4.9	6.6	5.4	90.4	5.5	7.1	15.1	79.8	42.8	1.1	4.5
Multi Mode	Dropout	0.7	2.7	0.7	2.6	93.7	2.3	2.7	9.7	91.0	26.4	11.5	**70.7**
	BNN	2.8	2.8	2.9	2.7	92.7	2.4	3.3	11.3	83.2	31.7	15.6	58.2
	SWAG	1.1	2.7	0.9	2.5	92.0	3.3	6.3	11.9	83.3	31.4	7.1	24.2
	Laplace	0.5	3.6	2.2	2.6	92.7	2.0	**2.5**	9.4	90.5	24.1	12.0	61.2
	HMC	2.3	**0.0**	2.2	**0.0**	90.3	5.7	8.1	15.0	90.0	23.7	29.7	90.5
	DE	**0.0**	2.3	**0.0**	2.2	**94.3**	2.3	3.1	**8.7**	**92.0**	**20.1**	13.3	67.1

Table: **Comparison of popular methods approximating the Bayesian posterior.** All scores are expressed in %, except for the ACEs, expressed in %. Acc stands for accuracy, and **ID**MI and **OOD**MI are in-distribution and out-of-distribution mutual information. NS is the MMD computed after removing the symmetries, and DE stands for Deep Ensembles. MMD / D and NS / D are the MMD and NS computed with a target distribution based on DE and MMD / H and NS / H are based on HMC. Multi-mode methods are based on ten independently trained models except for HMC which is based on three independent chains.

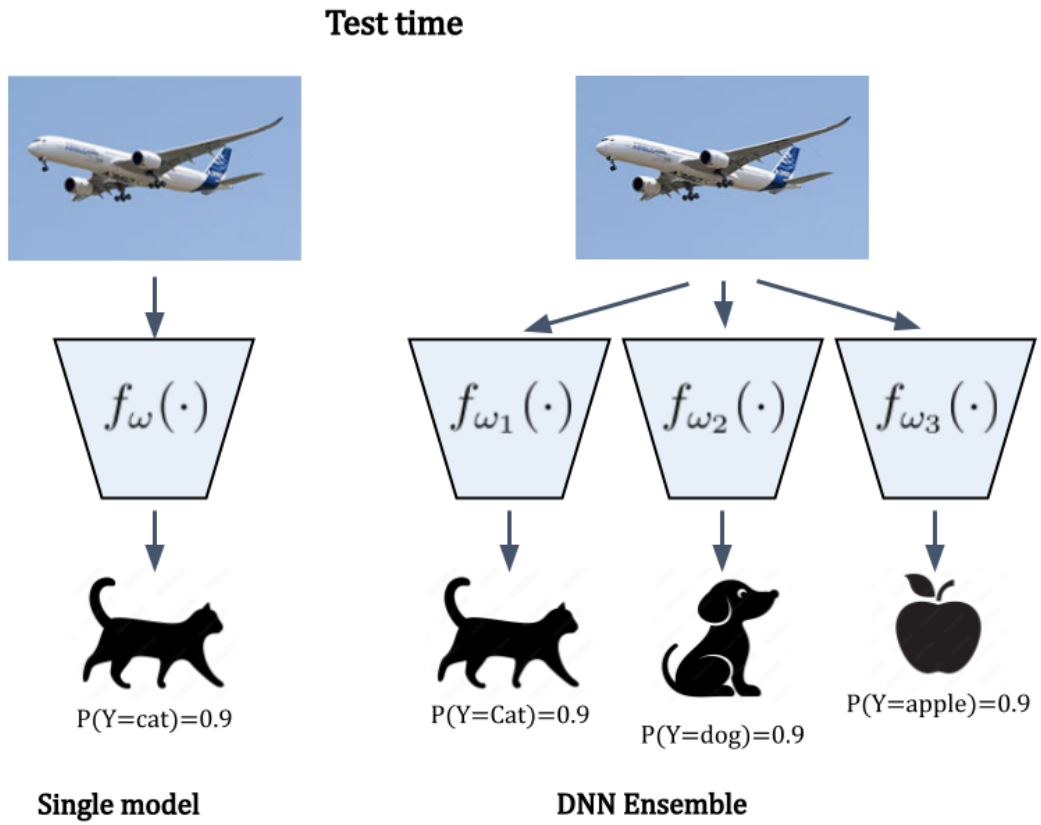
Code and dataset

Easy to download models & scripts:

- 1,024 ResNet-20 FRN/SiLU - CIFAR-10
- 2,048 ResNet-18 - CIFAR-10
- 9,216 ResNet-18 - CIFAR-100
- 2,048 ResNet-18 - TinyImageNet

Figure: QR code to the dataset stored on HF.

Motivation for Ensemble Methods



Deep Ensembles [13]

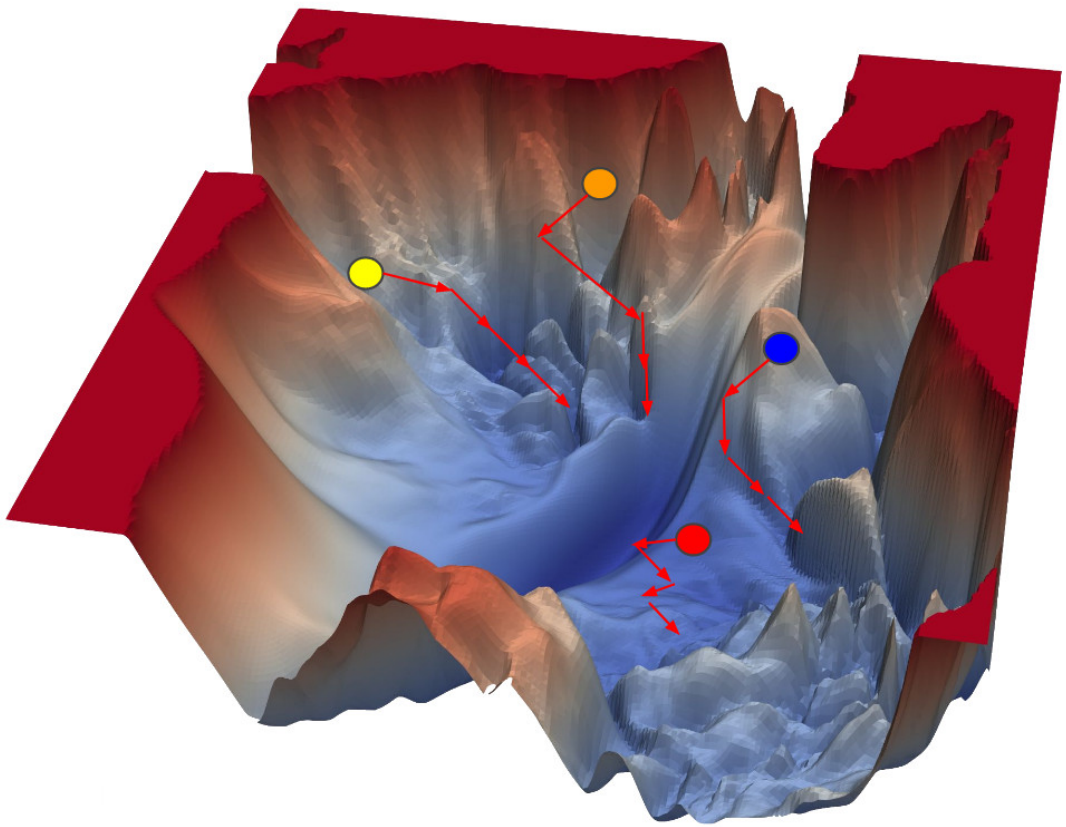
Deep Ensembles

- **Idea:** Deep Ensembles involves training multiple instances of the same DNN model using the same training data.
- **Diversity:** Contrary to Bagging and Boosting Deep Ensemble relies on 3 sources of stochasticity:
  - Stochastic Optimisation
  - Random Initialisation
  - Non-deterministic backpropagation studied in [25]
- **Aggregation:** Predictions are averaged.

Deep Ensembles [13]

They [13] propose to average the predictions of several DNNs with different initial seeds:

$$\mathcal{P}(y^*|x^*) = \frac{1}{N_{\text{model}}} \sum_{j=1}^{N_{\text{model}}} \mathcal{P}(y^*|\omega^j(t^*), x^*) \tag{1}$$



Deep Ensembles [18]

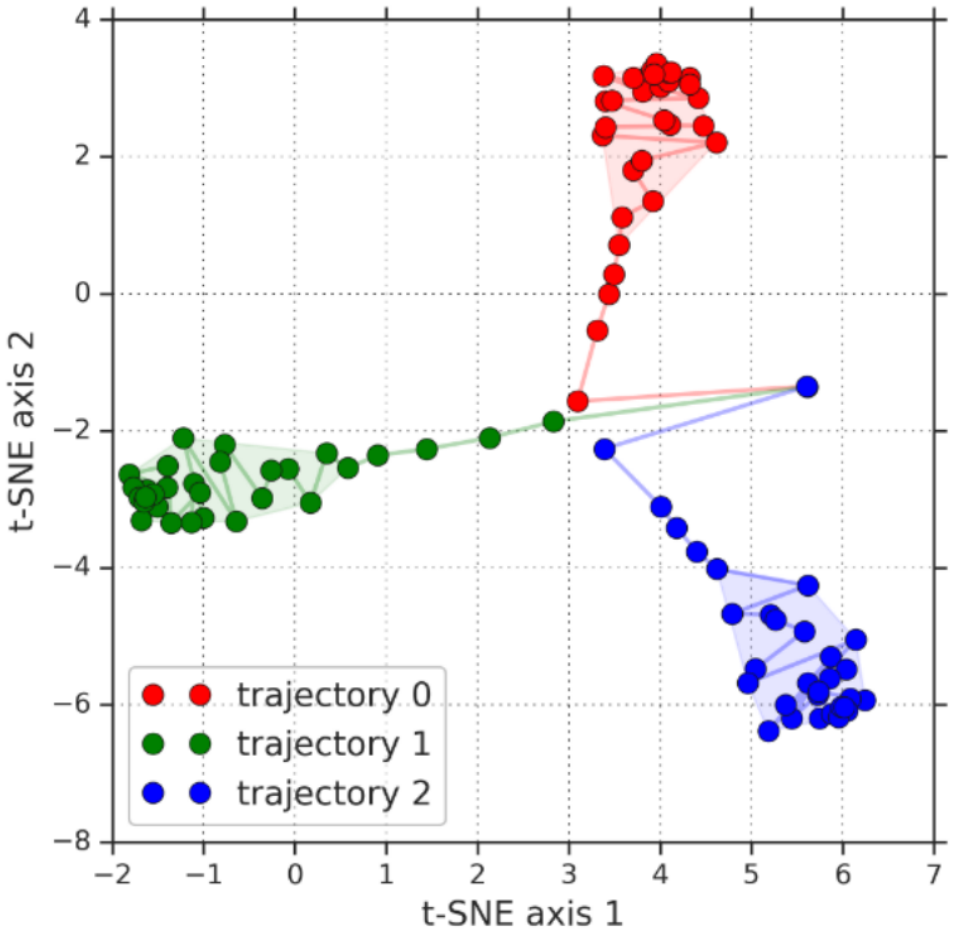


Figure: t-SNE plot of predictions from checkpoints corresponding to 3 different randomly initialized trajectorie

Deep Ensembles [18]

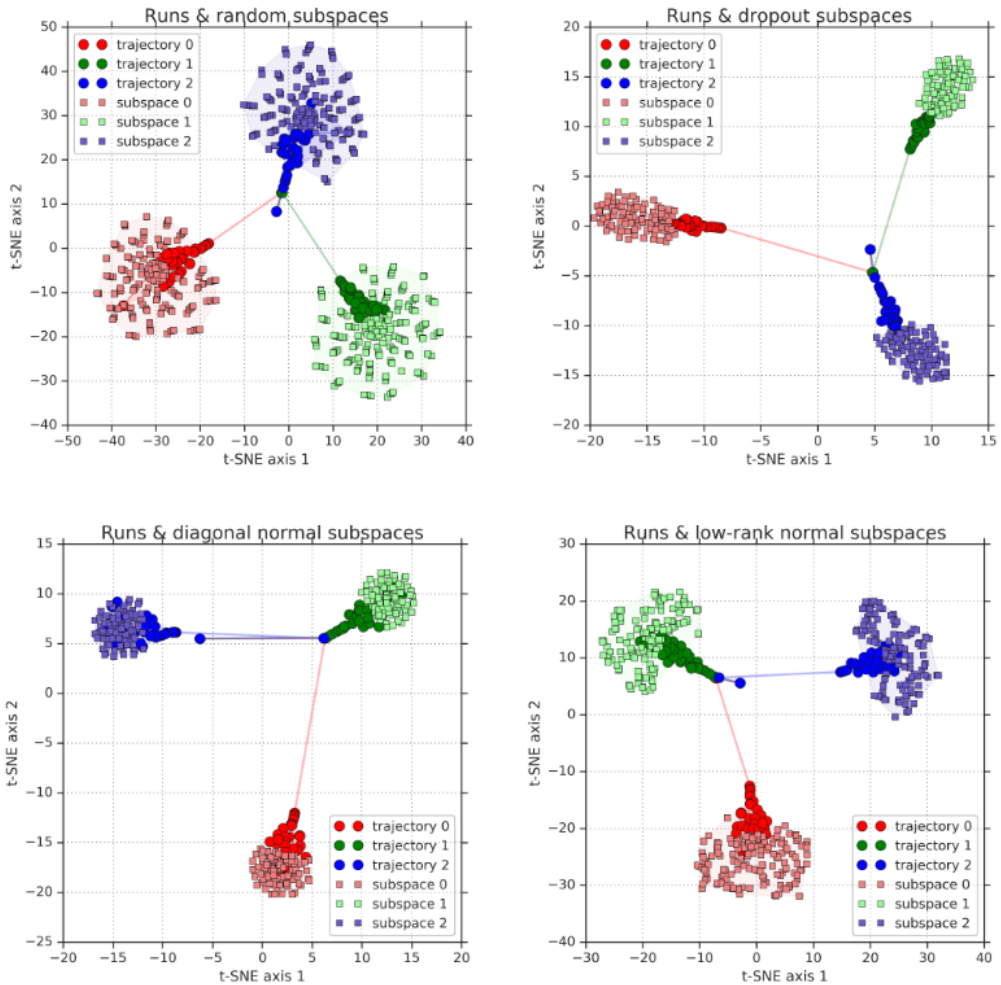


Figure: Results using SimpleCNN on CIFAR-10: t-SNE plots of validation set predictions for each trajectory along with four different subspace generation methods

Deep Ensembles [18]

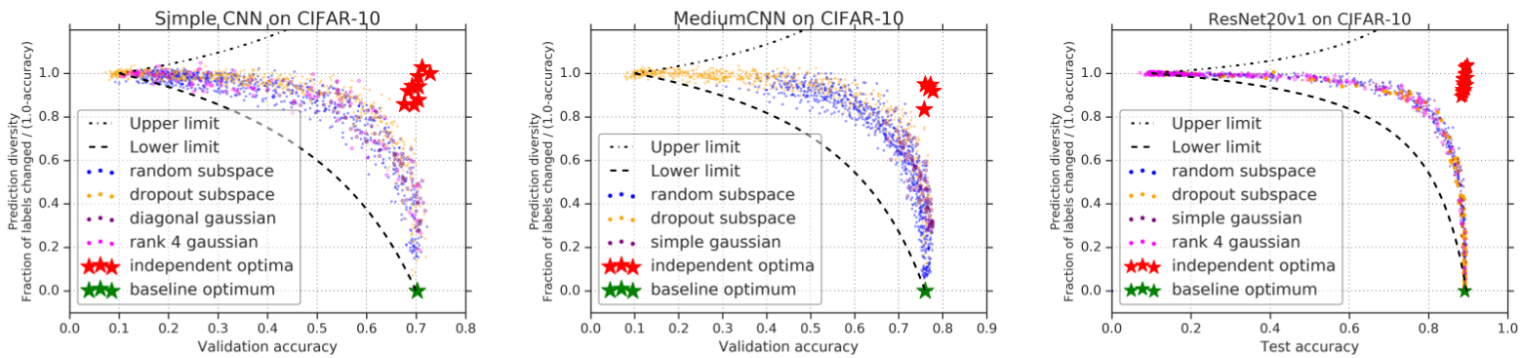
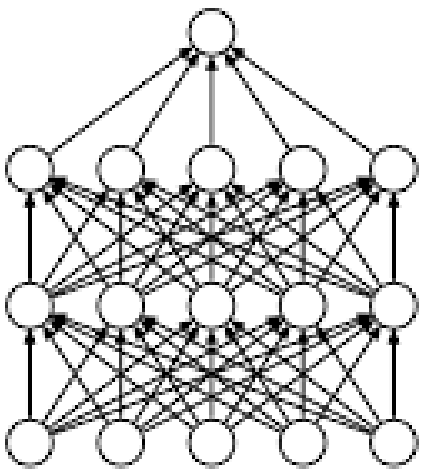


Figure: Diversity versus accuracy plots for 3 models trained on CIFAR-10

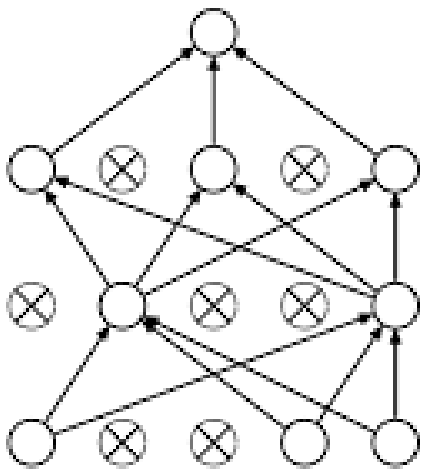
Introduction to Dropout[10]

What is Dropout?

- **Definition:** Dropout is a regularization technique used in neural networks to prevent overfitting.
- **Idea:** During training, randomly "drop out" (ignore) a fraction of neurons, forcing the network to be more robust and preventing reliance on specific neurons.



(a) Standard Neural Net



(b) After applying dropout.

How Dropout Works[10]

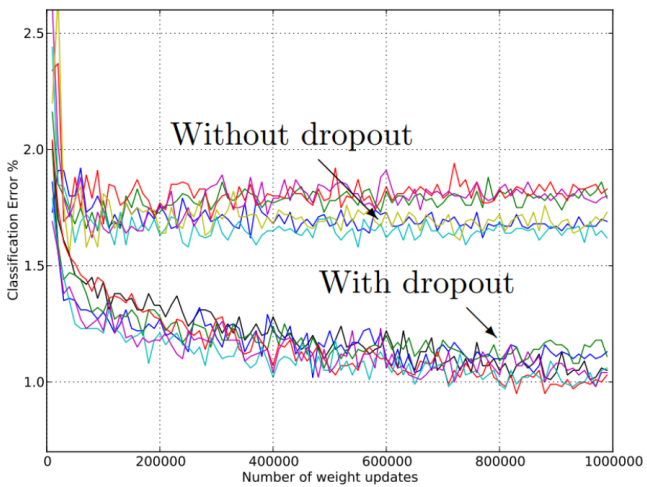
Mechanism of Dropout

- **Training Phase:** In each training iteration, random neurons are dropped out with a specified probability.
- **Variability:** Dropping out neurons introduces variability, making the network less sensitive to the presence of any individual neuron.
- **Ensemble Effect:** Dropout can be seen as training an ensemble of multiple subnetworks, each missing different neurons.

Benefits and Considerations

Advantages and Considerations of Dropout

- **Regularization:** Dropout helps prevent overfitting, improving the model's generalization to unseen data.
- **Ensemble Training:** Provides an implicit way to train multiple models simultaneously, enhancing robustness.
- **Hyperparameter:** The dropout rate is a hyperparameter that needs to be tuned based on the specific task and dataset.
- **Impact on Training Time:** While dropout is beneficial during training, it is typically turned off during inference.

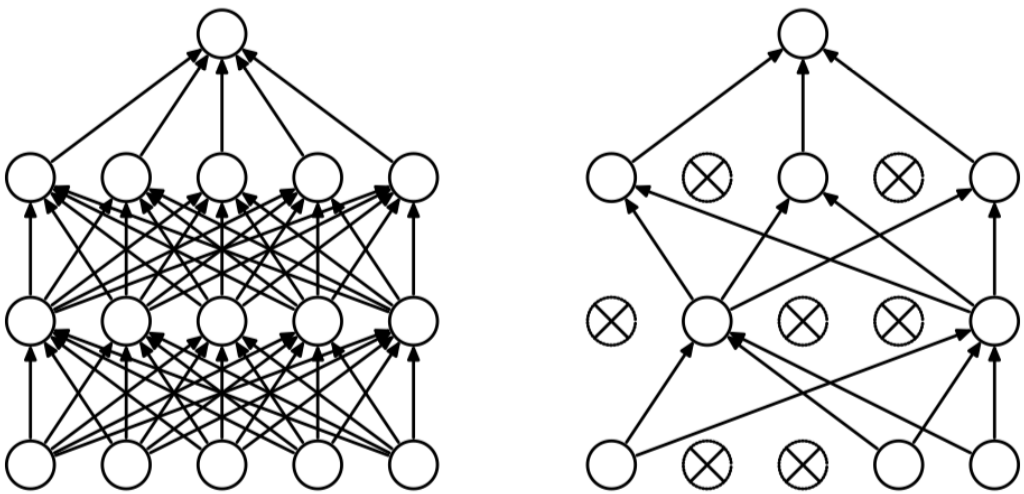


MC dropout [12]

They [12] propose to average the predictions of several DNNs where they apply the dropout:

$$\mathcal{P}(y^*|x^*) = \frac{1}{N_{\text{model}}} \sum_{j=1}^{N_{\text{model}}} \mathcal{P}(y^*|\omega(t^*) \odot b^j, x^*) \tag{2}$$

with b^j a vector of the same size of $\omega(t^*)$ which is a realization of a binomial distribution.



(a) Standard Neural Net

(b) After applying dropout.

Overview of Light Ensemble Methods

Overview of Light Ensemble Methods

- While Deep Ensemble is frequently considered state-of-the-art (SOTA), it comes with significant computational demands.
- Light Ensemble methods offer a faster alternative to achieve comparable results.
- Light Ensemble methods can be performed on a reduced dataset, or/and with fewer neurons, or/and for a shorter duration.

BatchEnsemble Overview [22]

What is BatchEnsemble?

- **Definition:** BatchEnsemble is an ensemble learning technique designed for improving the performance and robustness of neural networks.
- **Inspiration:** Inspired by ensemble methods, BatchEnsemble extends the concept to the batch dimension during training.

How BatchEnsemble Works

- **Batch-Level Ensembling:** Instead of ensembling models across different training runs, BatchEnsemble ensembles within the same training batch.
- **Variability Across Batches:** Introduces diversity by training multiple instances of the model within each batch, enhancing robustness.

BatchEnsemble Overview [22]

They [22] propose to approximate the average of the predictions of several DNN with different initial seeds by using a DNN with two king of weights. For simplicity is the ω has two set of weight ω^{slow} , ω^{fast} For simplicity let us consider a DNN with just one fully connected layer and let us write $\omega = \{\omega_j\}_{j=1}^{N_{\text{model}}} = \{W_j\}_{j=1}^{N_{\text{model}}}$ and $\omega^{\text{slow}} = W$ and $\omega^{\text{slow}} = \{F_j\}_{j=1}^{N_{\text{model}}}$. We have $W_j = W \cdot F = W \cdot (r_j s_j^t)$

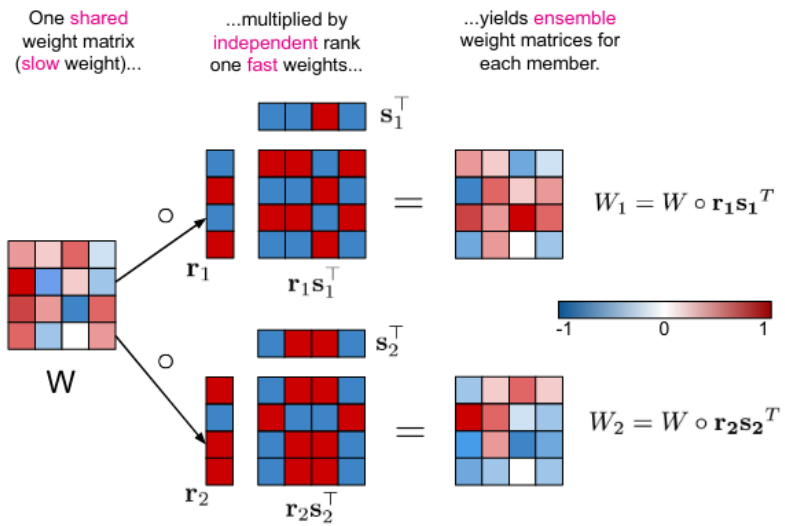


Figure: An illustration on how to generate the ensemble weights for two ensemble members

BatchEnsemble Overview [22]

We have a set of weight $W_j = W \cdot F = W \cdot (r_j s_j^t)$ with W that sees all images and $(r_j s_j^t)$ that does not see all the same images. If we denote ϕ an activation function then when we apply the BatchEnsemble on an image we perform:

$$y = \phi\left(W_j^t x\right) = \phi\left((W^t \cdot (r_j s_j^t))^t x\right) = \phi\left((W^t (x \cdot r_j) \cdot s_j)\right)$$

Similarly to Deep Ensembles, to perform inference we just perform ensembling :

$$\mathcal{P}(y^*|x^*) = \frac{1}{N_{\text{model}}} \sum_{j=1}^{N_{\text{model}}} \mathcal{P}(y^*|\omega_j, x^*) \tag{3}$$

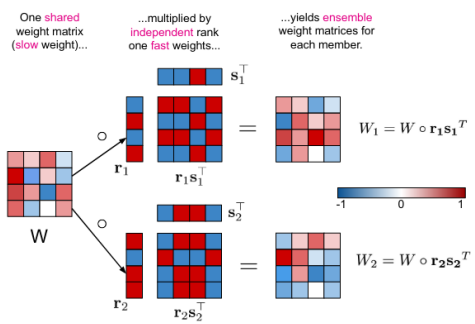


Figure: An illustration on how to generate the ensemble weights for two

MIMO Overview [11]

What is MIMO?

- **Definition:** MIMO stands for Multiple Input Multiple Output .
- **Objective:** MIMO aims to utilize a single model's capacity to train multiple subnetworks that independently learn the task at hand.

Key Concepts of MIMO [11]

How MIMO Works

- **MIMO principle:** The lottery ticket hypothesis shows that one can prune away 70-80% of the connections in a DNN without adversely affecting performance
- **MIMO Idea:**The basic Idea is that a neural network has sufficient capacity to fit 3-4 independent subnetworks simultaneously. Hence they just need to modify the input and output to handle this 3-4 subnetworks.

Key Concepts of MIMO [11]

```python
class MIMOModel(nn.Module):
    def __init__(self, hidden_dim: int = 784, ensemble_num: int = 3):
        super(MIMOModel, self).__init__()
        self.input_layer = nn.Linear(hidden_dim, hidden_dim * ensemble_num)
        self.backbone_model = BackboneModel(hidden_dim, ensemble_num)
        self.ensemble_num = ensemble_num
        self.output_layer = nn.Linear(128, 10 * ensemble_num)

    def forward(self, input_tensor: torch.Tensor):
        input_tensor = input_tensor.transpose(1, 0).view(
            batch_size, self.ensemble_num, -1)
        # (batch_size, ensemble_num, hidden_dim)
        input_tensor = self.input_layer(input_tensor)
        # (batch_size, ensemble_num, hidden_dim * ensemble_num)
        # usual model forward
        output = self.backbone_model(input_tensor)  # (batch_size, ensemble_num,
        ↪  128)
        output = self.output_layer(output)  # (batch_size, ensemble_num, 10 *
        ↪  ensemble_num)
        output = output.reshape(
            batch_size, ensemble_num, -1, ensemble_num
        )  # (batch_size, ensemble_num, 10, ensemble_num)
        output = torch.diagonal(output, offset=0, dim1=1, dim2=3).transpose(2, 1)
        ↪  # (batch_size, ensemble_num, 10)
        output = F.log_softmax(output, dim=-1)  # (batch_size, ensemble_num, 10)
        return output
```

Key Concepts of MIMO [11]

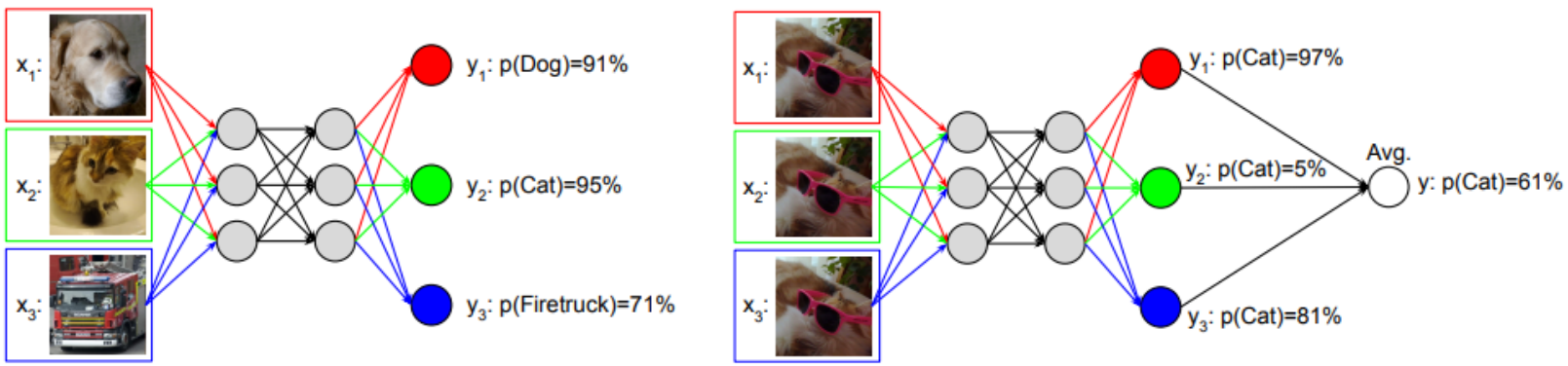


Figure: The multi-input multi-output (MIMO) configuration, the network takes M = 3 inputs and gives M outputs [11]

Packed-Ensembles Overview [25]

Seamless training of ensembles with Packed-Ensembles

- **Definition:** Packed-Ensembles estimate the posterior distributions restraining their support to smaller networks in a computationally efficient manner with grouped convolutions.
- **Objective:** Get the benefits of *deep ensembles* with reduced costs.

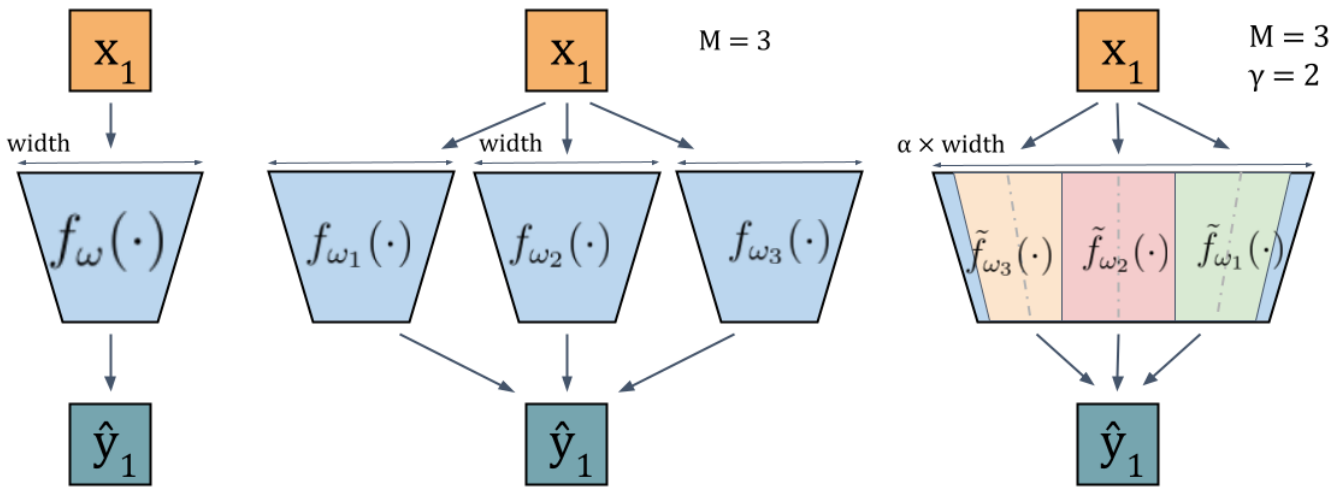


Figure: **Left:** A standard network, **Center:** A *deep ensembles*, **Right:** The corresponding Packed-Ensembles

How well does Packed-Ensembles perform? [25]

Performance of Packed-Ensembles

- **Performance:** For sufficiently large networks, Packed-Ensembles is equivalent to *deep-ensembles* in performance and UQ.
- **Computational efficiency:** Use Packed-Ensembles with *float16* to benefit from grouped-convolutions better.

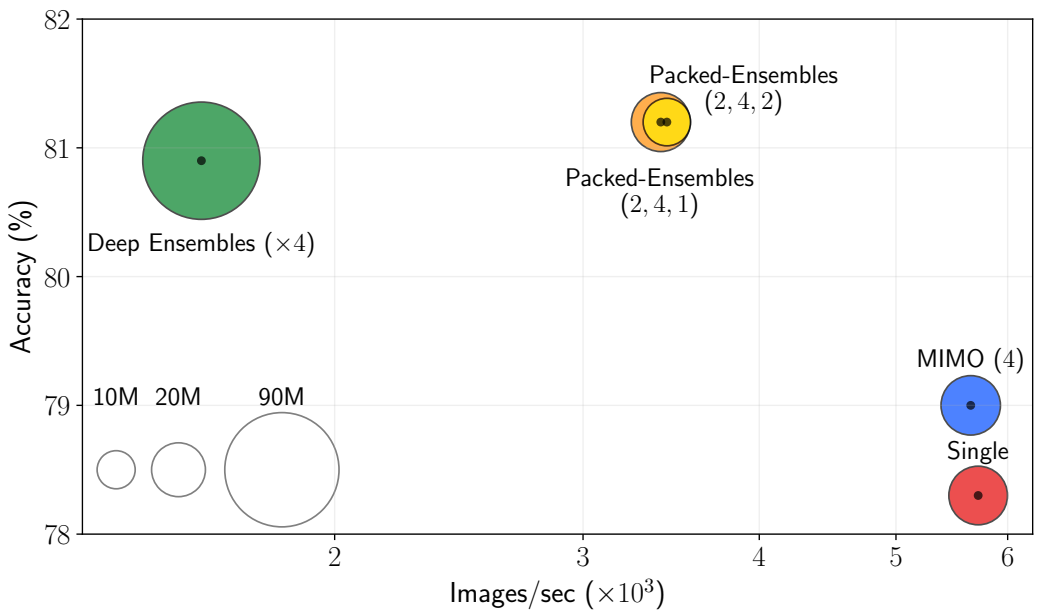


Figure: Performance (accuracy) wrt. the image throughput

Sources of stochasticity in deep ensembles [25]

Stochasticity			ResNet-50				
ND	**DI**	**DB**	**Acc** (↑)	**ECE** (↓)	**AUPR** (↑)	IDMI	OODMI
-	-	-	77.63±0.23	0.0825±0.0018	89.19±0.65	0±0	0±0
✓	-	-	80.94±0.10	0.0179±0.0010	90.23±0.62	0.1513	0.4022
-	✓	-	81.01±0.06	0.0202±0.0011	91.10±0.39	0.1524	0.4088
-	-	✓	80.87±0.10	0.0178±0.0010	90.80±0.30	0.1505	0.4115
✓	✓	✓	81.08±0.08	0.0198±0.0013	90.68±0.25	0.1534	0.4031

Figure: Impact of the three sources of stochasticity, non-deterministic backdrop. kernels (**ND**), different initialization (**DI**), and different batches (**DB**).

Uncertainty-sources are equivalent!

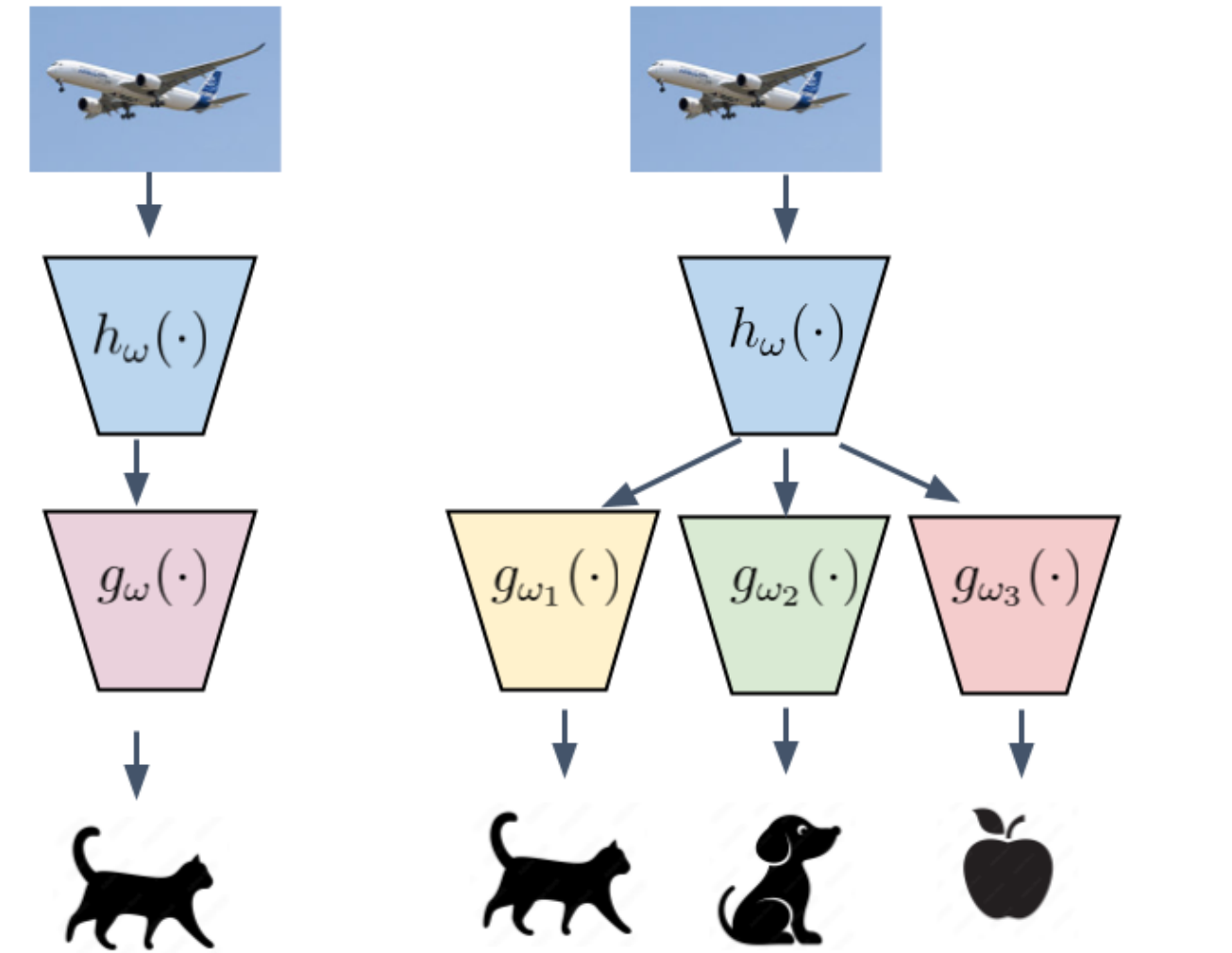
No source of stochasticity during training seems to single out. Having one source is sufficient, and adding more does not seem to affect the performance or the quantitative functional diversity (Mutual information).

Key Takeaways

Summary of Insights

- **Understanding Sources:** We explored various sources impacting DNNs, acknowledging the inherent uncertainties.
- **Distinguishing Types:** The distinction between aleatoric and epistemic uncertainty provided clarity on different uncertainty manifestations.
- **Quantification Techniques:** We delved into diverse methods for quantifying uncertainty in DNNs.
- **Evaluation Approaches:** Different techniques for evaluating the effectiveness of uncertainty quantification were discussed.

Last Layer sampling [29]



Last Layer sampling [29]

Last Layer Sampling

- **Idea:** Instead of estimation the posterior for the entire model, only we estimate the posterior only for the last layer.
- **Benefits:** Reduces computational cost and memory, while still benefiting from ensemble-based uncertainty estimation.
- **Diversity:** Each ensemble member has a unique final layer, leading to diverse predictions while sharing common features from earlier layers.
- **Aggregation:** Predictions are averaged across the last layer ensemble members, similar to full ensembles.

TRADI: a Bayesian DNN

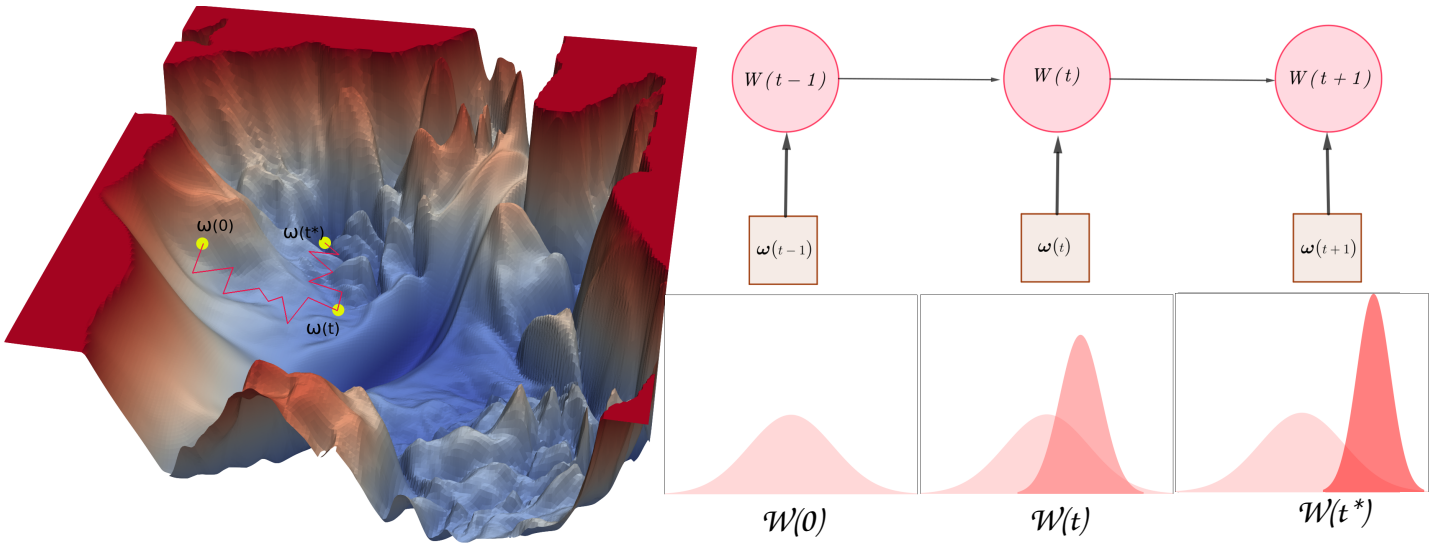


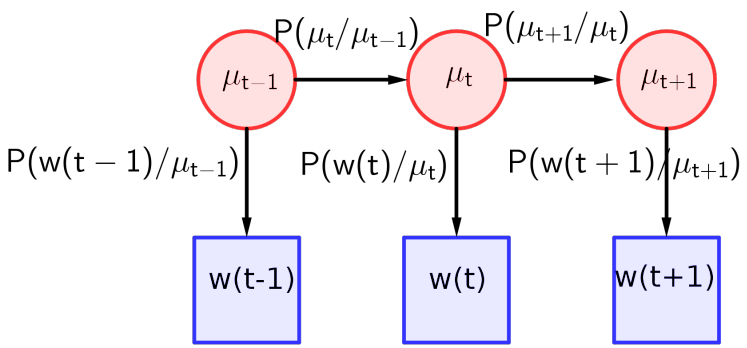
Figure: TRADI uses Kalman filtering for tracking the distribution $\mathcal{W}$ of all DNN weights across training steps from a generic prior $\mathcal{W}(0)$ to the final estimate $\mathcal{W}(t^*)$.

TRADI: Mean State and Measurement Equations

Equations for the Mean $\mu_k(t)$

$$\mu_k(t) = \mu_k(t-1) - \eta \nabla \mathcal{L}_{\omega_k(t)} + \varepsilon_\mu,$$
$$\omega_k(t) = \mu_k(t) + \tilde{\varepsilon}_\mu,$$

- $\varepsilon_\mu \sim \mathcal{N}(0, \sigma_\mu^2)$: State noise.
- $\tilde{\varepsilon}_\mu \sim \mathcal{N}(0, \tilde{\sigma}_\mu^2)$: Observation noise.



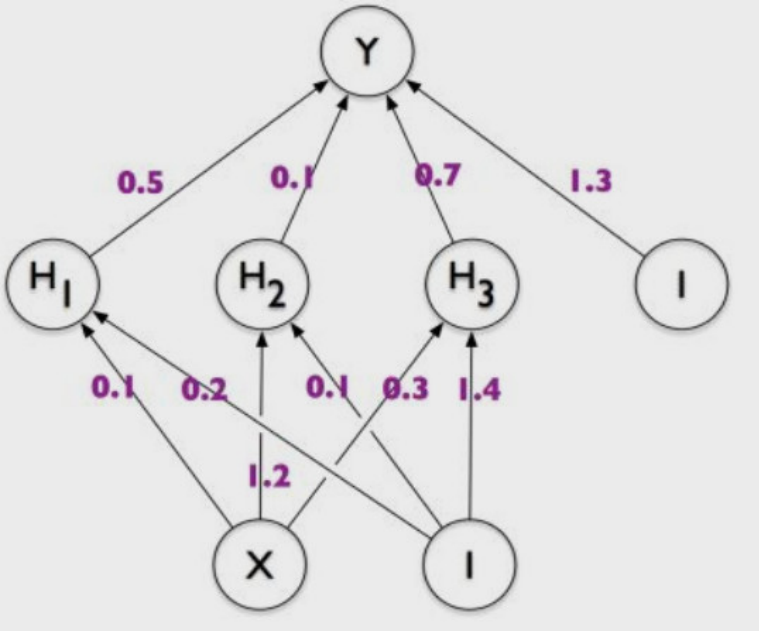
TRADI: Variance State and Measurement Equations

Equations for the Variance $\sigma_k^2(t)$

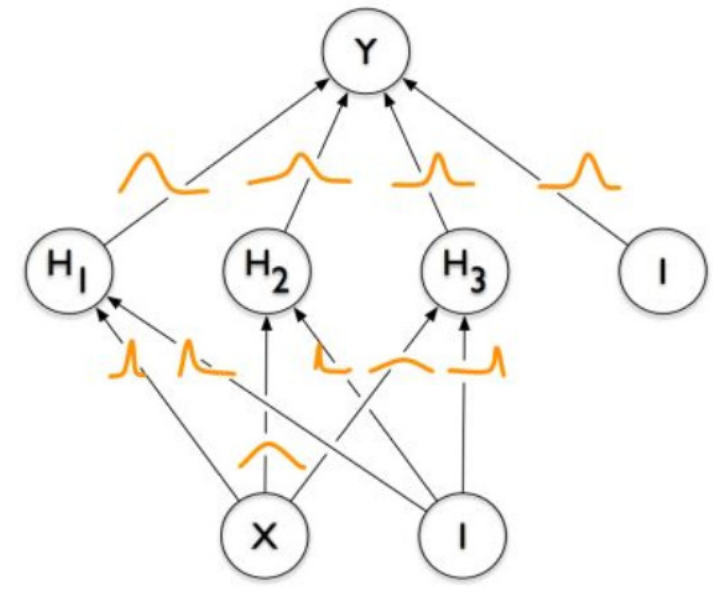
$$\sigma_k^2(t) = \sigma_k^2(t-1) + \left(\eta \nabla \mathcal{L}_{\omega_k(t)}\right)^2 - \eta^2 \mu_k(t)^2 + \varepsilon_\sigma,$$
$$z_k(t) = \sigma_k^2(t) - \mu_k(t)^2 + \tilde{\varepsilon}_\sigma.$$

- $\varepsilon_\sigma \sim \mathcal{N}(0, \sigma_\sigma^2)$: State noise.
- $\tilde{\varepsilon}_\sigma \sim \mathcal{N}(0, \tilde{\sigma}_\sigma^2)$: Observation noise.

TRADI: Comparison - Normal DNN vs Bayesian DNN



Normal DNN



Bayesian DNN

TRADI: Sampling New Realizations of Weights

Weight Sampling Equation

$$\tilde{\omega}(t^*) = \mu(t^*) + \Sigma^{1/2}(t^*) \times \boldsymbol{m}_1,$$

where Σ is the covariance matrix, and $\boldsymbol{m}_1 \sim \mathcal{N}(0_K, \boldsymbol{I}_K)$.

Prediction

$$\mathcal{P}(y^*|x^*) = \frac{1}{N_{\text{model}}} \sum_{j=1}^{N_{\text{model}}} \mathcal{P}(y^*|\tilde{\omega}^j(t^*), x^*).$$

TRADI: Efficient BNN Strategy

Key Idea

- TRADI is a Bayesian Neural Network (BNN) strategy that tracks the **posterior distribution** of the Deep Neural Network (DNN) during training.

Benefits of TRADI

- **Lightweight Approach**: Unlike traditional BNNs, TRADI does not introduce heavy computational overhead.
- **Non-intrusive**: It does not perturb the DNN training process, allowing it to run as efficiently as a standard DNN.
- **Posterior Tracking**: By monitoring the posterior, TRADI enhances the uncertainty estimation while keeping the model's original structure intact.

How to estimate the Posterior of BNN?

Classical VI-BNN

Using the "reparametrization trick", a layer j of an MLP can be written:

$$\begin{aligned} \boldsymbol{u}_j &= \text{norm}\left(\left[W_\mu^{(j)} + \boldsymbol{\epsilon}_j W_\sigma^{(j)}\right] \boldsymbol{h}_{j-1}, \beta_j, \gamma_j\right), \text{ and} \\ \boldsymbol{a}_j &= a(\boldsymbol{u}_j), \end{aligned} \tag{4}$$

where the matrices $W_\mu^{(j)}$ and $W_\sigma^{(j)}$ denote the mean and standard deviation of the posterior distribution of layer j, $\boldsymbol{\epsilon}_j \sim \mathcal{N}(0, \mathbb{1})$ and the operator $\text{norm}(\cdot, \beta_j, \gamma_j)$, of trainable parameters β_j and γ_j, can refer to any batch, layer, or instance normalization.

How to turn a DNN into a BNN?

ABNN

Our objective differs from VI-BNN, which requires training the posterior distribution parameters from scratch. Instead, our approach entails leveraging and converting an existing DNN into a BNN.

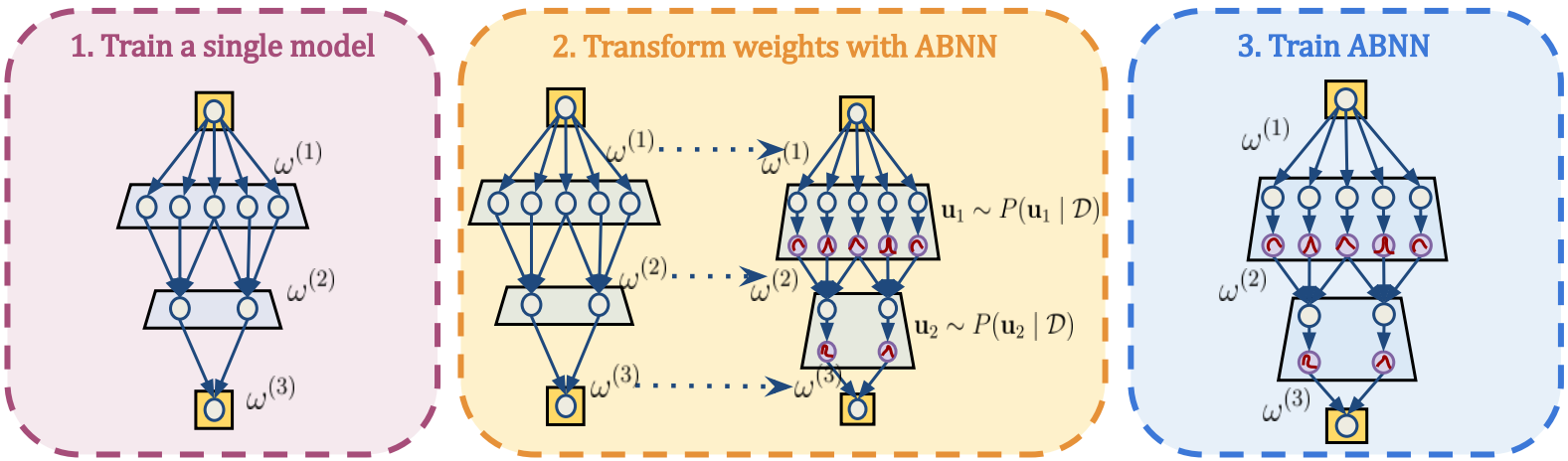


Figure: **Illustration of the training process for the ABNN.** The procedure begins with training a single DNN $\boldsymbol{\omega}_{\mathrm{MAP}}$, followed by architectural adjustments to transform it into an ABNN. The final step involves fine-tuning the ABNN model.

Converting DNNs into BNNs

Post-hoc Bayesian Strategy

- **Base Strategy:** Start with pre-trained DNNs with normalization layers like Batch, Layer, or Instance normalization.
- **Bayesian Adaptation:** Replace deterministic normalization layers with Bayesian Normalization Layers (BNL) that add Gaussian perturbation.
- **Goal:** Efficiently convert pre-trained DNNs into Bayesian Neural Networks (BNNs) with minimal modifications.

Bayesian Normalization Layer (BNL)

Transforming Normalization Layers

- **Key Equation:**

$$\boldsymbol{u}_j = \mathbf{BNL}\left(W^{(j)}\boldsymbol{h}_{j-1}\right), \text{ and } \boldsymbol{a}_j = a(\boldsymbol{u}_j), \text{ with}$$

$$\mathbf{BNL}(\boldsymbol{h}_j) = \frac{\boldsymbol{h}_j - \hat{\mu}_j}{\hat{\sigma}_j} \times \gamma_j(1+\boldsymbol{\epsilon}_j) + \beta_j,$$

  where $\boldsymbol{\epsilon}_j \sim \mathcal{N}(0, \mathbb{1})$

- **Explanation:** Gaussian perturbation is applied to normalization layers to introduce stochasticity like **gaussian dropout**, transforming deterministic layers into Bayesian layers.
- **Parameters:** γ_j and β_j are learnable vectors, retrained for a limited number of epochs.

Training ABNN

Training Strategy

- **Process:** - After replacing normalization layers with BNL, retrain the parameters for a few epochs.
- **Multi-Modality:** - Instead of training a single model, we train multiple ABNNs, each with different weight configurations $\omega_1, \ldots, \omega_M$.
- **Benefit:** This approach helps in improving the generalization and reliability of the BNN.

Inference in ABNN

Inference Strategy

- **Sampling from ABNN:** - For each ABNN sample, multiple ϵ_j are drawn independently from $\mathcal{N}(0, \mathbb{1})$.
- **Marginalization:** The prediction for a new sample $\boldsymbol{x}$ is the expected outcome from a finite ensemble of models and weight configurations:

$$P(y \mid \boldsymbol{x}, \mathcal{D}) \approx \frac{1}{ML} \sum_{l=1}^{L} \sum_{m=1}^{M} P(y \mid \boldsymbol{x}, \boldsymbol{\omega}_m, \boldsymbol{\epsilon}_l).$$

Benefits of ABNN

Key Advantages

- **Uncertainty Estimation:** BNNs provide a probabilistic interpretation of model predictions.
- **Efficient Conversion:** Pre-trained DNNs can be easily adapted into BNNs with minimal retraining.
- **Scalability:** The use of Bayesian Normalization Layers (BNL) allows leveraging modern architectures like ResNet and Vision Transformers.

Conclusions

Advantages of Deep Ensembles

- Ensembles allow sampling from **different modes** of the posterior distribution, enhancing model diversity.
- Each instance of the ensemble can capture distinct patterns, making it robust to overfitting.
- Averaging predictions across multiple models provides better **uncertainty estimates**, especially for Out-of-Distribution (OOD) detection.

Drawbacks of Deep Ensembles

- Training multiple DNNs requires significant **computational resources**, both in training and inference.
- Inference is computationally heavy due to the need for combining predictions from several models.

Conclusions

Advantages of Light Ensembles

- **Light ensembles** are faster to train and deploy, making them computationally efficient.
- However, they might suffer from **lower performance**, especially compared to deep ensembles.

Drawbacks of Light Ensembles

- Suitable for applications where **speed** is critical, but high model performance is less important.

Conclusions

Advantages of BNNs

- BNNs have strong **theoretical foundations**, allowing them to model uncertainty more formally.
- In principle, BNNs approximate the posterior distribution of the model weights.

Drawbacks of BNNs

- **Mode collapse**: BNNs often collapse into a single mode, missing out on multimodal distributions.
- Training is computationally complex, especially for vision tasks, making deployment more challenging.

Conclusions

Next Steps: Implementation

- **Practical Application:** Moving forward, we will explore practical implementations of uncertainty quantification.
- **Link for the Practical Application:** please visit the following link:

`https://drive.google.com/file/d/1GpeHCq5bQDEusUtYHroGNIXDNW4fKMf1/view?usp=sharing`

Exploring Further

- **Contribute to Torch Uncertainty:** If you are interested in advancing the field, consider contributing to Torch Uncertainty.

`https://github.com/ENSTA-U2IS-AI/torch-uncertainty`

- **Explore Our Resources:** Check out our curated list of resources on Uncertainty, available at our "awesome" repository.

`https://github.com/ENSTA-U2IS-AI/awesome-uncertainty-deeplearning`

Bibliography:

1 Pan, F., Shin, I., Rameau, F., Lee, S., & Kweon, I. S. (2020). Unsupervised intra-domain adaptation for semantic segmentation through self-supervision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 3764-3773).

2 Sakaridis, Christos, Dengxin Dai, and Luc Van Gool. "ACDC: The adverse conditions dataset with correspondences for semantic driving scene understanding." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021.

3 Bergmann, P., Fauser, M., Sattlegger, D., & Steger, C. (2019). MVTec AD–A comprehensive real-world dataset for unsupervised anomaly detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 9592-9600).

4 Hendrycks, D., Basart, S., Mazeika, M., Zou, A., Kwon, J., Mostajabi, M., ... & Song, D. (2019). Scaling out-of-distribution detection for real-world settings. arXiv preprint arXiv:1911.11132.

Bibliography:

5 Chan, R., Lis, K., Uhlemeyer, S., Blum, H., Honari, S., Siegwart, R., ... & Rottmann, M. (2021). Segmentmeifyoucan: A benchmark for anomaly segmentation. arXiv preprint arXiv:2104.14812.

6 Franchi, G., Yu, X., Bursuc, A., Tena, A., Kazmierczak, R., Dubuisson, S., ... & Filliat, D. (2022). MUAD: Multiple Uncertainties for Autonomous Driving, a benchmark for multiple uncertainty types and tasks. arXiv preprint arXiv:2203.01437.

7 Gawlikowski, J., Tassi, C.R.N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R. and Shahzad, M., 2023. A survey of uncertainty in deep neural networks. Artificial Intelligence Review, 56(Suppl 1), pp.1513-1589.

8 Hullermeier, Eyke, and Willem Waegeman. "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods." Machine Learning 110 (2021): 457-506.

Bibliography:

9 Guo, Chuan, et al. "On calibration of modern neural networks." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.

10 Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

11 Havasi, M., Jenatton, R., Fort, S., Liu, J. Z., Snoek, J., Lakshminarayanan, B., ... & Tran, D. (2020). Training independent subnetworks for robust prediction. arXiv preprint arXiv:2010.06610.

12 Gal, Yarin, and Zoubin Ghahramani. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." international conference on machine learning. 2016.

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

Bibliography:

14 Kendall, Alex, and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?." Advances in neural information processing systems. 2017.

15 A.G. Wilson, P. Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. Advances in Neural Information Processing Systems, 2020.

16 Ilg, E., Cicek, O., Galesso, S., Klein, A., Makansi, O., Hutter, F., & Brox, T. (2018). Uncertainty estimates and multi-hypotheses networks for optical flow. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 652-667).

17 Blundell, Charles, et al. "Weight uncertainty in neural networks." arXiv preprint arXiv:1505.05424 (2015).

18 Fort, Stanislav, Huiyi Hu, and Balaji Lakshminarayanan. "Deep Ensembles: A Loss Landscape Perspective." arXiv preprint arXiv:1912.02757 (2019).

Bibliography:

19 Yu, Xuanlong, Gianni Franchi, and Emanuel Aldea. "SLURP: Side learning uncertainty for regression problems." arXiv preprint arXiv:2110.11182 (2021).

20 Upadhyay, U., Karthik, S., Chen, Y., Mancini, M., & Akata, Z. (2022, October). Bayescap: Bayesian identity cap for calibrated uncertainty in frozen neural networks. In European Conference on Computer Vision (pp. 299-317). Cham: Springer Nature Switzerland.

21 A.G. Wilson, P. Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. Advances in Neural Information Processing Systems, 2020.

22 Wen, Yeming, Dustin Tran, and Jimmy Ba. "Batchensemble: an alternative approach to efficient ensemble and lifelong learning." arXiv preprint arXiv:2002.06715 (2020).

Bibliography:

22 Wen, Yeming, Dustin Tran, and Jimmy Ba. "Batchensemble: an alternative approach to efficient ensemble and lifelong learning." arXiv preprint arXiv:2002.06715 (2020).

23 Hendrycks, Dan, et al. "A Benchmark for Anomaly Segmentation." arXiv preprint arXiv:1911.11132 (2019).

24 Franchi, G., Bursuc, A., Aldea, E., Dubuisson, S., & Bloch, I. (2020). TRADI: Tracking deep neural network weight distributions. In ECCV 2020.

25 Laurent, O., Lafage, A., Tartaglione, E., Daniel, G., Martinez, J. M., Bursuc, A., and Franchi, G. (2022). Packed-Ensembles for Efficient Uncertainty Estimation. In ICLR 2023.

26 Bishop, C. M. (1994). Mixture density networks. Aston University.

27 Vossen, J., Feron, B., Monti and A. (2018). Probabilistic Forecasting of Household Electrical Load Using Artificial Neural Networks.

28 Lafage, A., Barbier, M., Franchi, G., and Filliat, D. (2024). Hierarchical Light Transformer Ensembles for Multimodal Trajectory Forecasting.

Bibliography:

29 Brosse, N., Riquelme, C., Martin, A., Gelly, S., & Moulines, E. (2020). On last-layer algorithms for classification: Decoupling representation from uncertainty estimation. arXiv preprint arXiv:2001.08049.