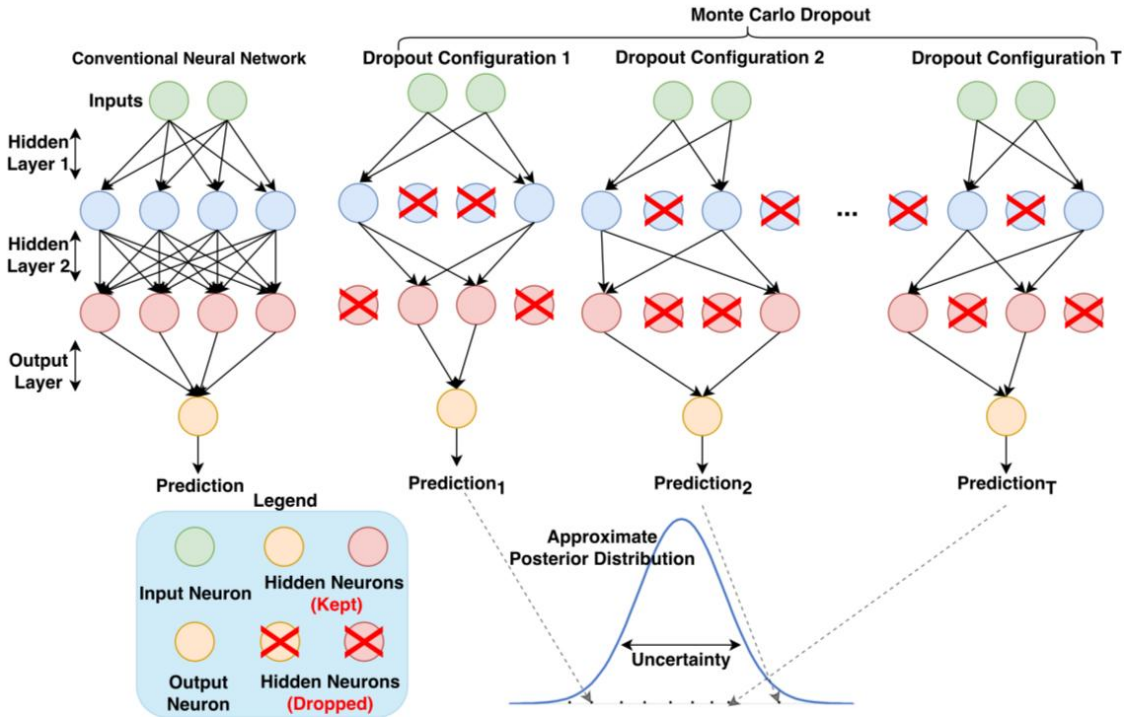


Lancaster University

Prob_AI

Introduction to Bayesian Deep Learning

Andreas Makris, 2nd year PhD student

Lancaster University, ProbAI Hub

Based on new Bayesian Deep Learning book!

Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning, JMLR, 2016
https://www.themoonlight.io/tw/review/testing-spintronics-implemented-monte-carlo-dropout-based-bayesian-neural-networks

Term 2 Schedule (So far)

Schedule

Title	Location	Date	Time	Speaker
Intro to BDL	PSC LT	14 Jan 2026	2 pm - 3 pm	Andreas
Laplace Methods	PSC LT	21 Jan 2026	2 pm - 3 pm	Lanya
Robust & Conjugate Spatio-Temporal GPs	PSC LT	28 Jan 2026	2 pm - 3 pm	William
TBD	PSC LT	4 Feb 2026	2 pm - 3 pm	Paul
Reviewing BNNs	PSC LT	11 Feb 2026	2 pm - 3 pm	Jixiang
GP Connections to BDL	PSC LT	27 Feb 2026	2 pm - 3 pm	Jonas
TBD	PSC LT	4 Mar 2026	2 pm - 3 pm	Connie
TBD	PSC LT	11 Mar 2026	2 pm - 3 pm	Chris
Probabilistic Numerics	PSC LT	18 Mar 2026	2 pm - 3 pm	Rui

+ potentially an active learning talk

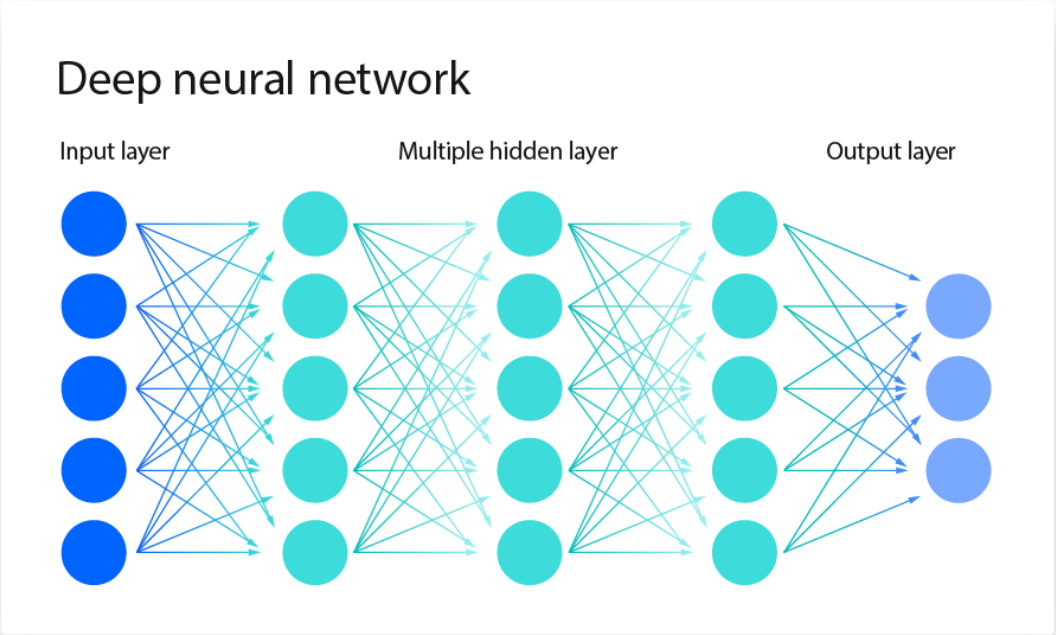
+ potentially a last layer inference talk

Why are we even here?

- **Deep Learning works extremely well. What is it missing?**

- Most neural networks give a point prediction.
- They are often extremely confident, even when they are wrong.
- They do not naturally tell us when they should not be trusted.

This is not a minor issue; it becomes **critical whenever predictions are used to make important decisions** (e.g. medical cases).



https://www.ibm.com/think/topics/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

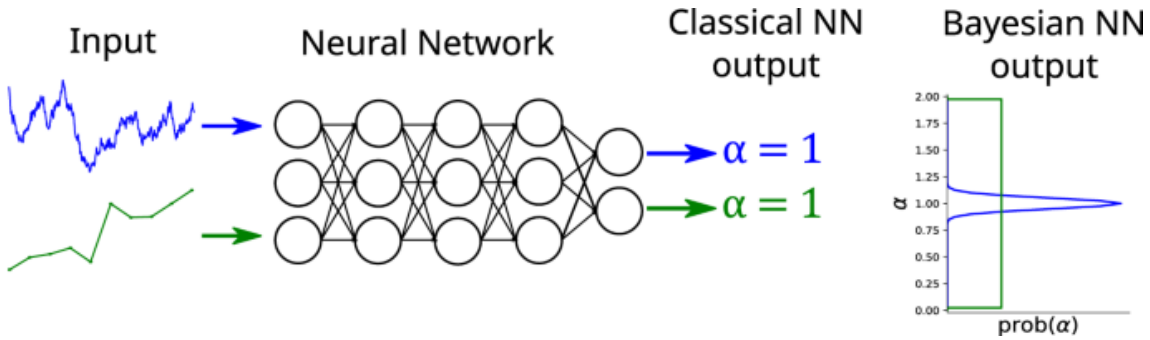
When is a point prediction not enough?

- Medical diagnosis.
- Autonomous driving.
- LLMs (hallucinations).
- Spatio temporal forecasting (uncertainty propagates over time).

In all of these cases (and many more), we don't just want a prediction. We want to know **how certain that prediction is**.

Bayesian Deep Learning (BDL)

- "Bayesian deep learning **integrates deep neural networks with Bayesian inference**. Unlike traditional deep learning, which produces deterministic predictions, Bayesian deep learning **treats model parameters as random variables** with associated probability distributions. This **allows the model to capture and quantify uncertainty** inherent in data, leading to more robust and reliable predictions."



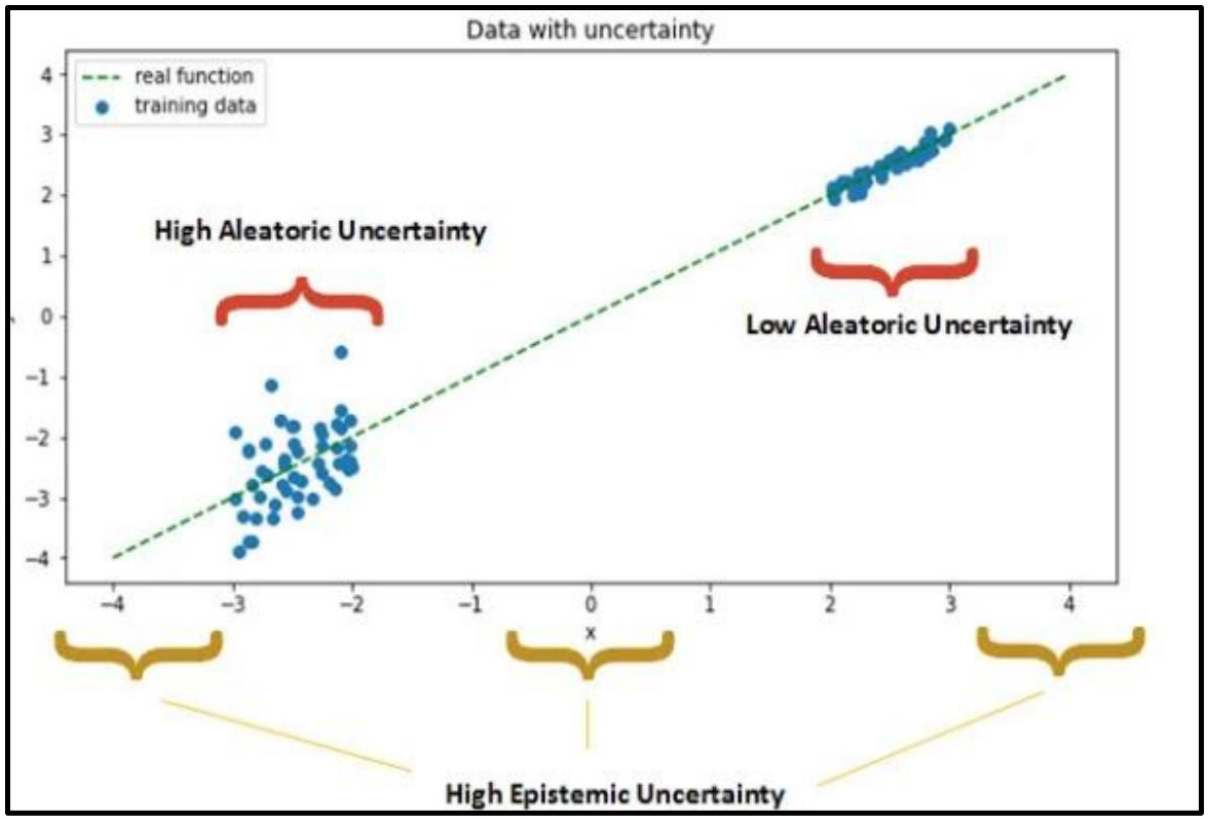
https://medium.com/@pinakdatta/bayesian-deep-learning-understanding-probabilistic-models-uncertainty-3f0602ee4189
https://www.nature.com/articles/s41467-022-34305-6

Type of Uncertainty

- Epistemic Uncertainty:
  - Due to limited data.
  - Reducible with more data.

- Aleatoric Uncertainty:
  - Inherent noise in the data.
  - Irreducible.
  - Measured by e.g. output of neural network are parameters of a Gaussian.

Bayesian deep learning is primarily concerned with epistemic uncertainty. This is exactly the uncertainty that classical deep learning ignores.



A Structured Review of Literature on Uncertainty in Machine Learning & Deep Learning (https://arxiv.org/pdf/2406.00332)
https://medium.com/%40pinakdatta/bayesian-deep-learning-understanding-probabilistic-models-uncertainty-3f0602ee4189

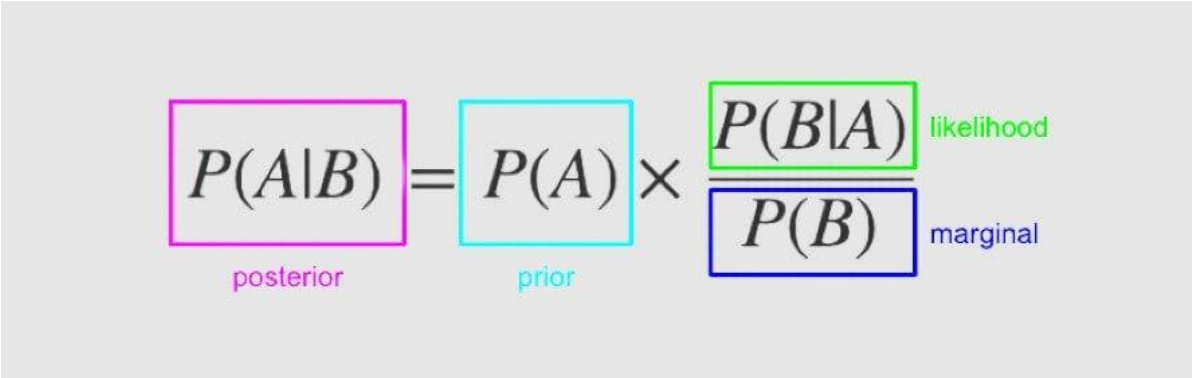
Neural Network Parameters

We have some data $\mathcal{D} = \{X, Y\}$. We want to find the posterior distribution of our neural network parameters θ.

$$p(\theta|\mathcal{D}) = \frac{p(Y|\theta, X)p(\theta)}{p(\mathcal{D})}$$

Our ultimate goal is estimating the posterior predictive distribution for a new input x^*.

$$\mathrm{p}(y^*|x^*, \mathcal{D})$$



pub.aimind.so/mastering-probability-exploring-joint-marginal-conditional-probability-and-bayes-theorem-b4780b20fd4b?gi=fc45868cf067

Estimating the Posterior

Standard deep learning only finds a **MAP** of the posterior. To better estimate the posterior, we can use:

1. Sampling-based approximations (SG-MCMC).
2. Laplace approximations.
3. Variational Inference.
4. Ensemble methods.

Due to the scale of the models, **no method gives exact samples**.

- MCMC is exact only asymptotically (computationally infeasible for deep learning).
- SG-MCMC is an approximation of MCMC.
- Laplace approximations, VI are considered approximate Bayesian inference.
- Ensemble methods are "non-Bayesian", but in practice they provide better approximations than traditional approximate Bayesian methods.

SG-MCMC

- We usually can't use simple sampling techniques (like rejection sampling, MCMC) due to the high dimensionality of the parameters and the size of the dataset.
- We can use Stochastic Gradient MCMC (SG-MCMC), a class of methods that **uses stochastic gradients to make MCMC suitable for deep learning models**.
- Compared to mini-batch GD, an additional noise term is introduced.

Algorithm 1: SGLD

Input: $\boldsymbol{\theta}_0$, $\{h_0, \ldots, h_K\}$.
for $k \in 1, \ldots, K$ **do**
  Draw $\mathcal{S}_n \subset \{1, \ldots, N\}$ without replacement
  Estimate $\hat{\nabla} U(\boldsymbol{\theta})^{(n)}$ using (4)
  Draw $\xi_k \sim N(0, h_k I)$
  Update $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k - \frac{h_k}{2} \hat{\nabla} U(\boldsymbol{\theta}_k)^{(n)} + \xi_k$
end

$$\hat{\nabla} U(\boldsymbol{\theta})^{(n)} = \frac{N}{n} \sum_{i \in \mathcal{S}_n} \nabla U_i(\boldsymbol{\theta}), \tag{4}$$

BDL Book, Chapter 2
Stochastic gradient Markov chain Monte Carlo (https://arxiv.org/pdf/1907.06986)

Laplace Approximations

Using sampling methods has huge memory requirements.

- Estimate the posterior by a Gaussian distribution.

$$p(\theta|\mathcal{D}) \approx \mathcal{N}(\theta_*, \Psi^{-1}(\theta_*))$$

- If we have a trained deep learning model, we can just use its parameters as θ_*.
- We can compute the Hessian using autodiff **for small models**.
- For large models, we need to approximate it with e.g. a diagonal matrix (next week).
- The memory and compute costs, and quality of the approximation of the Hessian are controlled by the user.

- We can do the Laplace approximation post hoc (if we have access to the training data)!
- Simple way to fix some nontrivial pathologies of standard deep learning.

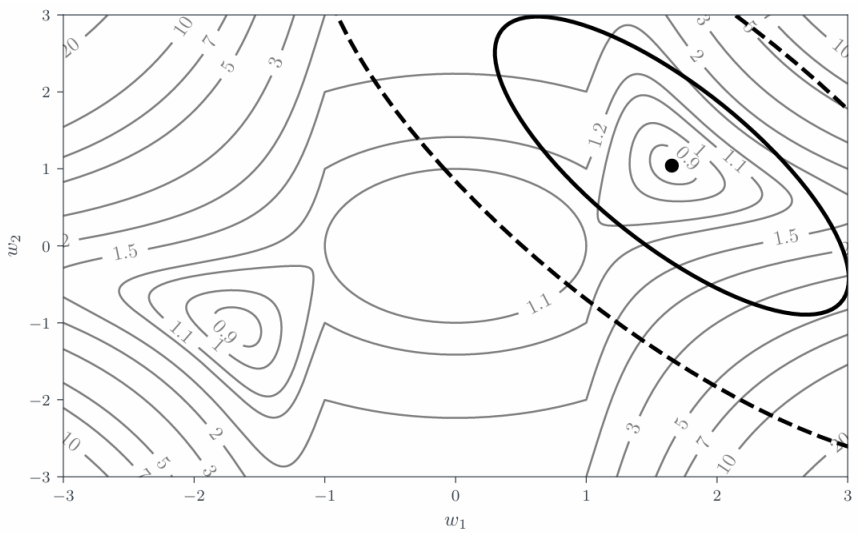


Figure 7.1 A linearized Laplace approximation on the most basic feed-forward perceptron $f(x, \theta) = \theta_2 \cdot \text{ReLU}(\theta_1 x + 1)$ for the dataset $X = (1, -1), Y = (-1, -1)$, with regularizer $\Omega(\theta) = 0.2 \cdot \|\theta\|^2$ and square loss. The gray contour lines denote the Energy function. The black dot represents the minimum θ_*; the thick black ellipses denote one and two standard deviations of the Laplace approximation.

Variational Inference

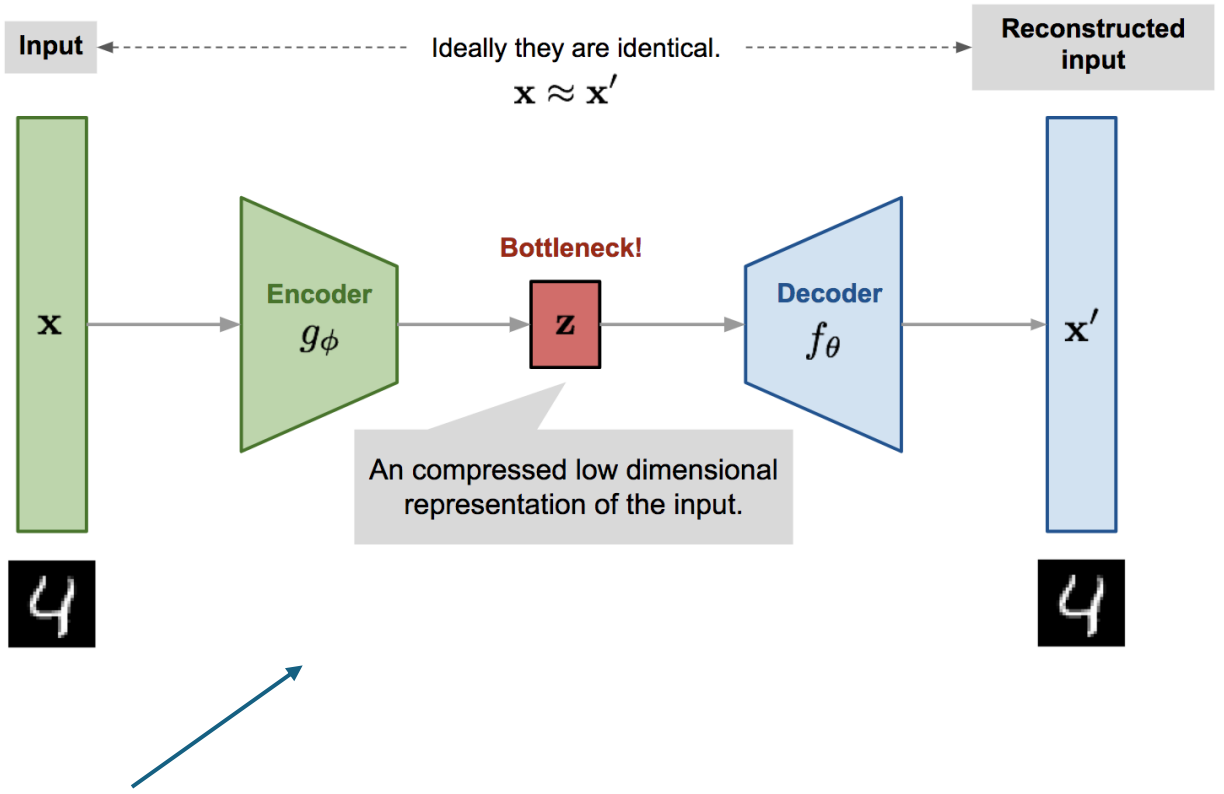
- VI reframes the Bayesian inference as an **optimization problem**.
- We introduce a variational family of distributions (user defined hyperparameter, simplest and most common case is a normal distribution) $q(\theta;\lambda)$ and find the "best" choice of the parameters λ.

$$\lambda^* = \text{argmin}_\lambda \, D(q(\theta;\lambda)||p(\theta|\mathcal{D}))$$

D is a divergence (similar to a distance but is not symmetric). An example is the KL-Divergence.

A simple choice of q is:

$$q(\theta;\lambda) = \prod \mathcal{N}(\theta_i|\mu_i,\sigma_i^2)$$



VAEs (not BDL). They are trained with VI.

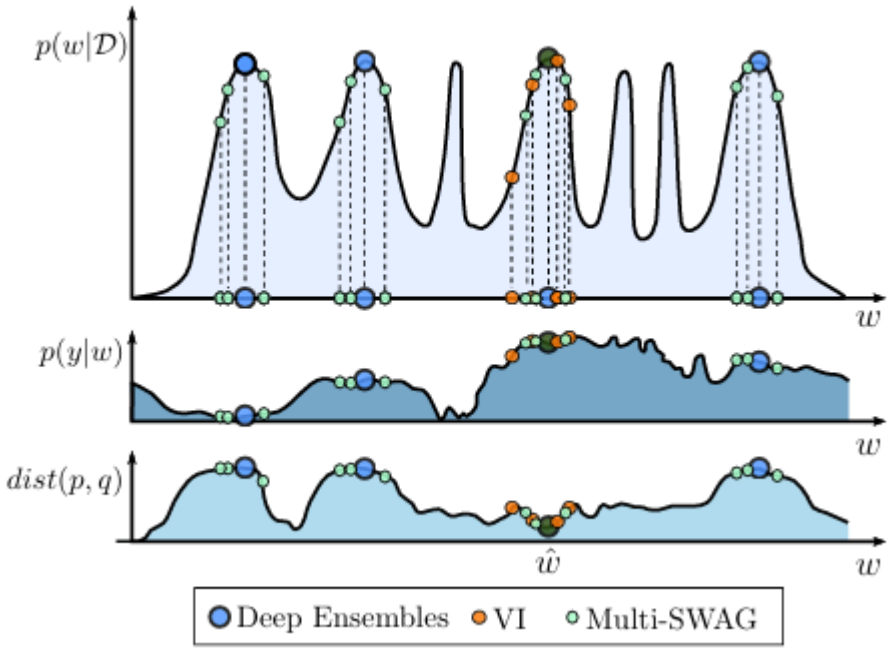
BDL Book, Chapter 12

https://lilianweng.github.io/posts/2018-08-12-vae/

Deep Ensembles

- Retrain a neural network with different random initializations and ensemble the results.
- Typically works better than standard approximate Bayesian methods.
- If we use K ensembles, then we have an additional K-fold training and testing cost.

- *What if instead we use the K-fold increase in computation to train a larger neural network?*
  - ➢ It seems that often, the larger neural network will yield better accuracy.



Last Layer Inference

- Having a posterior distribution over each parameter is very expensive, especially for large networks.
- Instead treat all parameters as being deterministic, **except the ones in the last layer**.
- This can be interpreted as performing Bayesian linear regression with covariates the output of a neural network (all but the last layer).

- So much cheaper during inference!!! We only pass through the model once (except the last layer) and perform multiple passes only through the last layer!

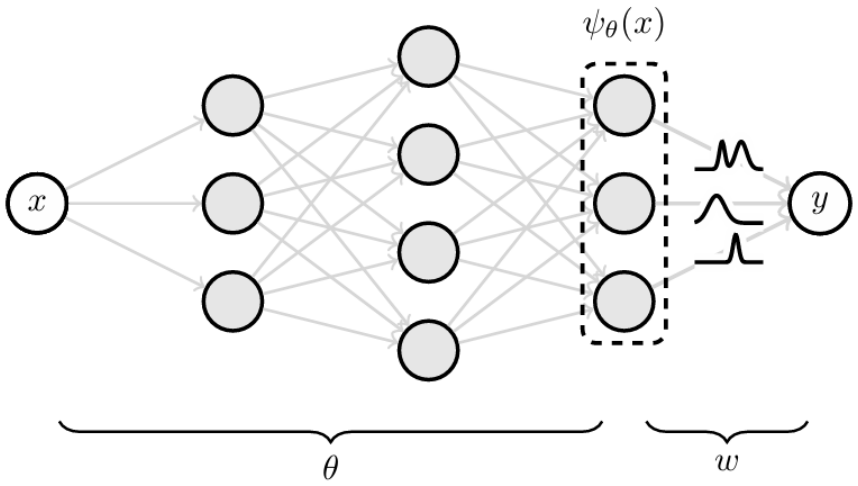
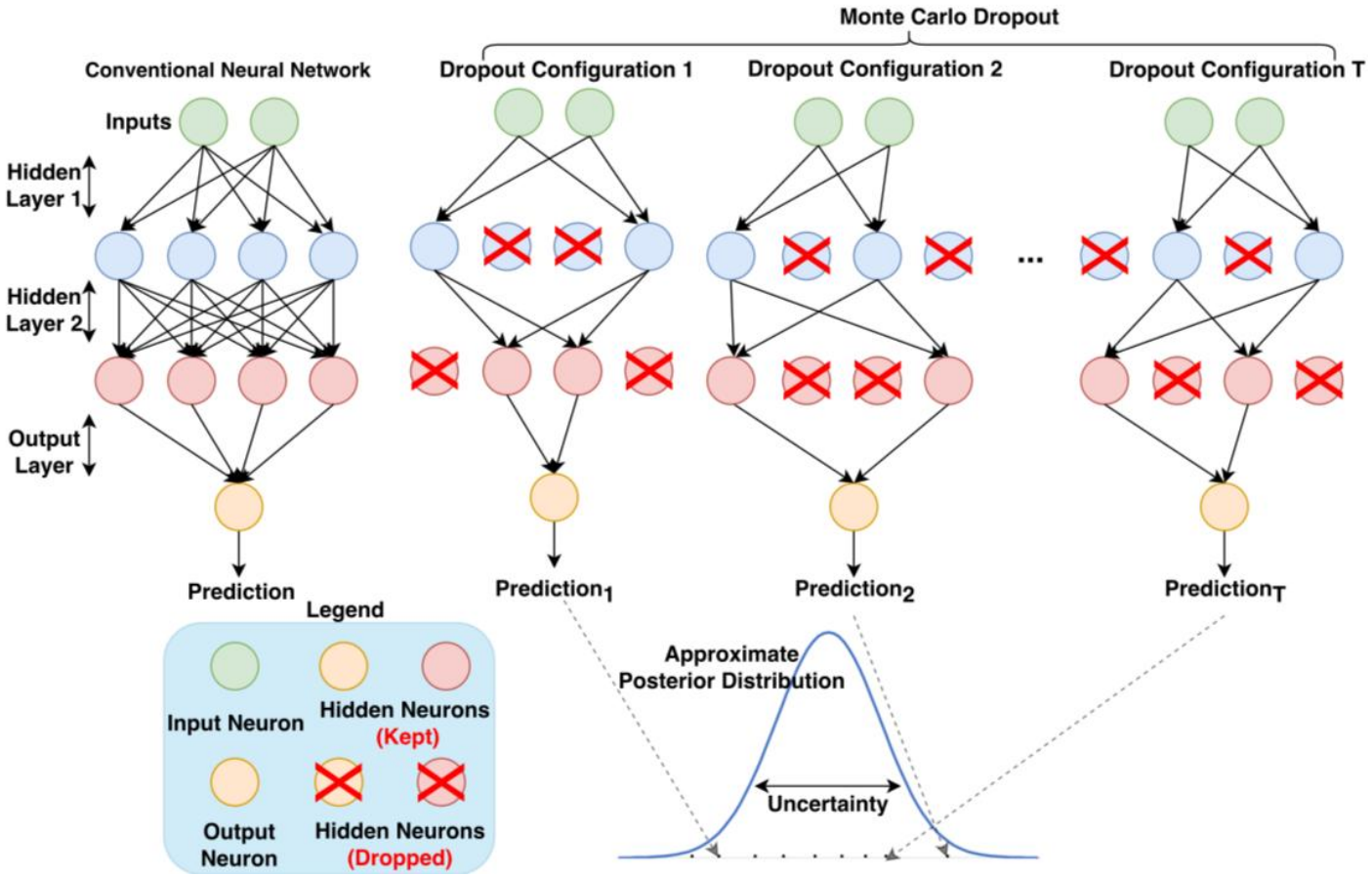


Figure 39.1 In last-layer inference, the parameters $\theta \in \mathbb{R}^{P-H_{L-1}}$ up to and including the penultimate layer are treated as deterministic. Only the posterior $p(w \mid \mathcal{D})$ for the last-layer weights is computed using the deep feature projection ψ_θ. Figure inspired by **blundell2015weight**.

MC Dropout

- Train a model with dropout (or use a pretrained one).
- Perform multiple passes during inference **with dropout**.



Gaussian Processes

- GPs are not deep learning (yet).
- GPs extend the notion of a Normal distribution to functions (infinite dimension).
- In practice, we often set $m(x) = 0$.
- The kernel is a measure of similarity, e.g. RBF kernel.

$$k(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right)$$

Definition 23.3 (Gaussian Process [**Rasmussen2006Gaussian**]). *Let $f : \mathcal{X} \to \mathbb{R}$ be a real-valued function defined on an input space $\mathcal{X}$. We say that f is a Gaussian Process, denoted*

$$f \sim \mathcal{GP}(m(\cdot), k(\cdot, \cdot)),$$

if for any finite collection of inputs $\mathbf{x} = [x_1, \ldots, x_n]^\top$, the random vector of function evaluations

$$\mathbf{f} = [f(x_1), \ldots, f(x_n)]^\top$$

follows a multivariate normal distribution

$$\mathbf{f} \sim \mathcal{N}(\boldsymbol{\mu}, K_{\mathbf{xx}}),$$

with mean $\boldsymbol{\mu}_i = m(x_i)$ and covariance $[K_{\mathbf{xx}}]_{ij} = k(x_i, x_j)$.

Prior

"The user's choice of kernel amounts to specifying a prior over function behaviour and determining what kinds of variation are expected in unseen regions of the input space."

Gaussian Processes for Regression

Given training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{P}$, we assume that observations are noisy evaluations of an underlying latent function f:

$$y_i = f(x_i) + \varepsilon_i, \qquad \varepsilon_i \sim \mathcal{N}(0, \sigma^2),$$

where σ^2 denotes the noise variance.

Let $\mathbf{y} = [y_1, \ldots, y_n]^\top$ be the observed targets, and consider a new input x_* with latent value $f_* = f(x_*)$. The joint prior over training and test outputs is

$$\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ 0 \end{bmatrix}, \begin{bmatrix} K_{\mathbf{xx}} + \sigma^2 I & K_{\mathbf{x}x_*} \\ K_{x_*\mathbf{x}} & K_{x_*x_*} \end{bmatrix} \right),$$

where $K_{\mathbf{x}x_*} = [k(x_1, x_*), \ldots, k(x_n, x_*)]^\top$.

Conditioning on the observed data gives the posterior distribution:

$$p(f_*|\mathbf{x}, \mathbf{y}, x_*) = \mathcal{N}(\mu_*, \sigma_*^2),$$

with

$$\mu_* = K_{x_*\mathbf{x}} \left[K_{\mathbf{xx}} + \sigma^2 I\right]^{-1} \mathbf{y}, \qquad \sigma_*^2 = k(x_*, x_*) - K_{x_*\mathbf{x}} \left[K_{\mathbf{xx}} + \sigma^2 I\right]^{-1} K_{\mathbf{x}x_*}.$$

- μ_* is just a linear combination of the observed targets.

GPs and Deep Learning

- Standard GPs use fixed, pre-defined kernels which operate directly on raw input features. This has the following downsides.
  - Uninformative raw features: For data like images, text or graphs, standard kernels fail to capture the dependencies that underlie the data.
  - High-dimensional inputs: In high-dimensions pairwise distances tend to concentrate, making common kernels nearly constant.
  - Limited adaptability: Once we choose a kernel (and learn its parameters) its inductive bias becomes static. “This lack of flexibility restricts their expressiveness in modern representation learning settings, where adaptive, data-driven feature extraction is essential.”

- **Deep Gaussians Processes** (small number of parameters compared to neural networks): Hierarchical composition of GPs. The output of one GP feeds into another.

- **Deep Kernel Learning**: Use a neural network to learn the kernel.

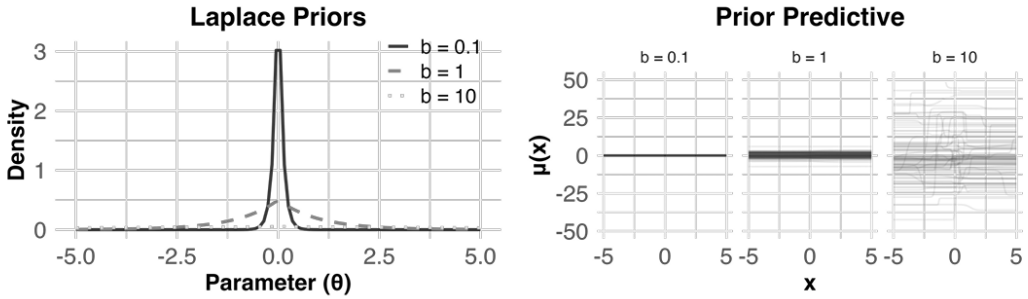
Note: When a network’s width tends to infinity, its outputs converge to a GP whose kernel is determined by the network architecture (see Neural Tangent Kernel, NIPS 2018, paper with ~5k citations).

BDL Book, Chapter 23

Neural Tangent Kernel: Convergence and Generalization in Neural Networks

Priors for BDL

- In traditional statistics, a prior allows us to represent our initial beliefs before observing any data. It allows us to integrate expert knowledge in our models.
- In deep learning, they usually serve a different purpose. Due to the complexity of the models, it is hard to have any "expert knowledge" on the distribution of the parameters.
- Priors are often selected for practical purposes such as to prevent overfitting.



Informative Priors

- Rare due to model complexity.
- Sometimes applied to specific parameters.
  - Bias term of the output layer.
  - Spike-and-slab priors for the weights associated with covariates in the input layer.

$$p(\theta_{ij}^{(1)} \mid \gamma_i) = \gamma_i \mathcal{N}(\theta_{ij}^{(1)}; \mu_i, \sigma_i^2) + (1 - \gamma_i)\delta_0(\theta_{ij}^{(1)}), \quad \gamma_i \sim \text{Bernoulli}(\rho_i)$$

  - Example: If we are predicting heart disease risk, we might assign a high inclusion probability to the weight associated with physical activity and a low one with the weight associated with eye color.

Uninformative Priors?

- Uninformative priors aim to minimize subjective influence. I common choice in classical statistics is a uniform over an interval $[a, b]$.
- **Marginally uniform priors do not always imply an uninformative joint prior especially in high-dimensional BDL models.**
- Reference priors (such as Jeffrey's priors) are designed to maximize the difference between prior and posterior.
- Due to their computational complexity, we can't use them in BDL models!
- In practice, **uninformative priors are rarely used in BDL**!

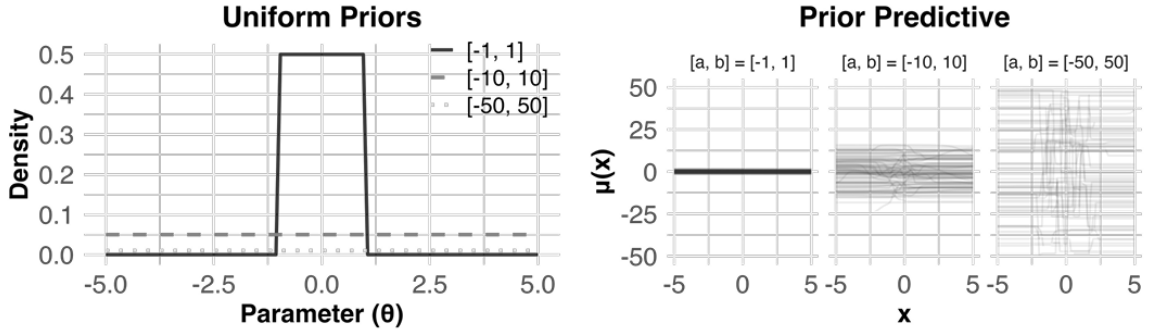


Figure 28.3 Uniform priors for single parameters (left) and the corresponding prior predictive distributions (right), showing a wide range of possible BDL functions with concentration effects for narrow intervals and non-uniform prior predictive.

Regularization Priors

- They constrain model complexity to prevent overfitting.
- The zero-centered isotropic Gaussian prior corresponds to ℓ_2 regularization in MAP estimation.

$$p(\boldsymbol{\theta}) = \prod_{i=1}^{P} \mathcal{N}(\theta_i; 0, \sigma^2)$$

- The Laplace prior corresponds to ℓ_1 regularization in MAP estimation (encourage parameters to be zero or near-zero).

$$p(\boldsymbol{\theta} \mid \mathcal{M}) = \prod_{i=1}^{P} \text{Laplace}(\theta_i; 0, b)$$

Priors

- Prior predictive behavior depends on the architecture of the model, particularly the **activation functions**.
- It is advised to conduct prior predictive checks on simplified versions of the model, to avoid prior misspecification.

- Figure shows the prior predictive distribution for Gaussian priors.

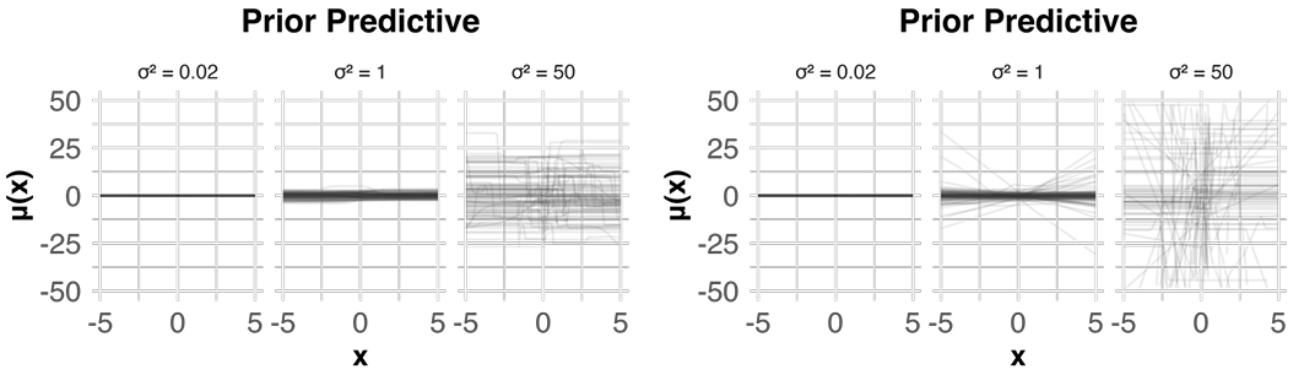


Figure 28.5 Prior predictive distributions for tanh (left) and ReLU (right) activations.

Bayesian LLMs

- LLMs are amazing but have some noticeable limitations.
- The most common of them is known as “hallucinations”, i.e. **confidently** generating incorrect outputs.
- If we had Bayesian LLMs, we would be able to quantify uncertainty. Prediction with model averaging should be more reliable (biases of single models are averaged out).
- Standard Bayesian methods do not work!
  - For training MCMC struggles with slow mixing, VI has high gradient variance due to the dimensionality of the posterior.
  - For inference BMA is very slow.
  - The motivation for being Bayesian is unclear; we have huge training datasets, so the epistemic uncertainty should be low, newer LLMs can defer to external search engines.
- Early works of Bayesian LLMs focused on fine-tuning, where the datasets are small.

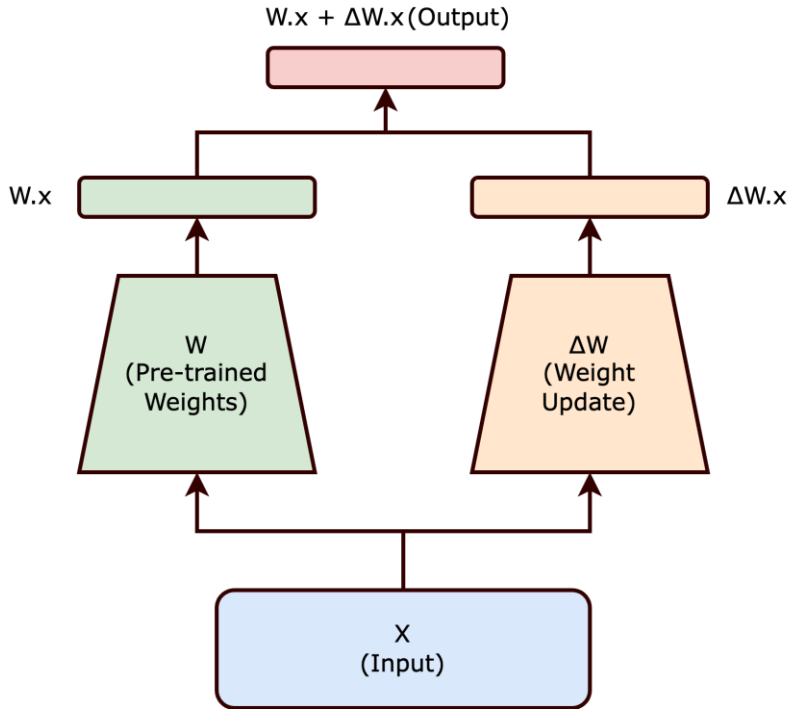
BDL Book, Chapter 43

https://www.linkedin.com/pulse/unlocking-enterprise-potential-large-language-models-llms-matt-moalem-fyvzc/

Making LoRA Bayesian

- LoRA: Freeze base model weights but add a low rank matrix (which we train).
- $W' = W + \Delta W = W + AB, \quad A \in \mathbb{R}^{d \times r},\ B \in \mathbb{R}^{r \times k},\ r \ll \min(d, k)$

- LoRA ensemble: Learn multiple ΔW and ensemble the result (either by multiple forward passes or by setting $\Delta W = \sum_{i=1}^{N} \lambda_i \Delta W_i$.
- Bayesian LoRA: Treat LoRA parameters as random variables and learn a posterior distribution over them.



LoRA: Low-Rank Adaptation of Large Language Models, *ICLR 2022*

LoRA ensembles for large language model fine-tuning

Bayesian Low-rank Adaptation for Large Language Models, ICLR 2024

https://towardsdatascience.com/understanding-lora-low-rank-adaptation-for-finetuning-large-models-936bce1a07c6/

Improved Variational Online Newton

- Uses VI – places a Normal variational approximation over each parameter (so we have to learn double the parameters).
- The computational costs (for training) are nearly identical to Adam, but its predictive uncertainty is better.
- We want to minimize the following objective:

$$\mathcal{L}(q) = \lambda \mathbb{E}_{q(\boldsymbol{\theta})}\left[\bar{\ell}(\boldsymbol{\theta})\right] + \mathbb{D}_{\mathrm{KL}}(q(\boldsymbol{\theta}) \,\|\, p(\boldsymbol{\theta}))$$

- **Natural gradients** perform gradient descent using the geometry of probability distributions, measuring distances with KL divergence rather than Euclidean distance. They are computed by **preconditioning the ordinary gradient with the inverse Fisher information matrix.**

$$\nabla^{\mathrm{nat}} = F^{-1}\nabla$$

Algorithm 1 Improved Variational Online Newton (IVON). Hyperparameter setting is described in App. A.

Require: Learning rates $\{\alpha_t\}$, weight-decay $\delta > 0$.
Require: Momentum parameters $\beta_1, \beta_2 \in [0, 1)$.
Require: Hessian init $h_0 > 0$.
Init: $\mathbf{m} \leftarrow$ (NN-weights), $\mathbf{h} \leftarrow h_0$, $\mathbf{g} \leftarrow 0$, $\lambda \leftarrow N$.
Init: $\boldsymbol{\sigma} \leftarrow 1/\sqrt{\lambda(\mathbf{h} + \delta)}$.
Optional: $\alpha_t \leftarrow (h_0 + \delta)\alpha_t$ for all t.

for $t = 1, 2, \ldots$ **do**
  $\widehat{\mathbf{g}} \leftarrow \widehat{\nabla}\bar{\ell}(\boldsymbol{\theta})$, where $\boldsymbol{\theta} \sim q$
  $\widehat{\mathbf{h}} \leftarrow \widehat{\mathbf{g}} \cdot (\boldsymbol{\theta} - \mathbf{m})/\boldsymbol{\sigma}^2$
  $\mathbf{g} \leftarrow \beta_1 \mathbf{g} + (1-\beta_1)\widehat{\mathbf{g}}$
  $\mathbf{h} \leftarrow \beta_2 \mathbf{h} + (1-\beta_2)\widehat{\mathbf{h}} + \frac{1}{2}(1-\beta_2)^2(\mathbf{h} - \widehat{\mathbf{h}})^2/(\mathbf{h} + \delta)$
  $\bar{\mathbf{g}} \leftarrow \mathbf{g}/(1-\beta_1^t)$
  $\mathbf{m} \leftarrow \mathbf{m} - \alpha_t(\bar{\mathbf{g}} + \delta\mathbf{m})/(\mathbf{h} + \delta)$
  $\boldsymbol{\sigma} \leftarrow 1/\sqrt{\lambda(\mathbf{h} + \delta)}$
end for
return $\mathbf{m}, \boldsymbol{\sigma}$

Improved Variational Online Newton

- IVON estimates curvature using a reparameterization-based Gauss–Newton approximation, which enables efficient natural-gradient updates!
- IVON@mean uses the mean of the weights.
- IVON uses a model average with 64 samples drawn from the approximation of the posterior.
- Still, lots of disadvantages:
  - Posterior inference requires multiple forward passes.
  - More hyperparameters to tune.

Dataset / Model	Method	Top-1 Acc. ↑	Top-5 Acc. ↑	NLL ↓	ECE ↓	Brier ↓
ImageNet ResNet-50 (26M params)	AdamW	(2%) $75.16_{\pm 0.14}$	$92.37_{\pm 0.03}$	$1.018_{\pm 0.003}$	$0.066_{\pm 0.002}$	$0.349_{\pm 0.002}$
	SGD	(1%) $76.63_{\pm 0.45}$	$93.21_{\pm 0.25}$	$0.917_{\pm 0.026}$	$0.038_{\pm 0.009}$	$0.326_{\pm 0.006}$
	IVON@mean	$77.30_{\pm 0.08}$	$93.58_{\pm 0.05}$	$0.884_{\pm 0.002}$	$0.035_{\pm 0.002}$	$\mathbf{0.316}_{\pm 0.001}$
	IVON	$\mathbf{77.46}_{\pm 0.07}$	$\mathbf{93.68}_{\pm 0.04}$	$\mathbf{0.869}_{\pm 0.002}$	$\mathbf{0.022}_{\pm 0.002}$	$\mathbf{0.315}_{\pm 0.001}$
TinyImageNet ResNet-18 (11M params)	AdamW	(15%) $47.33_{\pm 0.90}$	$71.54_{\pm 0.95}$	$6.823_{\pm 0.235}$	$0.421_{\pm 0.008}$	$0.913_{\pm 0.018}$
	SGD	(1%) $61.39_{\pm 0.18}$	$82.30_{\pm 0.22}$	$1.811_{\pm 0.010}$	$0.138_{\pm 0.002}$	$0.536_{\pm 0.002}$
	IVON@mean	$\mathbf{62.41}_{\pm 0.15}$	$\mathbf{83.77}_{\pm 0.18}$	$1.776_{\pm 0.018}$	$0.150_{\pm 0.005}$	$0.532_{\pm 0.002}$
	IVON	$\mathbf{62.68}_{\pm 0.16}$	$\mathbf{84.12}_{\pm 0.24}$	$\mathbf{1.528}_{\pm 0.010}$	$\mathbf{0.019}_{\pm 0.004}$	$\mathbf{0.491}_{\pm 0.001}$
TinyImageNet PreResNet-110 (4M params)	AdamW	(11%) $50.65_{\pm 0.0^*}$	$74.94_{\pm 0.0^*}$	$4.487_{\pm 0.0^*}$	$0.357_{\pm 0.0^*}$	$0.812_{\pm 0.0^*}$
	SGD	(2%) $59.39_{\pm 0.50}$	$81.34_{\pm 0.30}$	$2.040_{\pm 0.040}$	$0.176_{\pm 0.006}$	$0.577_{\pm 0.007}$
	IVON@mean	$\mathbf{60.85}_{\pm 0.39}$	$\mathbf{83.89}_{\pm 0.14}$	$1.584_{\pm 0.009}$	$0.053_{\pm 0.002}$	$\mathbf{0.514}_{\pm 0.003}$
	IVON	$\mathbf{61.25}_{\pm 0.48}$	$\mathbf{84.13}_{\pm 0.17}$	$\mathbf{1.550}_{\pm 0.009}$	$\mathbf{0.049}_{\pm 0.002}$	$\mathbf{0.511}_{\pm 0.003}$
CIFAR-100 ResNet-18 (11M params)	AdamW	(11%) $64.12_{\pm 0.43}$	$86.85_{\pm 0.51}$	$3.357_{\pm 0.071}$	$0.278_{\pm 0.005}$	$0.615_{\pm 0.008}$
	SGD	(1%) $74.46_{\pm 0.17}$	$92.66_{\pm 0.06}$	$1.083_{\pm 0.007}$	$0.113_{\pm 0.001}$	$0.376_{\pm 0.001}$
	IVON@mean	$74.51_{\pm 0.24}$	$92.74_{\pm 0.19}$	$1.284_{\pm 0.013}$	$0.152_{\pm 0.003}$	$0.399_{\pm 0.002}$
	IVON	$\mathbf{75.14}_{\pm 0.34}$	$\mathbf{93.30}_{\pm 0.19}$	$\mathbf{0.912}_{\pm 0.009}$	$\mathbf{0.021}_{\pm 0.003}$	$\mathbf{0.344}_{\pm 0.003}$
CIFAR-100 PreResNet-110 (4M params)	AdamW	(10%) $65.88_{\pm 0.84}$	$88.34_{\pm 0.56}$	$2.893_{\pm 0.088}$	$0.258_{\pm 0.006}$	$0.578_{\pm 0.014}$
	SGD	(2%) $74.19_{\pm 0.11}$	$92.41_{\pm 0.14}$	$1.204_{\pm 0.012}$	$0.137_{\pm 0.002}$	$0.393_{\pm 0.004}$
	IVON@mean	$75.23_{\pm 0.23}$	$93.45_{\pm 0.16}$	$1.149_{\pm 0.010}$	$0.136_{\pm 0.002}$	$0.380_{\pm 0.003}$
	IVON	$\mathbf{75.81}_{\pm 0.18}$	$\mathbf{93.93}_{\pm 0.19}$	$\mathbf{0.884}_{\pm 0.007}$	$\mathbf{0.030}_{\pm 0.003}$	$\mathbf{0.336}_{\pm 0.001}$

Table 1: IVON improves both accuracy and uncertainty over SGD and AdamW. Improvements in accuracy by IVON are shown in red. The performance of AdamW is not good on the smaller datasets likely due to overfitting when training for 200 epochs. IVON does not have this issue. Additional results are in the appendix in Tables 9 to 11 and Table 13.

Variational Learning is Effective for Large Deep Networks, ICML 2024

BDL in Practice

- In practice when would you use something like IVON?
- Medical image diagnosis (image classification).
- For each parameter we sample perform a forward pass.
- Build a credible interval to account for uncertainty!

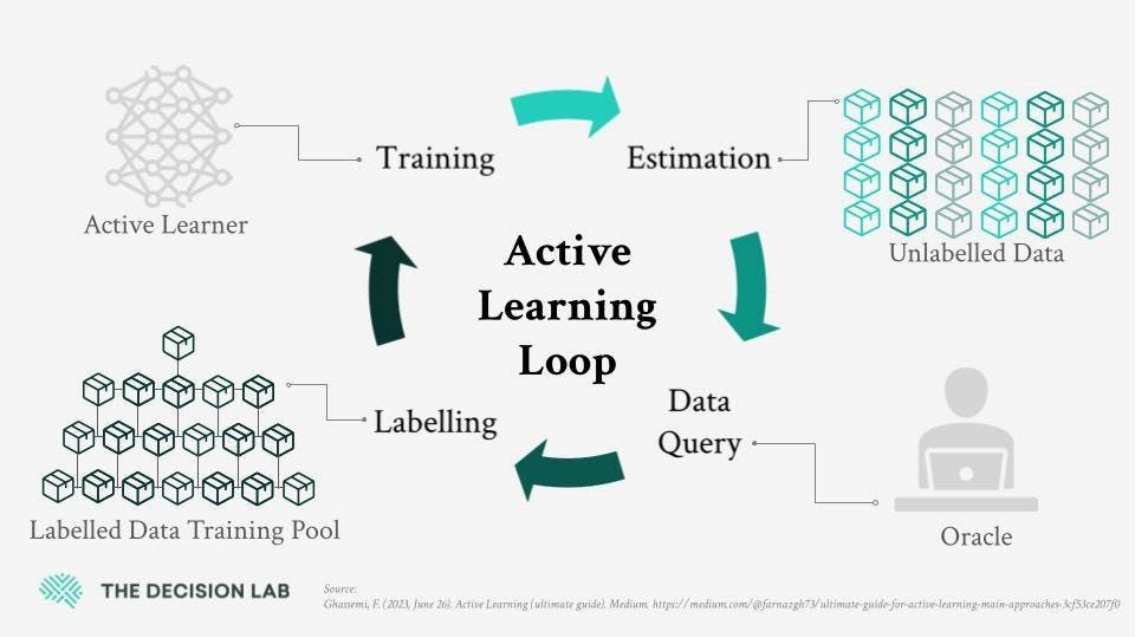
DO NOT USE BDL IF YOU DO NOT CARE ABOUT UNCERTAINTY.

Variational Learning is Effective for Large Deep Networks, ICML 2024

Active Learning

- **Which data do we want to acquire?**
- **Key Idea**: Labeled data are expensive to obtain. How do we choose which data points we want labeled?

- Toy active learning example:
  - Train a small model on a **labeled** dataset.
  - Pass a large pool of **unlabeled** data through the trained model.
  - Choose the most informative unlabeled data and label them (e.g., output probabilities close to 0.5).
  - Retrain model.



BDL Book, Chapter 53

https://thedecisionlab.com/reference-guide/computer-science/active-learning

Thank you for listening! Questions?

Lancaster University

Prob_AI

Stable Diffusion 3 (uses Flow model)