

# Bayesian Neural Networks

Variational Inference and Dropout  
(MAT8701 Advanced statistical methods  
in inference and learning)

**Kristian Gundersen**

UNIVERSITY OF BERGEN



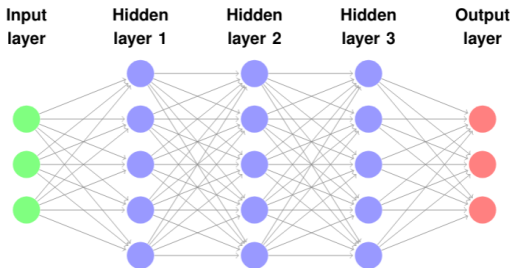
# Table of contents

- 1 Deep Learning
- 2 UQ in Deep Learning
- 3 Bayesian Neural Networks (BNN)
- 4 BNNs approximated with Variational Inference
- 5 Dropout



# Deep Learning

- Model:  $\hat{\mathbf{y}}^{(i)} = \mathbf{W}_2(\sigma(\mathbf{W}_1\mathbf{x}^{(i)} + \mathbf{b})) := \mathbf{f}^\omega$ ,  $\omega = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}\}$
- Input:  $\mathbf{x}^{(i)}$  ( $\mathbf{X}$  a data set of  $\mathbf{x}^{(i)}$ )
- Output:  $\hat{\mathbf{y}}^{(i)}$  ( $\mathbf{Y}$  a data set of  $\hat{\mathbf{y}}^{(i)}$ )
- Activation function:  $\sigma$



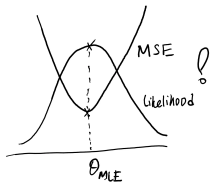
# Objective functions and optimization

## ■ Classification - Cross entropy

$$\mathcal{C}_1^{\mathbf{w}_1, \mathbf{w}_2, \mathbf{b}}(\mathbf{X}, \mathbf{Y}) = -\frac{1}{N} \sum_{i=1}^N \sum_{j=1}^K \mathbf{y}_j^{(i)} \log(\hat{\mathbf{y}}_j^{(i)}) = -\log p(\mathbf{Y}|\mathbf{f}^\omega(\mathbf{X})), \quad (1)$$

## ■ Regression - Mean Squared Error

$$\mathcal{C}_2^{\mathbf{w}_1, \mathbf{w}_2, \mathbf{b}}(\mathbf{X}, \mathbf{Y}) = \frac{1}{N} \sum_{i=1}^N (\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)})^2 = -\log p(\mathbf{Y}|\mathbf{f}^\omega(\mathbf{X})), \quad (2)$$

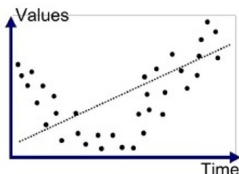


Minimizing both these loss functions maximize the likelihood of the parameters of the model. **Learning/training:** Which weights and biases are minimizing a certain loss function

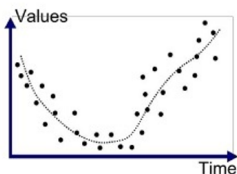


# Regularization

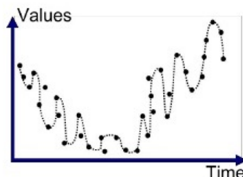
Deep learning models are prone to overfitting.



Underfitted



Good Fit/Robust



Overfitted

## L2 - regularisation

$$\mathcal{L}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) = \mathcal{C}^{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}}(\mathbf{X}, \mathbf{Y}) + \lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2 \quad (3)$$

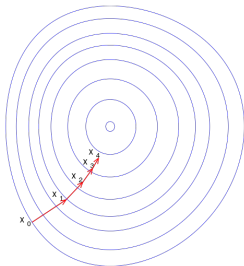
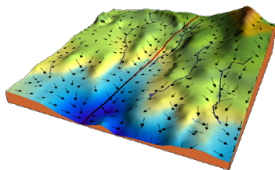
Other regularization techniques: **L1 - regularization**, **Dropout**



# Optimization of the model

- Stochastic gradient decent and mini batch optimization
- Forwardpropagation
- Backpropagation

$$\boldsymbol{\omega}_t = \boldsymbol{\omega}_{t-1} - \frac{\eta}{n} \sum_{i=1}^n \nabla \mathcal{L}_i(\boldsymbol{\omega}_{t-1}) \quad (4)$$



# UQ in Deep learning

*"Uncertainty quantification (UQ) is the science of quantitative characterization and reduction of uncertainties in both computational and real world applications. It tries to determine how likely certain outcomes are if some aspects of the system are not exactly known."* wiki

## ■ Delta method

- Classical method for uncertainty quantification in statistical models
- Taylor expansions of the objective function
- Needs calculation of the inverse of the hessian matrix (expensive for large models)

## ■ Deep ensembles

- Many models are trained with different initialization of the weights with min-batch optimization
- Assessing the variance over model ensemble predictions
- The process resembles bagging or bootstrapping.

## ■ Dropout

## ■ Bayesian Neural Networks



# Sources of uncertainty

- Aleatoric uncertainty (data uncertainty)
  - Uncertainty about the observation caused by a noisy data set
- Epistemic uncertainty (model uncertainty)
  - Epistemic uncertainty captures our ignorance about the models most suitable to explain our data.

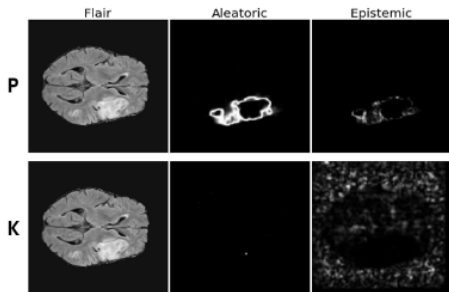


Figure: See also [1, 2]





# Sources of uncertainty

It is possible to calculate the aleatoric and epistemic uncertainty [2]

$$\begin{aligned} \text{Var}_{q_{\hat{\theta}}(y^*|x^*)}(y^*) &= E_{q_{\hat{\theta}}(y^*|x^*)}\{y^{*\otimes 2}\} - E_{q_{\hat{\theta}}(y^*|x^*)}(y^*)^{\otimes 2}, \\ &= \underbrace{\int_{\Omega} [\text{diag}\{E_{p(y^*|x^*,\omega)}(y^*)\} - E_{p(y^*|x^*,\omega)}(y^*)^{\otimes 2}] q_{\hat{\theta}}(\omega) d\omega}_{\text{aleatoric}} \\ &\quad + \underbrace{\int_{\Omega} \{E_{p(y^*|x^*,\omega)}(y^*) - E_{q_{\hat{\theta}}(y^*|x^*)}(y^*)\}^{\otimes 2} q_{\hat{\theta}}(\omega) d\omega}_{\text{epistemic}}, \end{aligned}$$

Approximations can be made [1, 2]

$$\underbrace{\frac{1}{T} \sum_{t=1}^T \text{diag}(\hat{p}_t) - \hat{p}_t^{\otimes 2}}_{\text{aleatoric}} + \underbrace{\frac{1}{T} \sum_{t=1}^T (\hat{p}_t - \bar{p})^{\otimes 2}}_{\text{epistemic}}, \quad \underbrace{\frac{1}{T} \sum_{t=1}^T \text{diag}(\hat{\sigma}_t^2)}_{\text{aleatoric}} + \underbrace{\frac{1}{T} \sum_{t=1}^T (\hat{\mu}_t - \bar{\mu})^{\otimes 2}}_{\text{epistemic}},$$

Figure: See [1] and [2]



# Bayesian statistics

- Bayesian statistics is a theory in the field of statistics based on the Bayesian interpretation of probability
- The degree of belief may be based on prior knowledge about the event
- This is in contrast to the frequentist interpretation, that views probability as the limit of the relative frequency of an event after many trials.
- Bayesian statistical methods use Bayes' theorem to compute and update probabilities after obtaining new data.

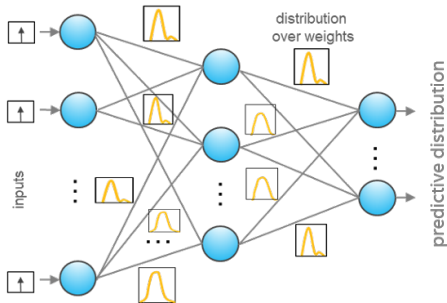
$$p(\omega|\mathbf{X}, \mathbf{Y}) = \frac{p(\omega)p(\mathbf{Y}|\mathbf{X}, \omega)}{p(\mathbf{Y}|\mathbf{X})} \quad (5)$$



# Posterior

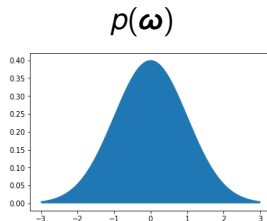
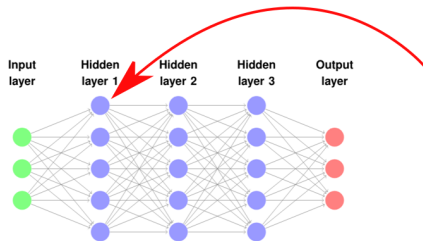
The posterior can be found with Bayes theorem (Figure from [3])

$$p(\omega|\mathbf{X}, \mathbf{Y}) = \frac{p(\omega)p(\mathbf{Y}|\mathbf{X}, \omega)}{p(\mathbf{Y}|\mathbf{X})}$$



# Priors

$$p(\omega|\mathbf{X}, \mathbf{Y}) = \frac{p(\omega) p(\mathbf{Y}|\mathbf{X}, \omega)}{p(\mathbf{Y}|\mathbf{X})}$$

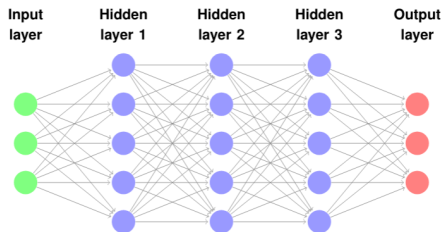


How probable was our hypothesis before observing the evidence?



# Model (likelihood function)

$$p(\omega|\mathbf{X}, \mathbf{Y}) = \frac{p(\omega) \boxed{p(\mathbf{Y}|\mathbf{X}, \omega)}}{p(\mathbf{Y}|\mathbf{X})}$$



How probable is the evidence, given that our hypothesis is true?

$$p(\mathbf{Y}|\mathbf{X}, \omega) = \prod_{i=1}^N p(\mathbf{y}^{(i)}|\mathbf{x}^{(i)}, \omega)$$



# Model evidence (marginal likelihood)

$$p(\omega|\mathbf{X}, \mathbf{Y}) = \frac{p(\omega)p(\mathbf{Y}|\mathbf{X}, \omega)}{p(\mathbf{Y}|\mathbf{X})}$$

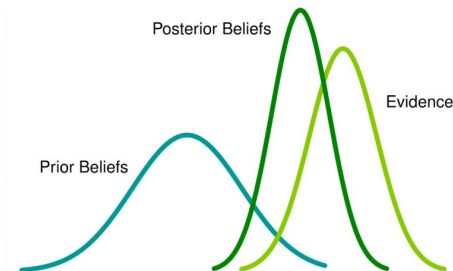


Figure from [4]

How probable is the new evidence under all possible hypothesis?

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \omega)p(\omega)d\omega$$



# Posterior predictive distribution

Given some new data  $\mathbf{x}^*$ , we can obtain a distribution over the output  $\mathbf{y}^*$  by integrating over the weights

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})d\boldsymbol{\omega} \quad (6)$$

The problem is to estimate  $p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})$  and marginalization term



# Approximating the marginal likelihoods

The marginal likelihood is intractable.

$$\mathbb{E}[p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y})] = \int p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega})p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y})d\boldsymbol{\omega} \quad (7)$$

Can be estimated through Monte Carlo Estimation

$$\mathbb{E}[p(\mathbf{y}^*|\mathbf{Y}, \mathbf{X}, \boldsymbol{\omega})] = \frac{1}{J} \sum_{j=1}^J p(\mathbf{y}^*|\mathbf{x}^*, \boldsymbol{\omega}_j), \quad \boldsymbol{\omega}_j \sim p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{Y}) \quad (8)$$

The problem is that this estimator is not good enough with large models...

- MCMC, Gibbs, HMC, etc.
- Variational Inference
- First an example





# Bayesian Linear Regression

## Example

- Assume  $\mathbf{X} \in \mathbb{R}^{N \times D}$  and  $\mathbf{y} \in \mathbb{R}^N$
- Let  $p(y_i | \mathbf{x}_i, \boldsymbol{\omega}) \sim \mathcal{N}(\mathbf{x}_i^T \boldsymbol{\omega}, \sigma_\epsilon^2)$
- Gaussian priors:  $\boldsymbol{\omega} \sim \mathcal{N}(0, \tau^2 \mathbf{I})$
- Assume that  $\sigma_\epsilon^2$  and  $\tau^2$  are known
- We can apply Bayes theorem to figure out which setting of  $\boldsymbol{\omega}$  best describes the data

$$p(\boldsymbol{\omega} | \mathbf{X}, \mathbf{y}) = \frac{p(\boldsymbol{\omega})p(\mathbf{y} | \mathbf{X}, \boldsymbol{\omega})}{p(\mathbf{y} | \mathbf{X})} \quad (9)$$

$$\propto p(\boldsymbol{\omega})p(\mathbf{y} | \mathbf{X}, \boldsymbol{\omega}) \quad (10)$$

$$= p(\boldsymbol{\omega}) \prod_{i=1}^N p(y_i | \mathbf{x}_i, \boldsymbol{\omega}) \quad (11)$$



# Bayesian Linear Regression

## Example

We can compute the log-likelihood

$$\log p(\boldsymbol{\omega}|\mathbf{X}, \mathbf{y}) \propto \log p(\boldsymbol{\omega}) + \sum_{i=1}^N \log p(y_i|\mathbf{x}_i, \boldsymbol{\omega}) \quad (12)$$

$$= \log \mathcal{N}(0, \tau^2 \mathbf{I}) + \sum_{i=1}^N \log \mathcal{N}(\mathbf{x}_i^T \boldsymbol{\omega}, \sigma_\epsilon^2) \quad (13)$$

$$= \frac{1}{2\sigma_\epsilon^2} \left( (\mathbf{y} - \mathbf{X}\boldsymbol{\omega})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\omega}) + \frac{\sigma_\epsilon^2}{\tau^2} \boldsymbol{\omega}^T \boldsymbol{\omega} \right) \quad (14)$$

- Ridge estimate with  $\lambda = \sigma_\epsilon^2 / \tau^2$
- Minimizing the log-likelihood with Gaussian priors  $\rightarrow$  ridge regression where the regularization strength  $\lambda$  is dependent on the variance of  $y_i$  and the precision of  $\boldsymbol{\omega}$



# Bayesian Linear Regression

## Example

- Regularising effect of the prior distribution
- The resulting estimates of  $\omega$  are called the maximum a posteriori (MAP) estimates.
- Marginal likelihood ignored  $p(\mathbf{y}|\mathbf{X})$  because the MAP estimate of  $\omega$  is unaffected by it.
- Point estimate of  $\omega$ .
- Integration over the posterior distribution of  $\omega \rightarrow$  requires evaluation of  $p(\mathbf{y}|\mathbf{X})$ .



# Variational Inference

$p(\omega|\mathbf{X}, \mathbf{Y})$  can in most cases not be approximated analytically

## Key concept

Approximate  $p(\omega|\mathbf{X}, \mathbf{Y})$  by define a *variational distribution*  $q_\phi(\omega)$ , parametrized by  $\phi$ , and make  $p(\omega|\mathbf{X}, \mathbf{Y})$  and  $q_\phi(\omega)$  as similar as possible by adjusting the variational parameters  $\phi$ .

- For this purpose we can use the *Kullback Leibler divergence* (KL divergence) [8]

$$KL[q_\phi(\omega)||p(\omega|\mathbf{X}, \mathbf{Y})] = \int q_\phi(\omega) \log \frac{q_\phi(\omega)}{p(\omega|\mathbf{X}, \mathbf{Y})} d\omega \quad (15)$$



# Kullback Leibler Divergence

- The KL divergence is a measure of how a probability distribution differs from another probability distribution (A similarity measure).
- In a Bayesian setting, the KL divergence measures the distance from the approximate distribution  $q$  to the true distribution  $p$ .

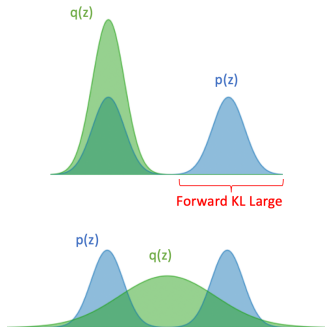
## Some important properties of the KL divergence

- $KL[q||p] \neq KL[p||q]$
- $KL[q||p] \geq 0 \quad \forall q, p$



# Forward KL divergence

$$KL[p||q] = \sum_z p(z) \log \frac{p(z)}{q(z)} \longrightarrow \lim_{q(z) \rightarrow 0} \log \frac{p(z)}{q(z)} \rightarrow \infty$$

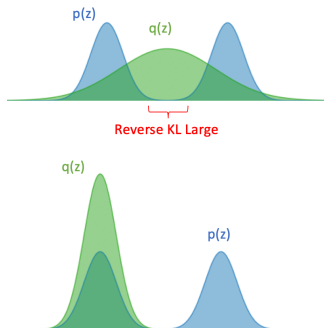


Learning a  $q$  that spreads out to cover all regions where  $p$  has any density (zero avoiding behaviour)



# Reverse KL divergence

$$KL[q||p] = \sum_z q(z) \log \frac{q(z)}{p(z)}$$



KL divergence blow up anywhere  $p(z) = 0$  (zero forcing behaviour)



# Evidence Lower Bound (ELBO)

We want to min the KL divergence with respect to the variational parameters to obtain a variational distribution  $q_{\varphi^*}(\omega)$

$$\varphi^* = \arg \min_{\varphi} KL[q_{\varphi}(\omega) || p(\omega | \mathbf{X}, \mathbf{Y})] \quad (16)$$

We can rewrite the KL divergence in terms of Bayes rule

$$KL[q_{\varphi}(\omega) || p(\omega | \mathbf{X}, \mathbf{Y})] = \int q_{\varphi}(\omega) \log \frac{q_{\varphi}(\omega)}{\frac{p(\omega)p(Y|\mathbf{X},\omega)}{p(\mathbf{Y}|\mathbf{X})}} d\omega \quad (17)$$





# Evidence Lower Bound (ELBO)

$$= \int q_{\phi}(\omega) \log q_{\phi}(\omega) d\omega - \int q_{\phi}(\omega) \log \frac{p(\omega) \mathbf{p}(\mathbf{Y}|\mathbf{X}, \omega)}{\mathbf{p}(\mathbf{Y}|\mathbf{X})} d\omega \quad (18)$$

$$= \int q_{\phi}(\omega) \log \frac{q_{\phi}(\omega)}{p(\omega)} d\omega - \int q_{\phi}(\omega) \log \mathbf{p}(\mathbf{Y}|\mathbf{X}, \omega) d\omega \quad (19)$$
$$+ \int q_{\phi}(\omega) \mathbf{p}(\mathbf{Y}|\mathbf{X}) d\omega$$

$$= KL[q_{\phi}(\omega) || p(\omega)] - \mathbb{E}_{q_{\phi}(\omega)}[\log p(\mathbf{Y}|\mathbf{X}, \omega)] + \log p(\mathbf{Y}|\mathbf{X}) \quad (20)$$

■ Here  $\mathbb{E}_{q_{\phi}(\omega)}[\log p(\mathbf{Y}|\mathbf{X})] = \log p(\mathbf{Y}|\mathbf{X})$



# Evidence Lower Bound (ELBO)

This new expression is still dependent on the intractable marginal likelihood  $p(\mathbf{Y}|\mathbf{X})$

$$KL[q_\phi(\omega)||p(\omega|\mathbf{X}, \mathbf{Y})] = KL[q_\phi(\omega)||p(\omega)] - \mathbb{E}_{q_\phi(\omega)}[\log p(\mathbf{Y}|\mathbf{X}, \omega)] + \log p(\mathbf{Y}|\mathbf{X}) \quad (21)$$

Moving  $\log p(\mathbf{Y}|\mathbf{X})$  to the left-hand side and changing signs defines the ELBO (or VI objective function)

$$\mathcal{L}_{VI}(\phi) \leq \log p(\mathbf{Y}|\mathbf{X}) - KL[q_\phi(\omega)||p(\omega|\mathbf{X}, \mathbf{Y})] \quad (22)$$

where

$$\mathcal{L}_{VI}(\phi) = \mathbb{E}_{q_\phi(\omega)}[\log p(\mathbf{Y}|\mathbf{X}, \omega)] - KL[q_\phi(\omega)||p(\omega)] \quad (23)$$

Since  $KL[q_\phi(\omega)||p(\omega|\mathbf{X}, \mathbf{Y})] \geq 0$  we have that  $\mathcal{L}_{VI}(\phi)$  is a lower bound of  $\log p(\mathbf{Y}|\mathbf{X})$ .



# Variational Inference

- Maximization of the ELBO with respect to the variational parameters defining  $q_{\phi}(\omega)$ , are equivalent to maximization of the log-likelihood  $\log p(\mathbf{Y}|\mathbf{X})$ .
- Maximizing the VI objective we can find the best approximating density  $q_{\phi^*}(\omega)$  by

$$\phi^* = \arg \max_{\phi} \mathcal{L}_{VI}(\phi) \quad (24)$$

- Optimizing  $\mathcal{L}_{VI}(\phi)$  and finding  $q_{\phi^*}(\omega)$  is a compromise between fitting the observed data properly by maximizing the likelihood, and minimizing the "distance" of prior distribution  $p(\omega)$  vs the variational distribution  $q_{\phi}(\omega)$ .



# Approximate posterior predictive distribution

By substituting the true posterior with the variational distribution  $q_{\phi}^*(\omega)$ , we can obtain an approximation of the true posterior distribution

$$\begin{aligned} p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) &= \int p(\mathbf{y}^*|\mathbf{x}^*, \omega) p(\omega|\mathbf{X}, \mathbf{Y}) d\omega \\ &\approx \int p(\mathbf{y}^*|\mathbf{x}^*, \omega) q_{\phi}^*(\omega) d\omega \\ &\approx q_{\phi}^*(\mathbf{y}^*|\mathbf{x}^*). \end{aligned} \tag{25}$$

In even small BNNs, the posterior is either difficult or intractable to calculate, and approximations are needed.



# How to solve this in practice?

- To solve  $\mathcal{L}_{VI}(\varphi)$  requires computations over the entire data set.
- We can write the  $\mathcal{L}_{VI}(\varphi)$  in terms of the model output  $f^{\omega}(\mathbf{x}^{(i)})$
- $\int q_{\varphi}(\omega) \log p(\mathbf{y}^{(i)} | f^{\omega}(\mathbf{x}^{(i)})) d\omega$  is not tractable for BNNs with more than one hidden layer.
- The integral can be approximated with different Monte Carlo estimators (See Gal section 3.1.1 [7]).
  - 1 The score function estimator
  - 2 **Re-parametrisation trick** (Path-wise derivative estimator, stochastic backpropagation)
  - 3 Characteristic function estimator
  - 4 Local reparametrization trick (bayesbybackprop)

We will use the reparametrization trick estimator, introduced by Kingma and Welling [9] to estimate the integral further



# The reparametrization trick

- An iteration during training of an BNN with VI, consists of a forward and backward pass for updating of the model parameters.
- A sample is drawn from the variational posterior distribution during the forward pass to evaluate  $\mathcal{L}_{VI}(\phi)$ , i.e. a stochastic sampling step.
- In the backward pass we need to calculate the gradients of  $\phi$ .
- Since  $\phi$  is stochastic sampled in the forward pass it is not possible to directly calculate the gradient with the chain rule and backpropagation.
- The reparametrization trick samples from a parameter-free distribution and maps it to a deterministic function, where a gradient can be defined.



# The reparametrization trick

We can approximate  $\mathcal{L}_{VI}(\varphi)$  with mini-batch optimization

$$\hat{\mathcal{L}}_{VI}(\varphi) = -\frac{N}{M} \sum_{i \in S} \int q_{\varphi}(\boldsymbol{\omega}) \log p(\mathbf{y}^{(i)} | \mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x}^{(i)})) d\boldsymbol{\omega} + KL[q_{\varphi}(\boldsymbol{\omega}) || p(\boldsymbol{\omega})],$$

with a random index set  $S$  of size  $M$ . The approximation above is an unbiased stochastic estimator, i.e.  $\mathbb{E}[\hat{\mathcal{L}}_{VI}(\varphi)] = \mathcal{L}_{VI}(\varphi)$ .

- The reparametrization trick introduces a new independent random variable  $\epsilon$  used to reparametrize the weights.
- In each weight matrix  $\mathbf{W}_{l,i}$  we factorize the distribution of the rows  $\mathbf{w}_{l,i}$  ( $l = 1 \dots L$  (Layers) and  $i = 1 \dots I$  (Nodes)).
- Reparametrization takes place by collecting  $q_{\varphi_{l,i}}^*(\mathbf{w}_{l,i})$  so that  $\mathbf{w}_{l,i} = g(\varphi_{l,i}, \epsilon_{l,i})$ .
- we write  $p(\epsilon) = \prod_{l,i} p(\epsilon_{l,i})$  and  $\boldsymbol{\omega} = g(\varphi, \epsilon)$ .



# The reparametrization trick

$$\hat{\mathcal{L}}_{VI}(\phi) = -\frac{N}{M} \sum_{i \in S} \int p(\epsilon) \log p(\mathbf{y}^{(i)} | \mathbf{f}^{g(\phi, \epsilon)}(\mathbf{x}^{(i)})) d\epsilon - KL[q_{\phi}(\omega) || p(\omega)]$$

Applying the reparametrization trick gives us the following Monte Carlo estimator:

$$\hat{\mathcal{L}}_{MC}(\phi) = -\frac{N}{M} \sum_{i \in S} \log p(\mathbf{y}^{(i)} | \mathbf{f}^{g(\phi, \epsilon)}(\mathbf{x}^{(i)})) + KL[q_{\phi}(\omega) || p(\omega)], \quad (26)$$

where  $\mathbb{E}_{S, \epsilon}[\hat{\mathcal{L}}_{MC}(\phi)] = \mathcal{L}_{VI}(\phi)$ , i.e. an unbiased estimator.





# The reparametrization trick

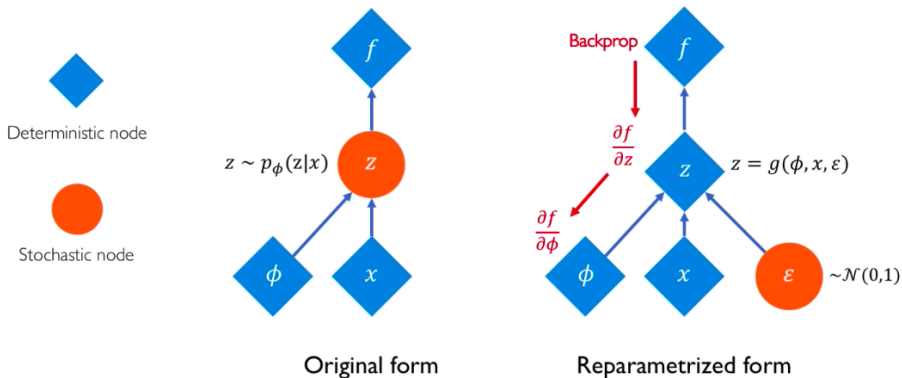


Figure is from [5]



# Variational Auto-Encoder (VAE)

- A data set  $\mathbf{X}$  is generated by a random process that involves an unobserved random variable  $\mathbf{z}$
- A VAE consists of a two step process
  - 1)  $\mathbf{z}$  is sampled from a prior  $p_{\theta}(\mathbf{z})$
  - 2)  $\mathbf{X}$  is generated from a conditional distribution  $p_{\theta}(\mathbf{X}|\mathbf{z})$
- $p_{\theta}(\mathbf{z})$  and  $p_{\theta}(\mathbf{X}|\mathbf{z})$  comes from a parametric family of distributions and their density functions are differentiable almost everywhere w.r.t.  $\mathbf{z}$  and  $\theta$ .
- A probabilistic auto-encoder is a neural network that is trained to represent its input  $\mathbf{X}$  as  $p_{\theta}(\mathbf{X})$  via a latent representation  $\mathbf{z} \sim p_{\theta}(\mathbf{z})$

$$p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p_{\theta}(\mathbf{x}|\mathbf{z}) p_{\theta}(\mathbf{z}) d\mathbf{z}. \quad (27)$$



# Variational Auto-Encoder (VAE)

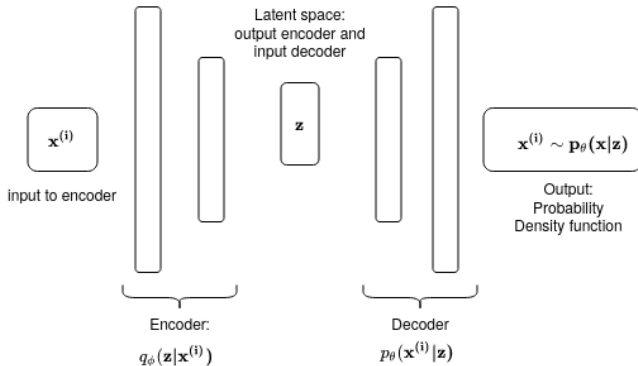
- $p(\mathbf{z})$  is unknown and observations  $\mathbf{z}^{(i)}$  are not accessible
- Have to use  $\mathbf{X}$  to generate  $\mathbf{z}^{(i)} \sim p(\mathbf{z}|\mathbf{X})$
- $p_{\theta}(\mathbf{z}|\mathbf{X})$  is intractable but can be approximated with a recognition model  $q_{\phi}(\mathbf{z}|\mathbf{X})$  (Variational inference)
- An auto-encoder that uses a recognition model is called Variational Auto-Encoder (VAE)

$$L_{VAE}(\theta, \phi, \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} \left[ \log p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z}) \right] - KL[q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\theta}(\mathbf{z})]$$

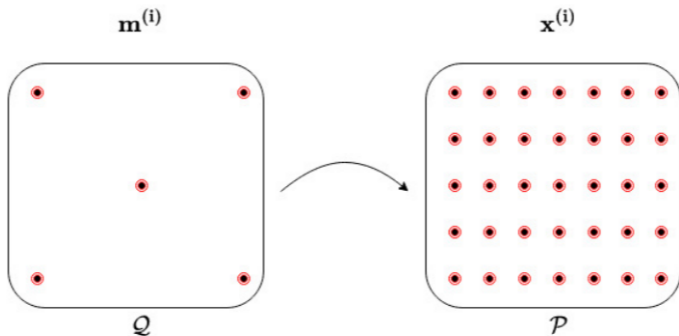
$$L_{CVAE}(\theta, \phi, \mathbf{x}^{(i)}, \mathbf{c}) = \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c})} \left[ \log p_{\theta}(\mathbf{x}^{(i)}|\mathbf{c}, \mathbf{z}) \right] + KL \left[ q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{c}) || p_{\theta}(\mathbf{z}|\mathbf{c}) \right]$$



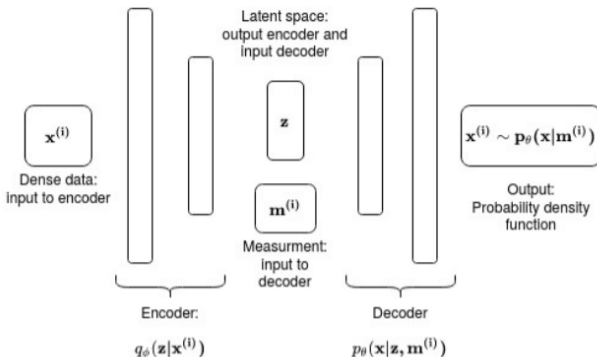
# Variational Auto-Encoder (VAE)



# Application - reconstruction from sparse measurements



# Semi Conditional VAE

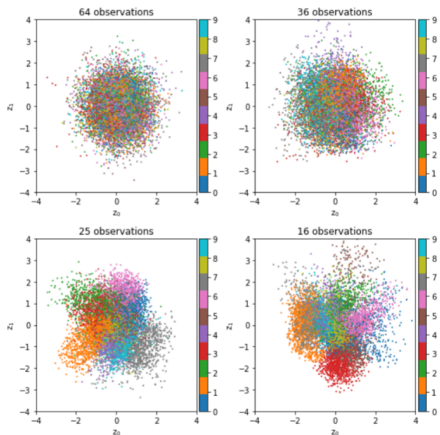
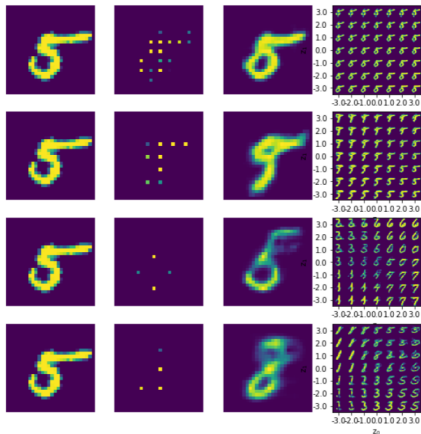


$$p_\theta(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{m}^{(i)}) = p_\theta(\mathbf{z}|\mathbf{x}^{(i)}) \quad \text{and} \quad q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{m}^{(i)}) = q_\phi(\mathbf{z}|\mathbf{x}^{(i)})$$

$$\mathcal{L}_{SCVAE}(\theta, \phi, \mathbf{x}^{(i)}, \mathbf{m}^{(i)}) = KL \left[ q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) || p_\theta(\mathbf{z}|\mathbf{m}^{(i)}) \right] + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \left[ \log p_\theta(\mathbf{x}^{(i)}|\mathbf{m}^{(i)}, \mathbf{z}) \right]$$



# MNIST Data set



# Fluid flow reconstruction

- Bergen Ocean Model Data
- Velocity driven by weather, tides, river influx, high low pressure
- Data consists of two features ( $u$  and  $v$  velocity components)
- Data set  $(8500 \times 32 \times 32 \times 2) \rightarrow \text{timestep} = 10 \text{ min}$

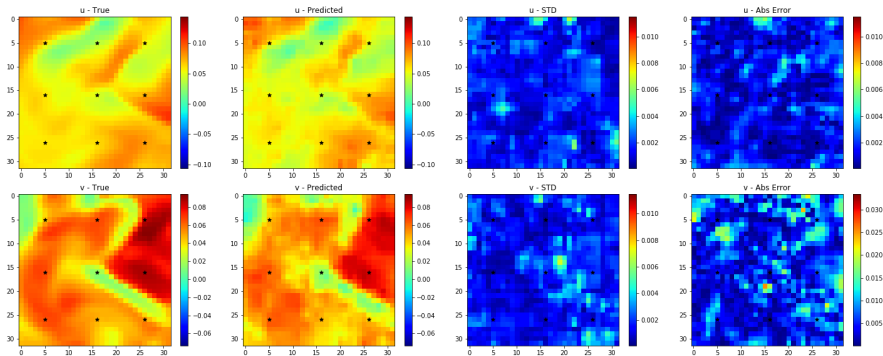


Figure:  $25.6 \text{ km} \times 25.6 \text{ km}$  area (animation)





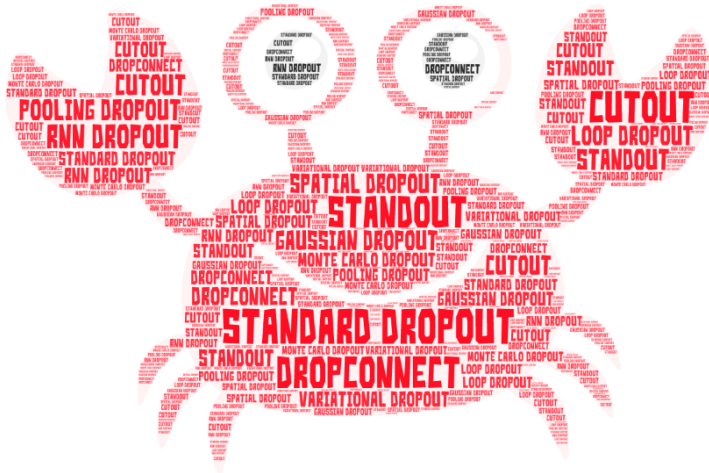
**End of part one...**  
**Next, Uncertainty**  
**Quantification with Dropout**

# Regularization in deep learning

- Regularization addresses issues related to overfitting
  - Low bias and/or high variance
  - Little data
  - Regularization reduces the variance of the model
- 1 Modify the Loss function (L1/L2 penalization, entropy)
- 2 Modify data sampling (Data augmentation, cross validation)
- 3 Changing training approach (Stochastic regularization)
  - Injecting noise
  - Dropout

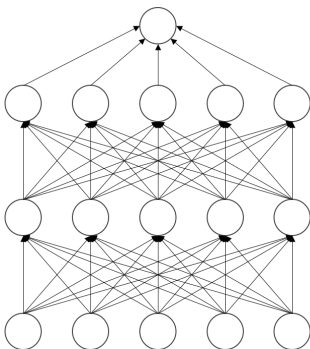


# Dropout

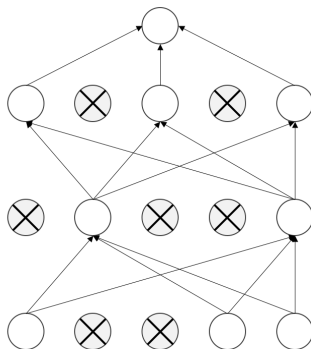


# Dropout

- Dropout introduces noise by randomly forcing a proportion of the nodes in the model to have zero output [6].
- The nodes that are set to zero are determined by a Bernoulli distribution



Standard Neural Net



After applying dropout



# Dropout

- During prediction, dropout is turned off, resulting in a point estimate of the class probabilities.
- MC Dropout is similar to regular Dropout
- During prediction, dropout is still turned on, randomly shutting a proportion of the nodes off.
- MC Dropout generates a distribution over the model parameters by repeating the nodes sampling several times and predicting for each configuration.
- The process is similar to a bootstrap procedure (scaled with the dropout rate).
- A dropout rate of 0.5 gives the largest variance.
- Relationship between Dropout and Regularization
- Can be showed that Dropout approximates a BNN (Yarin Gal) [7]



# Dropout

We consider dropout as a function of the parameters.

- We introduce the vectors  $\mathbf{z}_1$  and  $\mathbf{z}_2$ , which have the same number of elements as the input and hidden layer
- Whether an element of the vector is 0 or 1 follows from a Bernoulli distribution such that  $\mathbf{z}_i$  is 1 is  $0 \leq 1 - p_i \leq 1$  for  $i = 1, 2$  and we write  $\hat{\mathbf{x}} = \mathbf{z}_1 \odot \mathbf{x}$ .
- The output of the first layer can be written as:  $\mathbf{h} = \sigma(\mathbf{W}_1 \hat{\mathbf{x}} + \mathbf{b})$ .
- The same procedure can be done with the hidden layer  $\mathbf{h}$  but with a percentage  $p_2$  instead so that  $\hat{\mathbf{h}} = \mathbf{z}_2 \odot \mathbf{h}$ .
- linearly transform the output such that the output of the model becomes  $\hat{\mathbf{y}} = \hat{\mathbf{h}} = \mathbf{W}_2 \hat{\mathbf{h}}$ .



# Dropout

- During training of the network, we simply sample from the Bernoulli distribution for each vector  $\mathbf{z}_1$  and  $\mathbf{z}_2$  in each forward propagation and use the same samples in the backpropagation.

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{W}_2 \hat{\mathbf{h}} \\ &= \mathbf{W}_2 (\mathbf{z}_2 \odot \mathbf{h}) \\ &= \mathbf{W}_2 (\text{diag}(\mathbf{z}_2) \mathbf{h}) \\ &= \hat{\mathbf{W}}_2 (\sigma(\mathbf{W}_1 (\mathbf{z}_1 \odot \mathbf{x}) + \mathbf{b})) \\ &= \hat{\mathbf{W}}_2 (\sigma(\mathbf{W}_1 \text{diag}(\mathbf{z}_1) \mathbf{x} + \mathbf{b})) \\ &= \hat{\mathbf{W}}_2 (\sigma(\hat{\mathbf{W}}_1 \mathbf{x} + \mathbf{b})) = \mathbf{f}^{\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2, \mathbf{b}}\end{aligned}\tag{28}$$

where  $\hat{\mathbf{W}}_1 = \mathbf{W}_1 \text{diag}(\mathbf{z}_1)$  and  $\hat{\mathbf{W}}_2 = \mathbf{W}_2 \text{diag}(\mathbf{z}_2)$  and  $\omega = \{\hat{\mathbf{W}}_1, \hat{\mathbf{W}}_2, \mathbf{b}\}$ .



# Dropout

$$\hat{\mathcal{L}}_{Dropout}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) = \frac{1}{M} \sum_{i \in S} \mathcal{L}^{\hat{\mathbf{W}}_1^i, \hat{\mathbf{W}}_2^i, \mathbf{b}}(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) + \lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2 \quad (29)$$

We can express the cost function in terms of the negative log likelihood (here classification)

$$\hat{\mathcal{L}}_{Dropout}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) = -\frac{1}{M} \sum_{i \in S} \log p(\mathbf{y}^{(i)} | \mathbf{f}^{\hat{\mathbf{W}}_1^i, \hat{\mathbf{W}}_2^i, \mathbf{b}}(\mathbf{x}^{(i)})) + \lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2 \quad (30)$$

with  $\hat{\mathbf{W}}_1^i, \hat{\mathbf{W}}_2^i$  corresponding to new masks  $\mathbf{z}_1^i, \mathbf{z}_2^i$  sampled for each data point  $i$  (Here we used data sub-sampling with a random index set  $S$  of size  $M$ ).





# Dropout

The dropout operation can be written in terms of the function

$$\hat{\omega}_i = \left\{ \hat{\mathbf{W}}_1^i, \hat{\mathbf{W}}_2^i, \mathbf{b} \right\} = \left\{ \mathbf{W}_1 \text{diag}(\hat{\epsilon}_1^i), \mathbf{W}_2 \text{diag}(\hat{\epsilon}_2^i), \mathbf{b} \right\} = g(\psi, \hat{\epsilon}_i)$$

- $\hat{\epsilon}_1^i \sim p(\epsilon_1)$  and  $\hat{\epsilon}_2^i \sim p(\epsilon_2)$  where  $1 \leq i \leq N$ , and  $p(\hat{\epsilon}_l)$  is a vector of zeros and ones,  $l = \{1, 2\}$ ,  $\psi = \{\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}\}$
- $\hat{\epsilon}_1^i$  and  $\hat{\epsilon}_2^i$  Bernoulli distributed with a probability  $p_l$ , with same size as the columns of the  $\mathbf{W}_l$
- For each column of weights of the different neural network layer weights  $\mathbf{W}_l$ , there is a probability  $p_l$  that a particular column will be multiplied with zero, and thus be "dropped out".



# Dropout

We can write the dropout neural network cost function in terms of  $g(\psi, \hat{\epsilon}_i)$ :

$$\begin{aligned}\hat{\mathcal{L}}_{Dropout}(\mathbf{W}_1, \mathbf{W}_2, \mathbf{b}) = & -\frac{1}{M} \sum_{i \in S} \log p(\mathbf{y}^{(i)} | \mathbf{f}^{g(\psi, \hat{\epsilon}_i)}(\mathbf{x}^{(i)})) \\ & + \lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2 \quad (31)\end{aligned}$$



# Comparing the objective functions

Gradient decent  $\rightarrow$  derivatives w.r.t. the parameters

$$\begin{aligned}\frac{\partial}{\partial \psi} \hat{\mathcal{L}}_{Dropout}(\psi) = & -\frac{1}{M} \sum_{i \in S} \frac{\partial}{\partial \psi} \log p(\mathbf{y}^{(i)} | \mathbf{f}^{g(\psi, \hat{\epsilon}_i)}(\mathbf{x}^{(i)})) + \dots \\ & \frac{\partial}{\partial \psi} (\lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2),\end{aligned}\quad (32)$$

and the derivative of the variational objective can be expressed as:

$$\frac{\partial}{\partial \varphi} \hat{\mathcal{L}}_{VI}(\varphi) = -\frac{N}{M} \sum_{i \in S} \frac{\partial}{\partial \varphi} \log p(\mathbf{y}^{(i)} | \mathbf{f}^{g(\varphi, \epsilon)}(\mathbf{x}^{(i)})) d\epsilon + \frac{\partial}{\partial \varphi} KL[q_{\varphi}(\omega) || p(\omega)]$$

- The two objective functions are very similar to one another. The difference is the regularization term, and a different scaling of the log-likelihood term.



# KL-condition

We can define the prior  $p(\omega)$  so that the following condition holds:

$$\frac{\partial}{\partial \psi} N(\lambda_1 \|\mathbf{W}_1\|^2 + \lambda_2 \|\mathbf{W}_2\|^2 + \lambda_3 \|\mathbf{b}\|^2) = \frac{\partial}{\partial \phi} KL(q_\phi(\omega) \| p(\omega)) \quad (33)$$

Assuming that this condition holds, we have the following relation between the objective function of dropout NN and VI:

$$\frac{\partial}{\partial \psi} \hat{\mathcal{L}}_{Dropout}(\psi) = \frac{1}{N} \frac{\partial}{\partial \phi} \hat{\mathcal{L}}_{VI}(\phi), \quad (34)$$

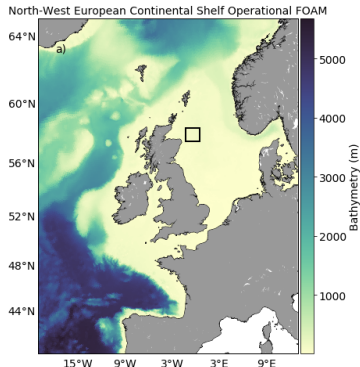
The KL-condition hold given

- large enough number of hidden units
- the model priors  $p(\omega)$  is a product of uncorrelated Gaussian distributions over each weight



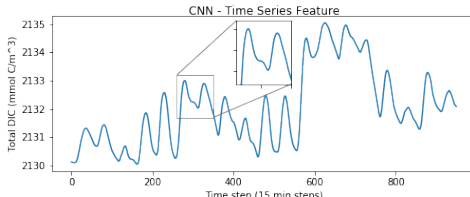
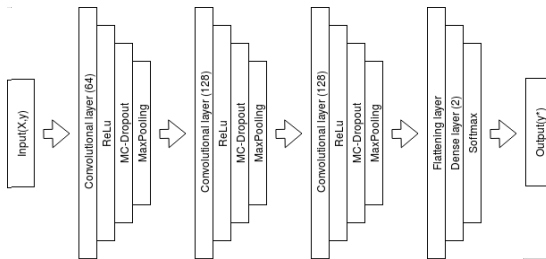
# MC Dropout Application

- **Background:** A device monitor the CO<sub>2</sub> concentration in the ocean near a storage formation
- **Challenges:**
  - Few measurements
  - Natural variation or a leakage?
  - Should we stay or should we go?
- Binary time series classification (leak or no-leak)
- Bayesian Convolutional Neural Networks can classify time series probabilistically



# Bayesian Convolutional Neural Networks

$$\hat{\mathbf{y}} = \sigma_S(\sigma_R(\sigma_R(\sigma_R(\sigma_R(\mathbf{x}\hat{\mathbf{W}}_1 + \mathbf{b})\hat{\mathbf{W}}_2)\hat{\mathbf{W}}_3)\hat{\mathbf{W}}_4)\mathbf{W}_5$$



# BCNN objective function

## ■ MC Dropout objective function

$$\hat{\mathcal{L}}(\boldsymbol{\omega}) = -\frac{1}{M} \sum_{i \in S} \log p(\mathbf{Y}_i | \mathbf{f}^{\hat{\boldsymbol{\omega}}_i}(\mathbf{X}_i)) \quad (35)$$

$$+ \lambda_1 \|\mathbf{W}_1\|_2^2 + \lambda_2 \|\mathbf{W}_2\|_2^2 + \lambda_3 \|\mathbf{b}_1\|_2^2 + \lambda_4 \|\mathbf{b}_2\|_2^2.$$

## ■ Dropped out weights

$$\hat{\boldsymbol{\omega}}_i = \{\hat{\mathbf{W}}_1^i, \hat{\mathbf{W}}_2^i, \mathbf{b}_1, \mathbf{b}_2\} = \{\text{diag}(\hat{\boldsymbol{\epsilon}}_1^i) \mathbf{W}_1, \text{diag}(\hat{\boldsymbol{\epsilon}}_2^i) \mathbf{W}_2, \mathbf{b}_1, \mathbf{b}_2\},$$

with  $\hat{\boldsymbol{\epsilon}}_1^i \sim p(\hat{\boldsymbol{\epsilon}}_1^i)$  and  $\hat{\boldsymbol{\epsilon}}_2^i \sim p(\hat{\boldsymbol{\epsilon}}_2^i)$ , for  $1 \leq i \leq N$  where  $p(\boldsymbol{\epsilon}_1)$  and  $p(\boldsymbol{\epsilon}_2)$  is Bernoulli distributions with probabilities  $p_1$  and  $p_2$  respectively.



# Optimization and prediction

Trough back-propagation and stochastic gradient decent we find optimal parameters for the model  $\mathbf{f}^\omega$  as

$$\arg \min_{\omega} \hat{\mathcal{L}}(\omega).$$

- With weights optimized we can use the model to predict new classes given some new input  $\hat{\mathbf{x}}^*$ .
- The weights  $\hat{\omega}_t \sim p(\omega|\mathbf{X}, \mathbf{Y})$  are generated from the posterior distribution such that,

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*, \omega) p(\omega|\mathbf{X}, \mathbf{Y}) d\omega \approx \frac{1}{T} \sum_{t=1}^T \mathbf{f}^{\hat{\omega}_t}(\mathbf{x}^*) \xrightarrow{T \rightarrow \infty} \mathbb{E}[\mathbf{y}^*],$$

which is the unbiased estimator and what we referred to as MC-dropout.





# Uncertainty quantification

The empirical mean of the predicted leak is defined as

$$\hat{\mu}_{Leak} = \frac{1}{T} \sum_{t=1}^T p(\mathbf{y}^* = \mathbf{c}_1 | \mathbf{x}^*, \hat{\omega}_t),$$

and empirical standard deviation as

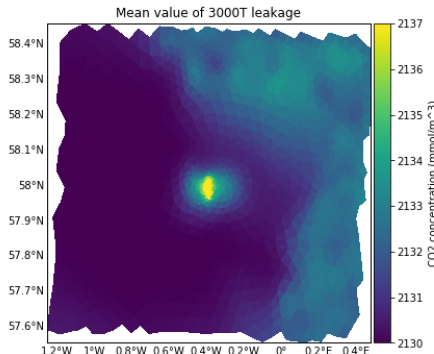
$$\hat{\sigma}_{Leak} = \sqrt{\frac{1}{T} \sum_{t=1}^T [p(\mathbf{y}^* = \mathbf{c}_1 | \mathbf{x}^*, \hat{\omega}_t) - \hat{\mu}_{Leak}]^2},$$

where  $T$  is number of forward passes and  $\mathbf{c}_1$  leakage class.



# BCNN in time series classification

- 3 3D simulations of leaks at Goldeneye (100x100 km)
- No leak, 30T, 300T and 3000T
- Test data independent in time (bottom layer)
- Train, test and validation data sets
- Difference transform

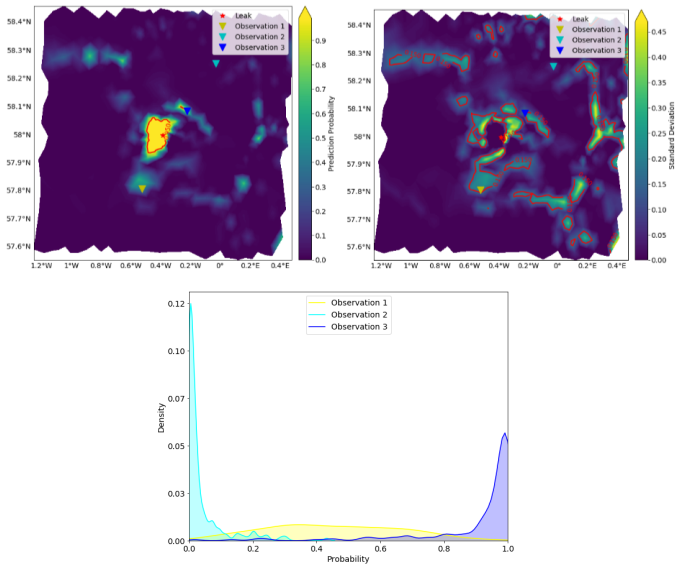


	Leak	No-Leak	# Time Series	Start/End
Training Data	75.8 %	24.2 %	116659	Start
Validation Data	75.8 %	24.2 %	49997	Start
Test Data	69.8 %	36.2 %	6944	End

Table 1. Overview of the train, validation and test data



# Results







# Summary

- Bayesian Neural Networks
- Variational Inference
- ELBO
- The Reparametrization trick
- Variational Auto-Encoders
- Dropout approximating Bayesian Neural Networks








**Thank you!**

# References I

-  Kwon, Yongchan, et al. "Uncertainty quantification using bayesian neural networks in classification: Application to ischemic stroke lesion segmentation." (2018).
-  Kendall, Alex, and Yarin Gal. "What uncertainties do we need in bayesian deep learning for computer vision?." Advances in neural information processing systems. 2017.
-  Mahesh Subedar and Ranganath Krishnan and Paulo Lopez Meyer and Omesh Tickoo and Jonathan Huang, "Uncertainty aware audiovisual activity recognition using deep Bayesian variational inference", 2019, arXiv 1811.10811.
-  <https://www.analyticsvidhya.com/blog/2016/06/bayesian-statistics-beginners-simple-english/>



# References II

-  <https://towardsdatascience.com/reparameterization-trick-126062cfd3c3>
-  Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1):1929–1958, 2014.
-  Gal, Yarin. "Uncertainty in deep learning." University of Cambridge 1.3 (2016): 4.
-  Solomon Kullback and Richard A Leibler. On information and sufficiency. The annals of mathematical statistics, 22(1):79–86, 1951.3.3
-  Diederik P Kingma and Max Welling. Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114, 2013.2.5, 3.3.1, 3.3.1, 3.3.1, 4.4, 4.4

