Uncertainty Quantification

ML Training Course

Mariana Clare

ECMWF

Outline

- Why quantifying uncertainty is important?
- Different types of uncertainty
- Different methods to quantify uncertainty

Why quantifying uncertainty is important?

ECMWF

What happens if we only have a deterministic prediction?

Neural Network output:

COFFEE

TEA

But this has limited value…

For machine learning methods to be trustworthy, they must tell us when they're ***not sure***

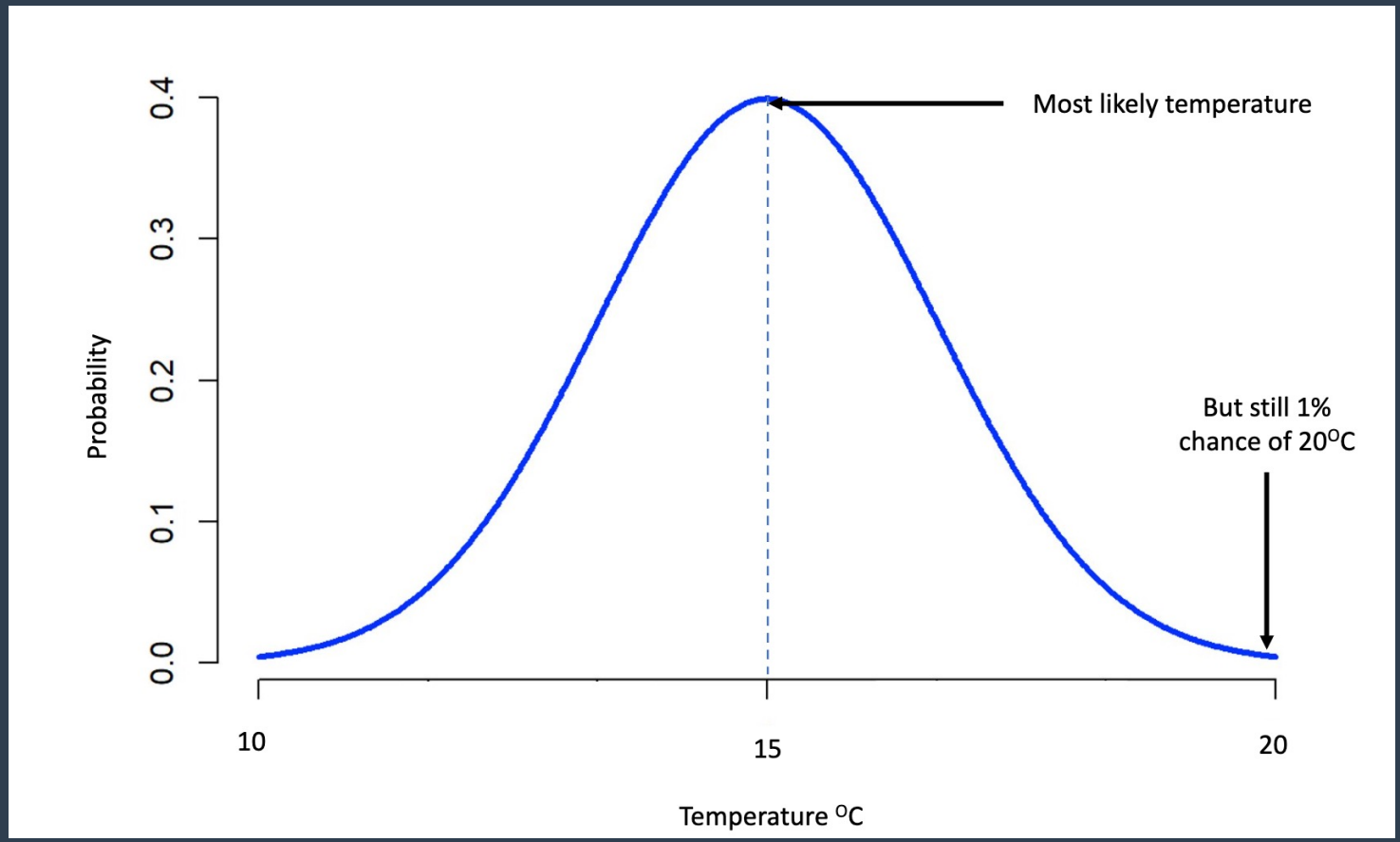
This is especially true for weather and climate forecasts, which are inherently uncertain because of:

- Chaotic nature of Earth System
- Uncertainty in Input Conditions
- Uncertainty in Model itself

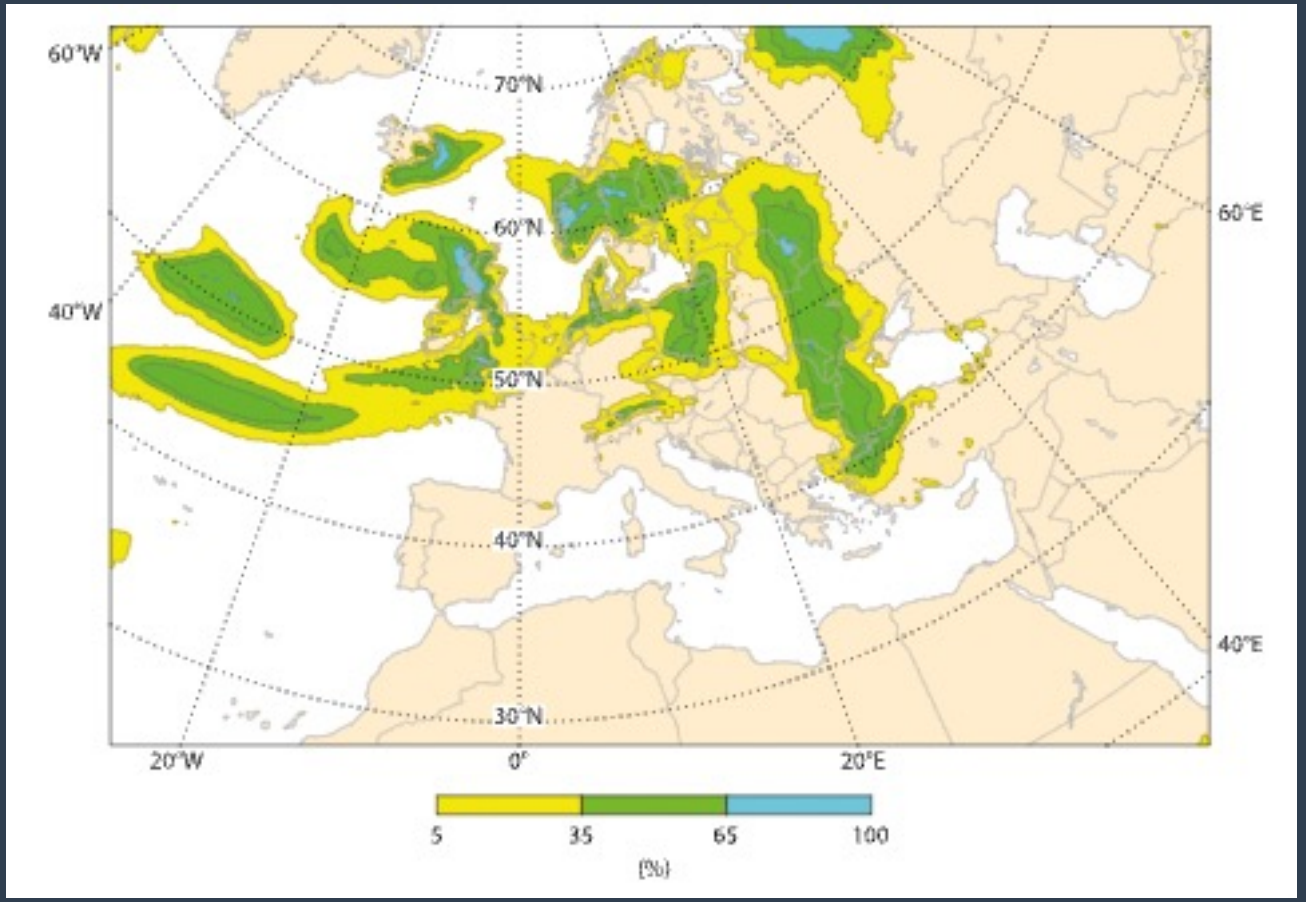
and is particularly important for extreme events when we need to know if there is even a slight chance of one occurring

Probabilistic forecast

Instead of predicting that the temperature in Reading in 4 days time will be 15°C (likely wrong), we need:



In weather models, we quantify this uncertainty using an ensemble forecast with perturbed initial conditions and perturbations in model itself



Ensemble-based probability of precipitation in excess of 5mm

Different types of uncertainty

ECMWF

Aleatoric vs. Epistemic Uncertainty

Epistemic Uncertainty:

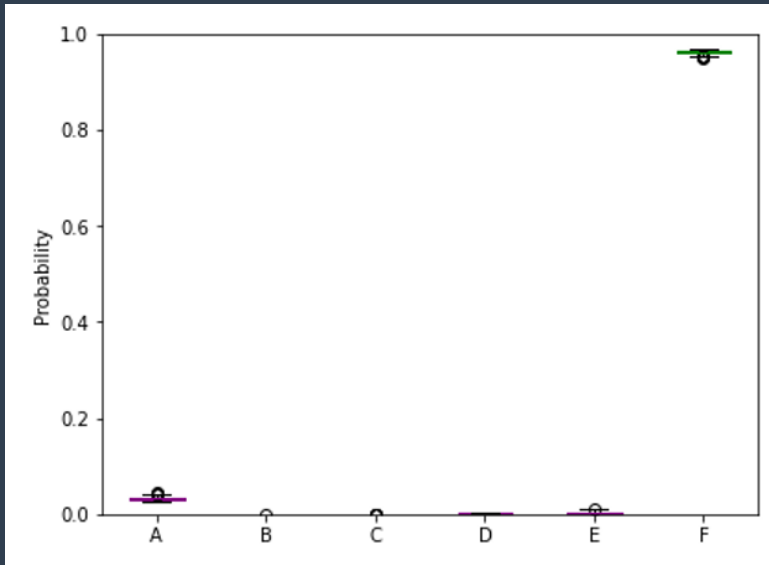
- Uncertainty in the model
- Possible to reduce with more data
- Quantify using an ensemble of models

Aleatoric Uncertainty:

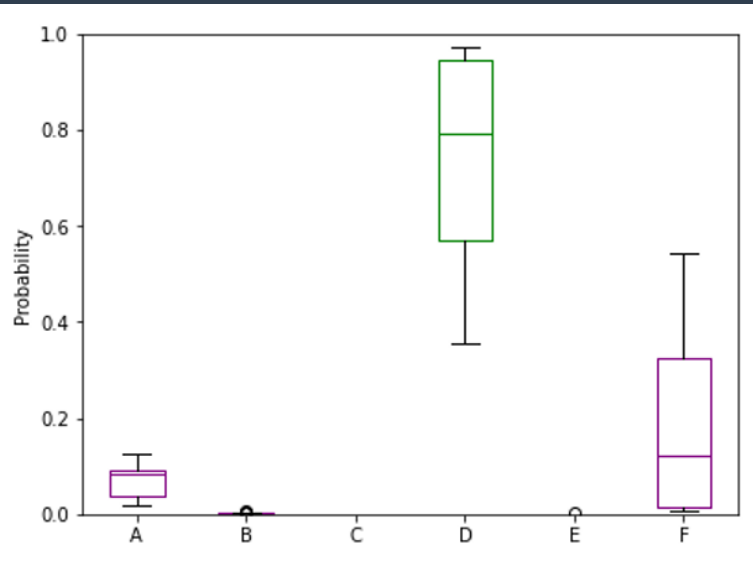
- Uncertainty in input data
- Irreducible uncertainty but still needs to be quantified
- Quantify by perturbing input data and/or predicting a distribution rather than single output

A box-and-whisker plot can help us distinguish between epistemic and aleatoric uncertainty in the results

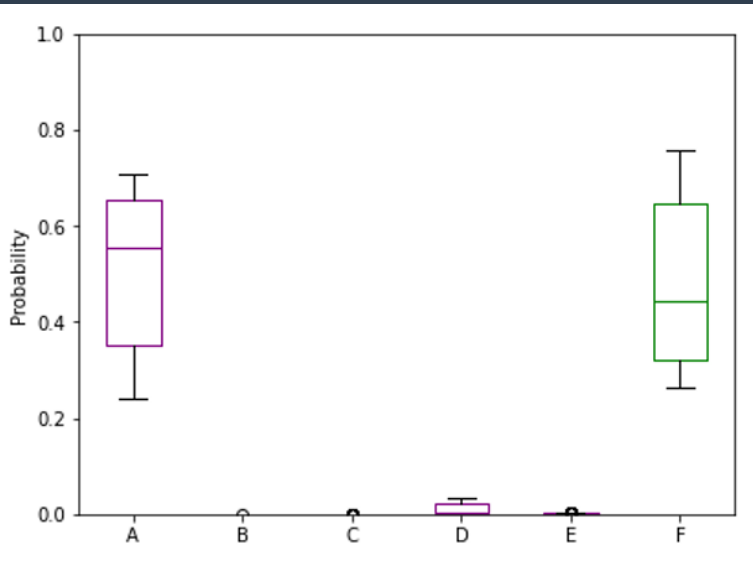
Box and whisker plots



Correct category predicted with high certainty



Correct category predicted with some epistemic certainty

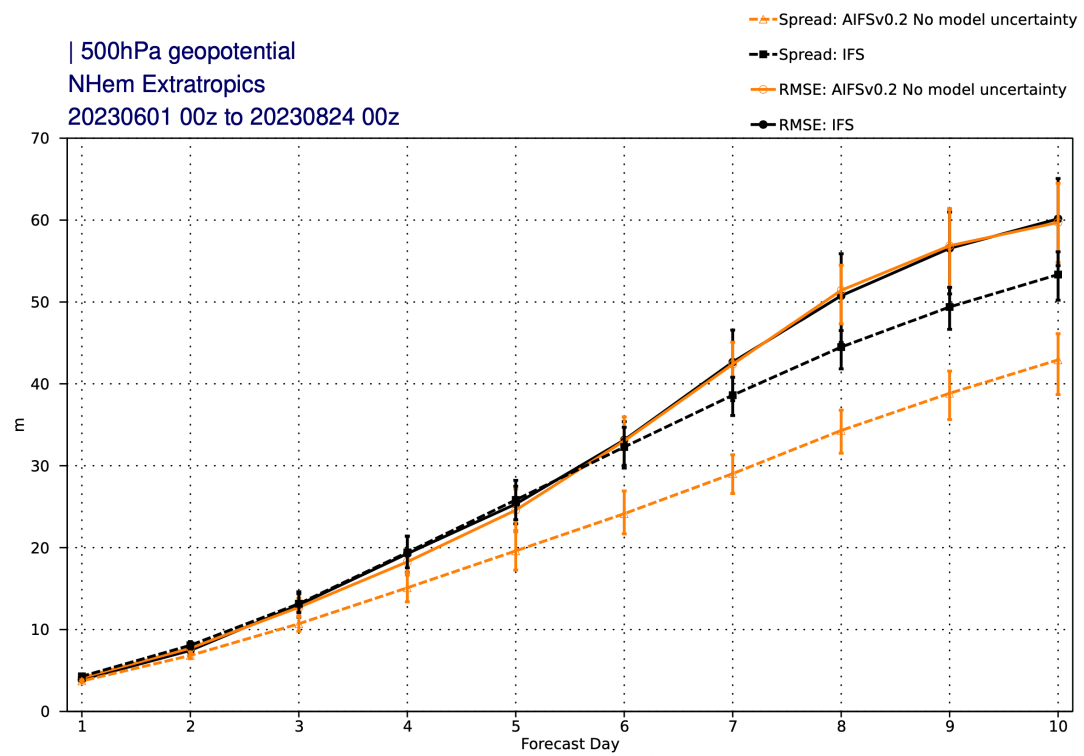


Incorrect category predicted with high aleatoric and epistemic uncertainty

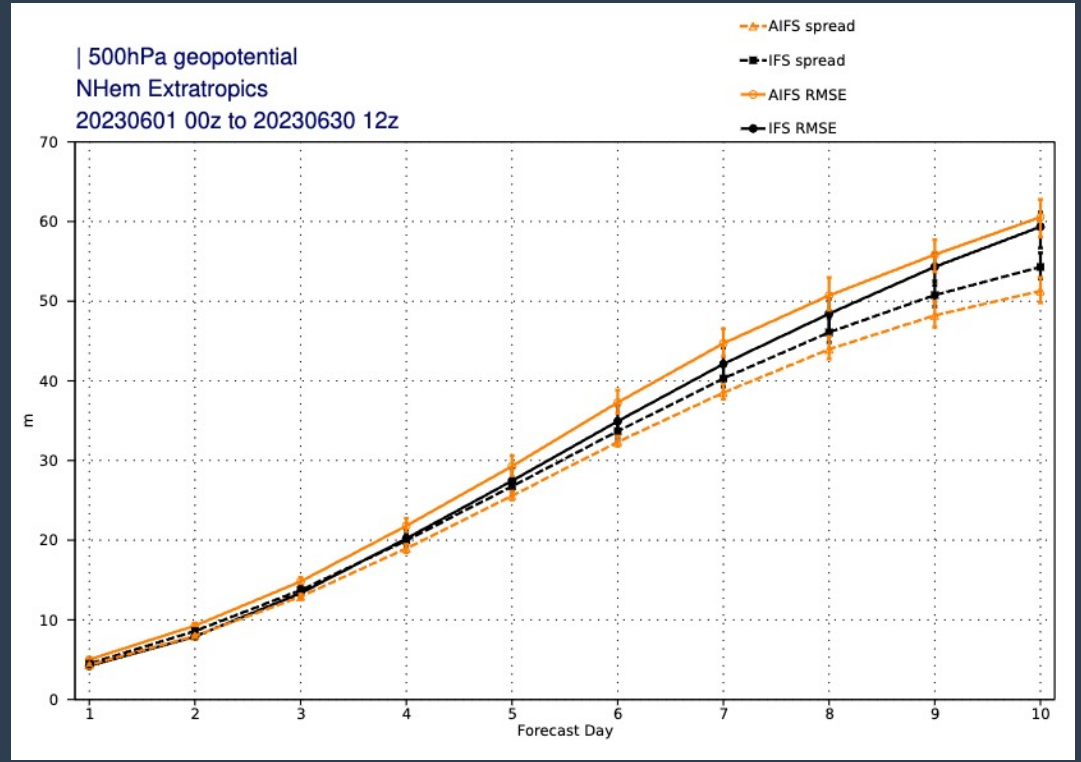
Predictions with low aleatoric and low epistemic uncertainty are more trustworthy.
Predictions with high aleatoric and high epistemic uncertainty suggest the ML method is unsure

Variance-Error relationships

To quantify uncertainty properly, we need to quantify both aleatoric and epistemic



Only aleatoric uncertainty



Aleatoric and epistemic uncertainty

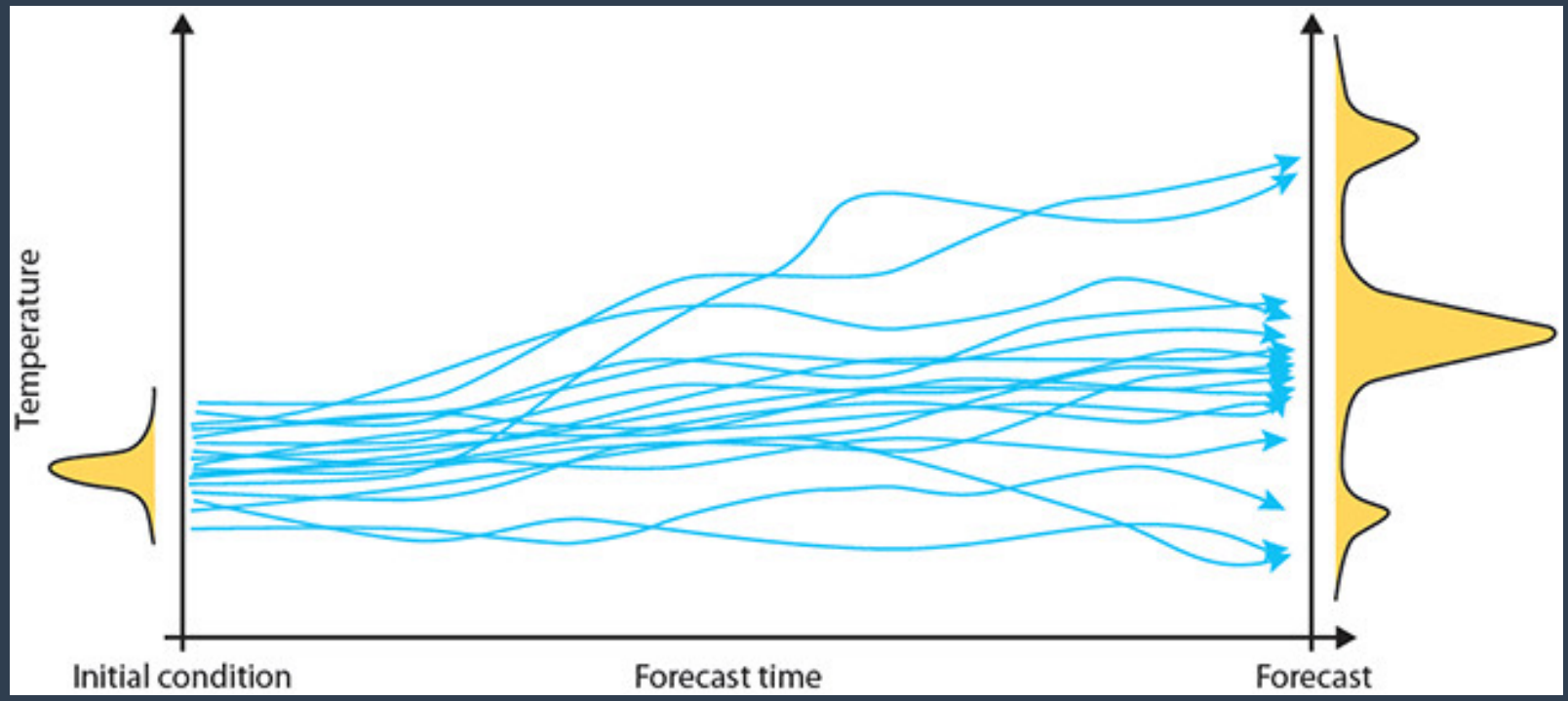
Different methods to quantify uncertainty

ECMWF

What if all we have is a model checkpoint?

Test-time augmentation

Run model predictions multiple times perturbing initial conditions and propagating uncertainty to the outputs



Results can be improved by making sensible choices for initial condition perturbations

This is similar to how we estimate aleatoric uncertainty in the IFS ensemble

Test-time augmentation

Advantages:

- Very easy to implement
- In an ML model, this is not too expensive because prediction is cheap

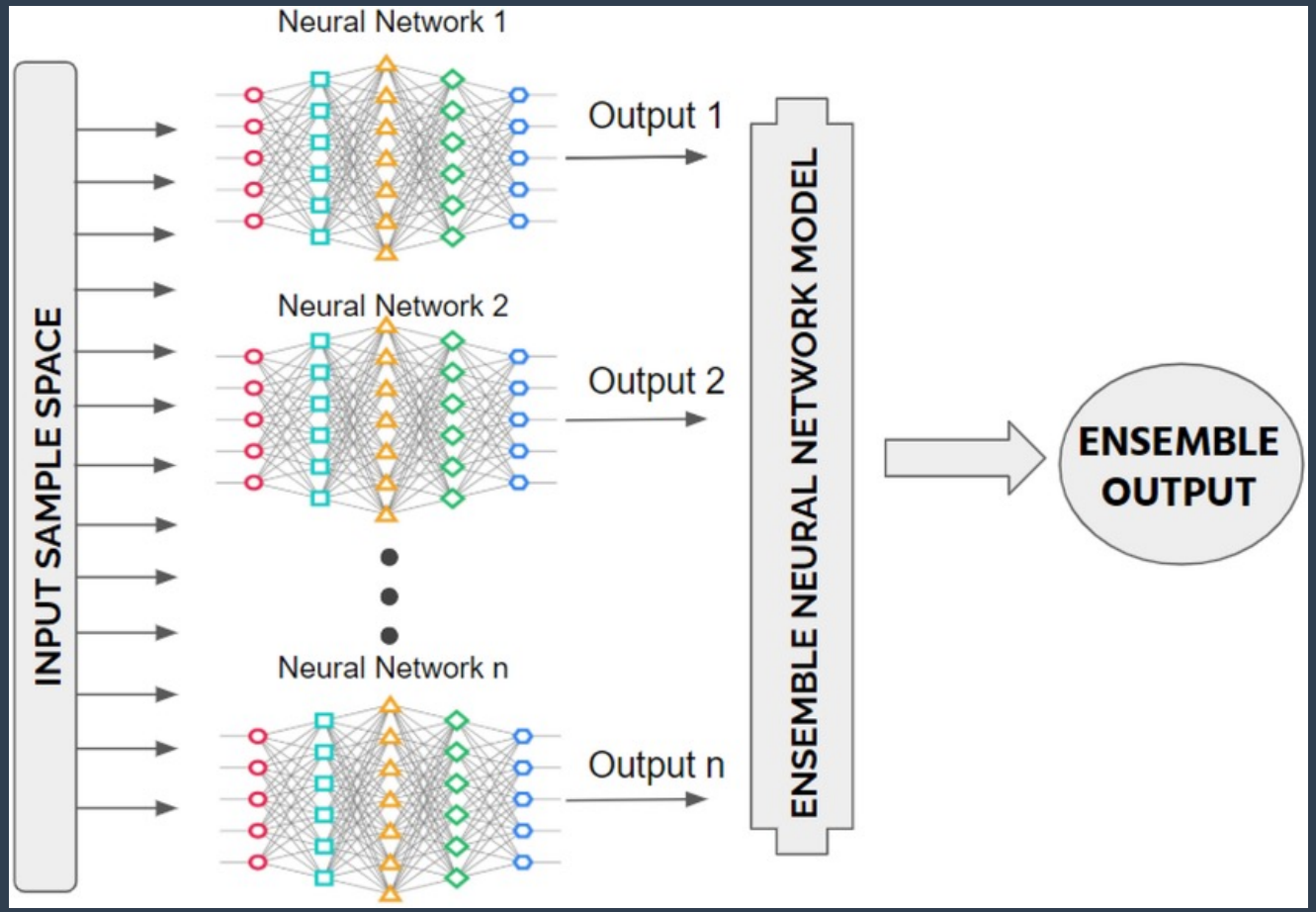
Disadvantages:

- Can be difficult to generate sensible/realistic perturbations
- Only quantifies aleatoric uncertainty

Can we just train 50 models?

Deep ensembles

Train multiple (deep) Neural Networks and use them to create an ensemble of models. These neural networks usually have different architectures and can even be different types of model



Deep ensembles

Advantages:

- Easy to implement without changing any of the training procedure
- Capture epistemic uncertainty and can be easily combined with test-time augmentation to capture aleatoric uncertainty too

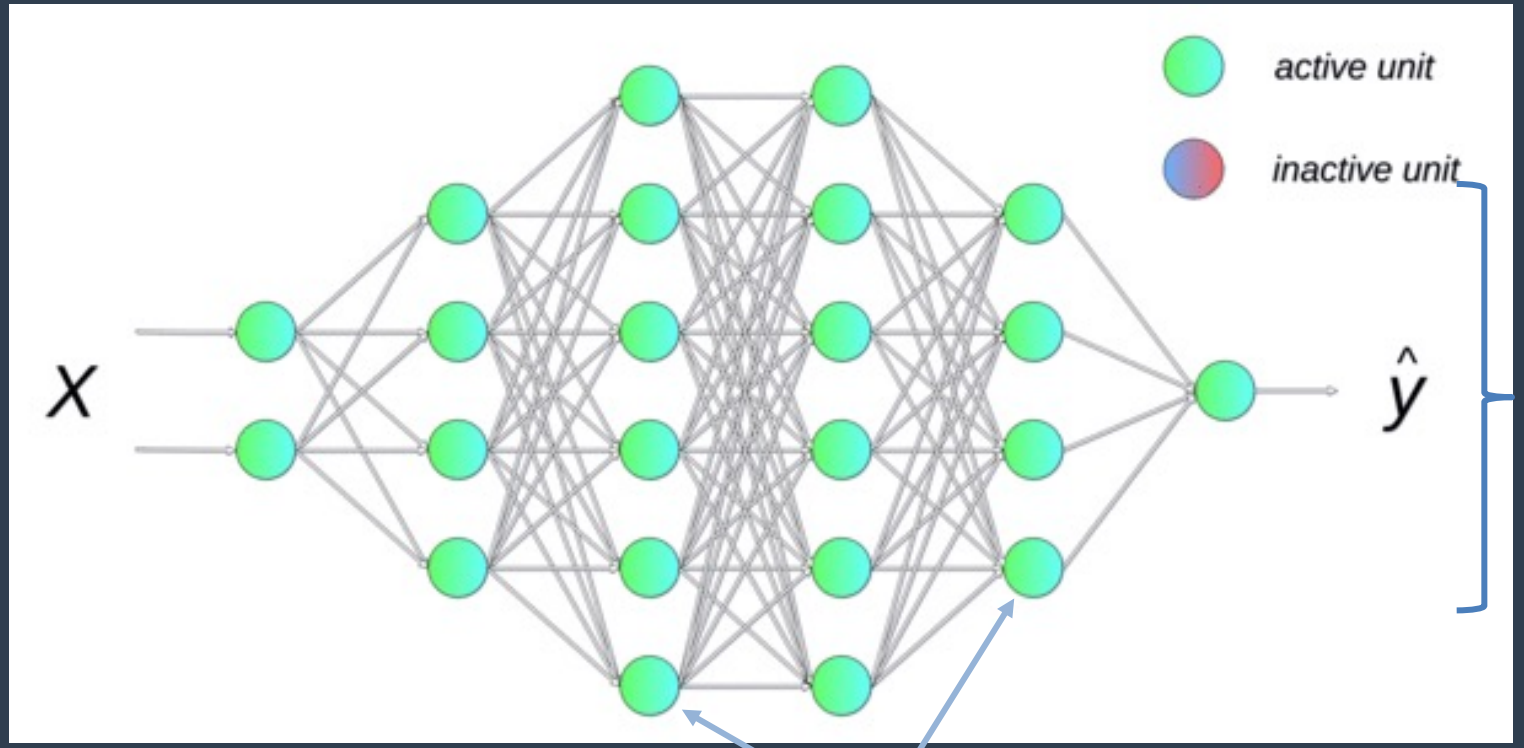
Disadvantages:

- Expensive because have to train multiple models (but can be parallelised)
- The models in the ensemble will have very different levels of accuracy and it can be non-trivial to combine them into a coherent distribution after

What if we don’t have access to a huge supercomputer?

Dropout layers

Dropout layers were originally designed to stop overfitting during training, but they can be used during the inference stage to generate an ensemble of outputs



Dropout layers

Advantages:

- Very easy to implement
- Has the advantage of also preventing overfitting in training

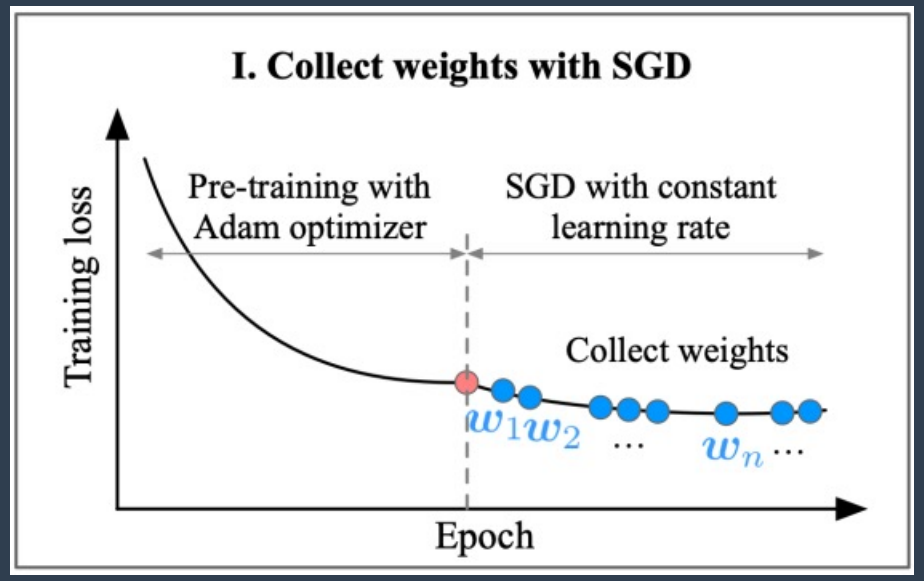
Disadvantages:

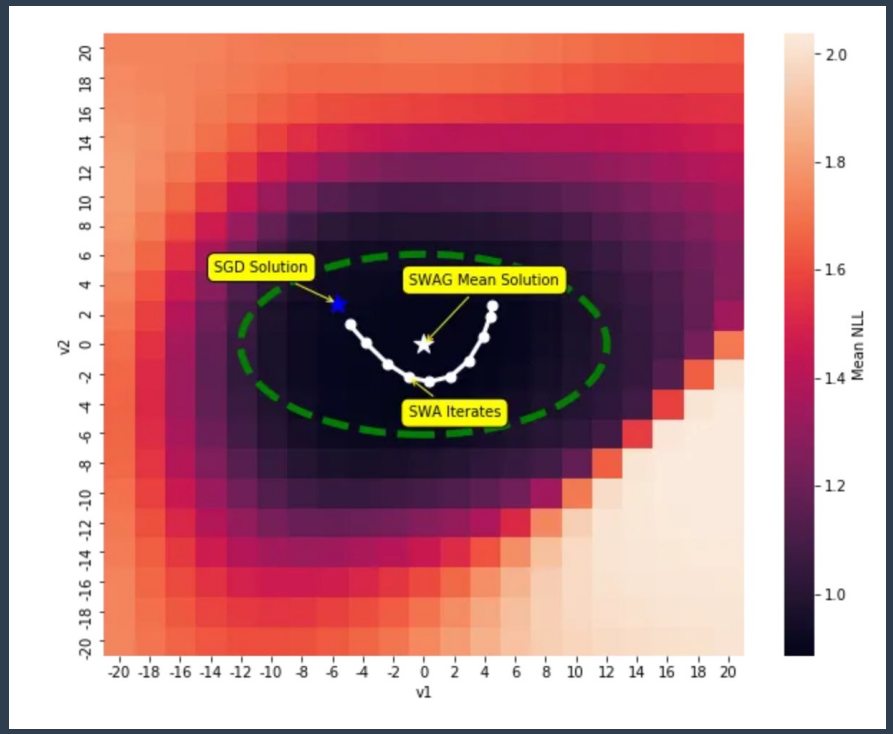
- Only quantifies epistemic uncertainty
- Often underestimates uncertainty of the model (underspread)

Stochastic Weighted Averaging (SWAG)

During training, when the optimum has almost been reached, the learning rate is pushed to explore the local geometry using Stochastic Gradient Descent (SGD).

At each epoch, a new model solution is obtained generating an ensemble of models





Morimoto, M., Fukami, K., Maulik, R., Vinuesa, R., & Fukagata, K. (2022). Assessments of epistemic uncertainty using Gaussian stochastic weight averaging for fluid-flow regression. Physica D: Nonlinear Phenomena, 440, 133454.

SWAG

Advantages:

- Uses same training procedure as a deterministic neural and has same computational cost
- Creates an ensemble of models that can be combined with test-time augmentation to quantify both epistemic and aleatoric uncertainty

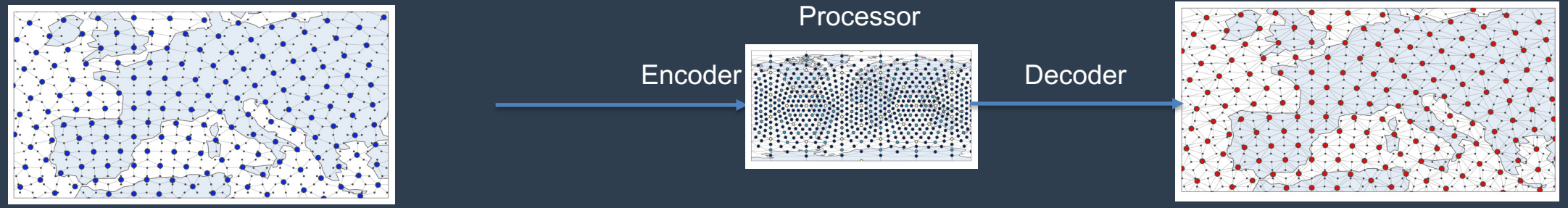
Disadvantages:

- Only works if the uncertainty in the gradient descent algorithm is gaussian
- Often underestimates the uncertainty of the problem

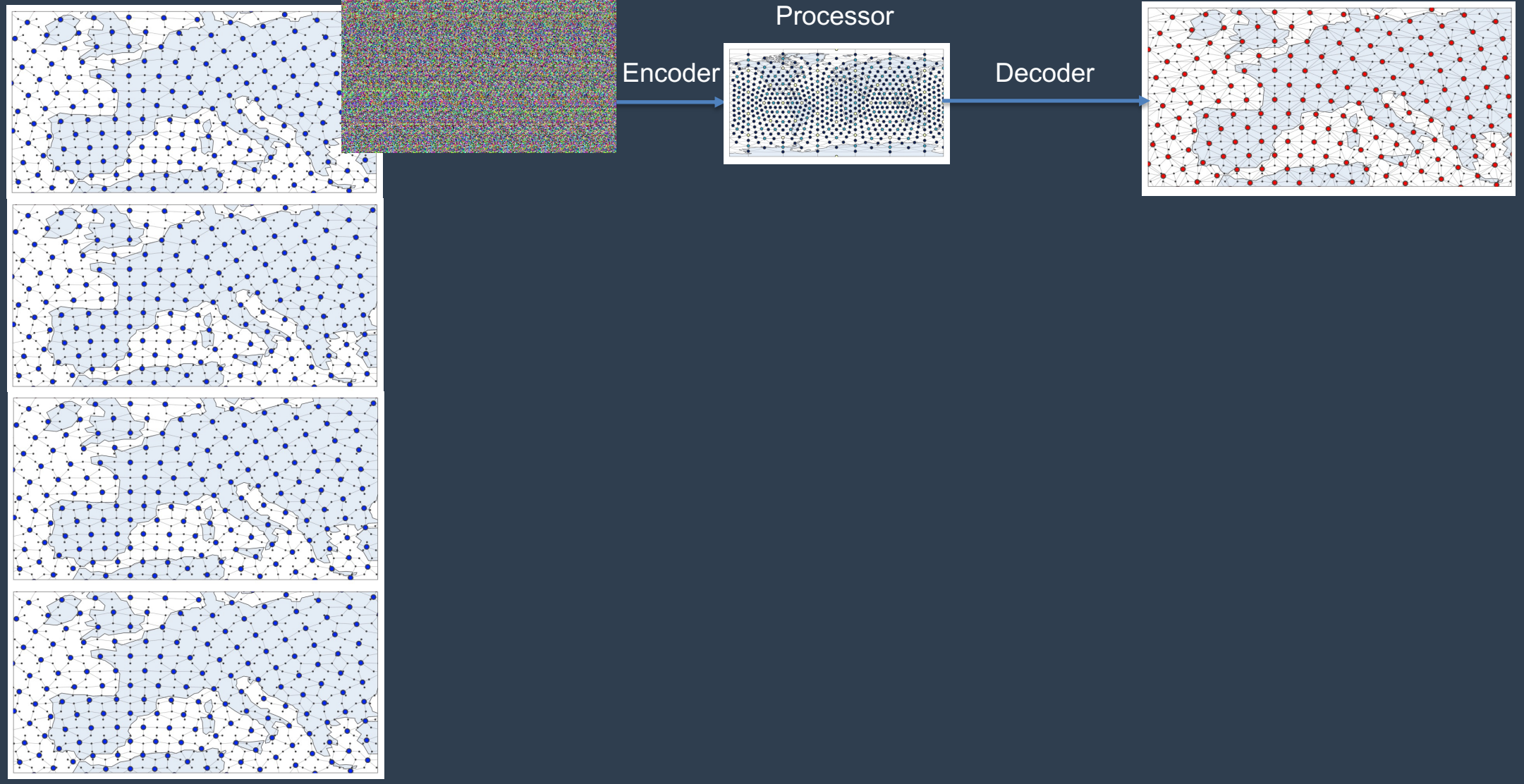
What if we train on noisy uncertain data?

ECMWF

Injecting Noise

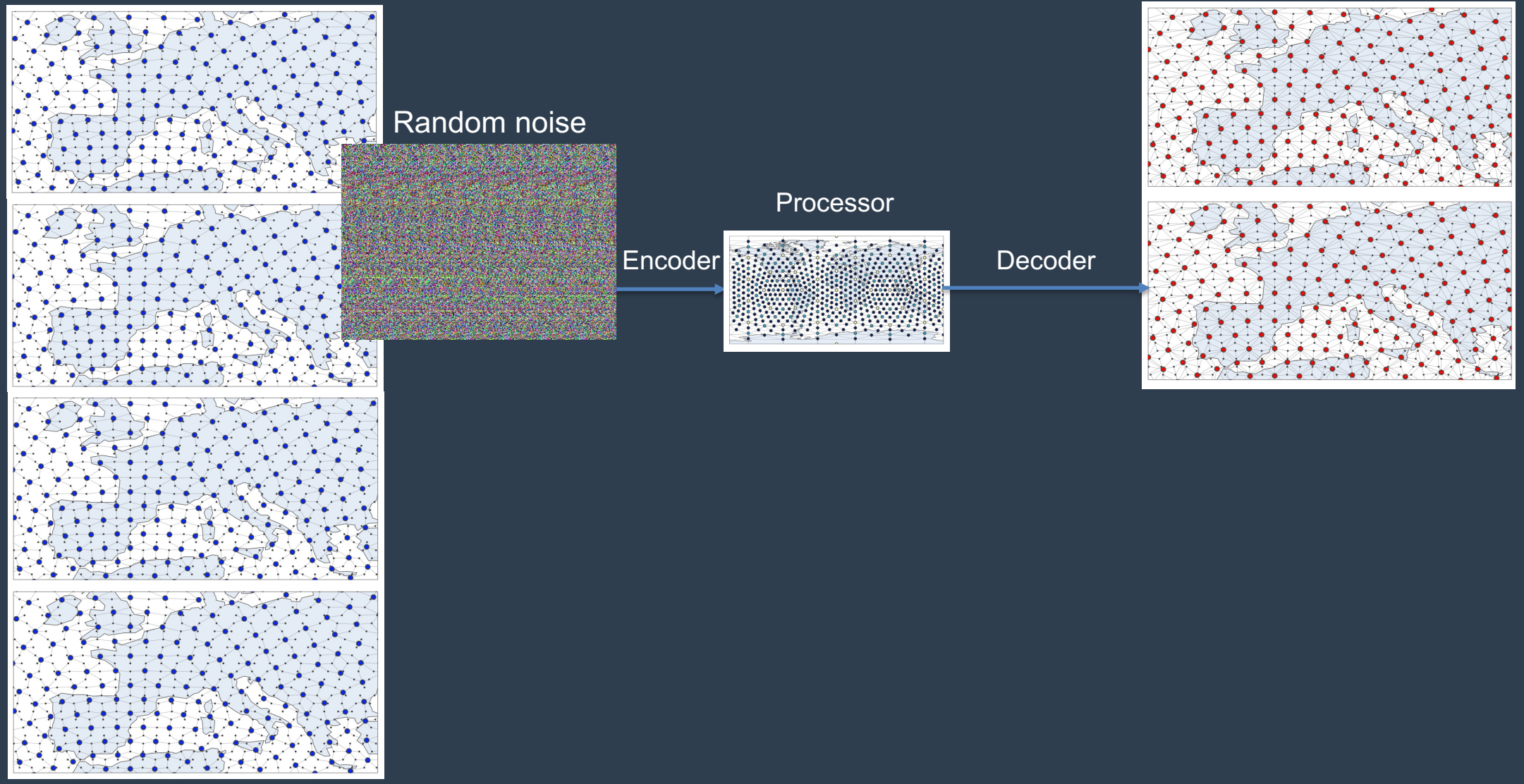


Injecting Noise



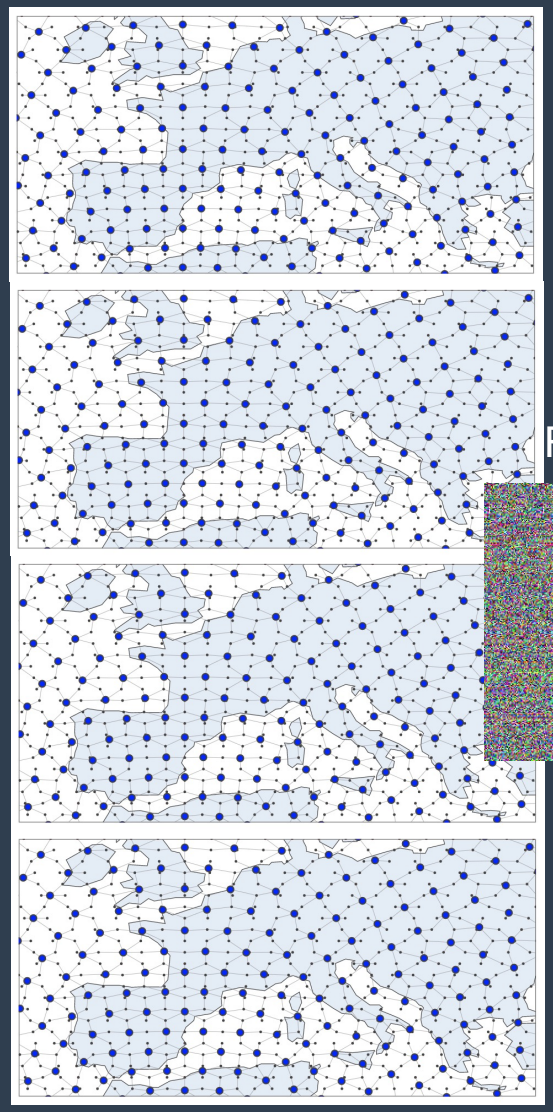
Ensemble of initial conditions

Injecting Noise



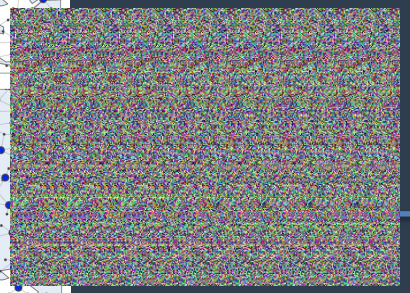
Ensemble of initial conditions

Injecting Noise



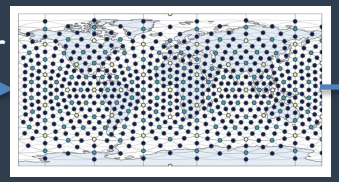
Ensemble of initial conditions

Random noise

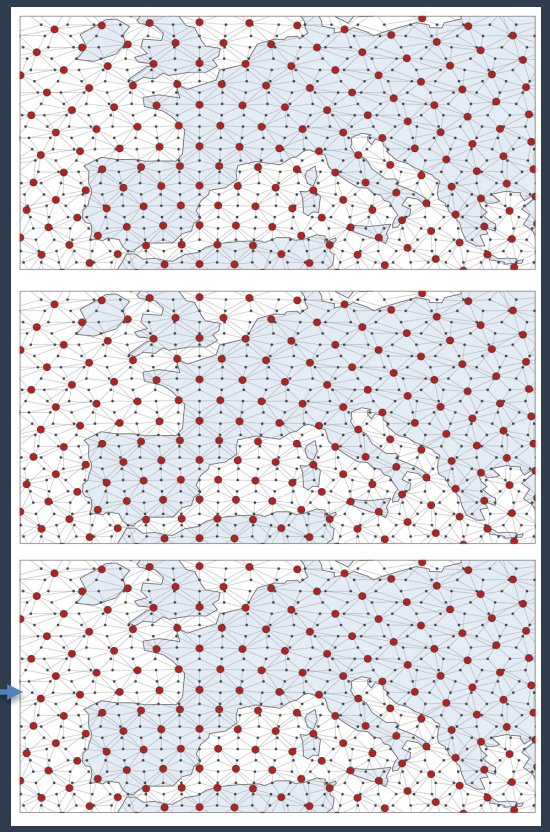


Encoder

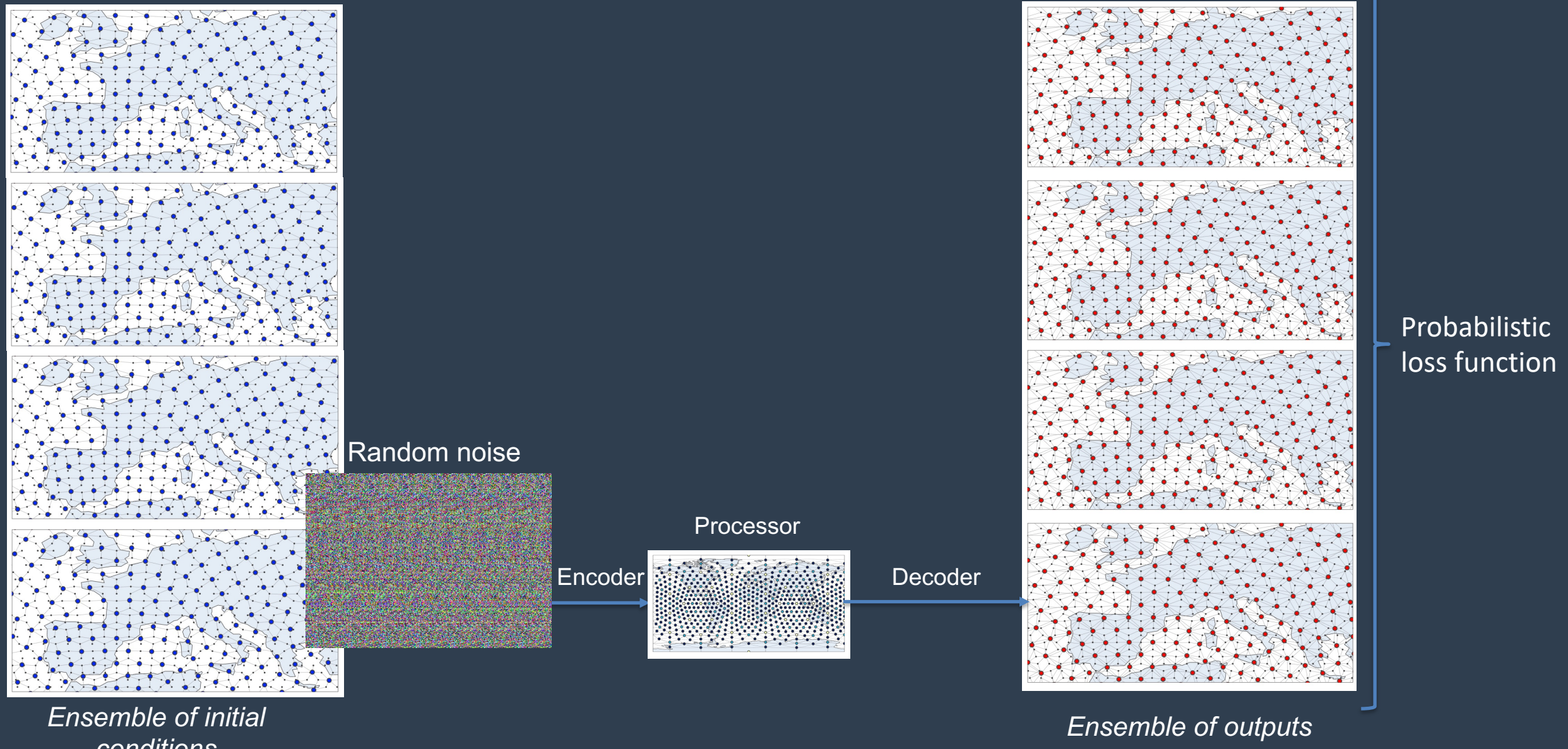
Processor



Decoder



Injecting Noise



Injection and shaping of noise

Advantages:

- Can be combined with an ensemble of initial conditions to capture both epistemic and aleatoric uncertainty
- Theoretically simple to implement if the correct loss function is chosen, because the neural network itself learns the uncertainties
- Relatively cheap to implement

Disadvantages:

- Still unknown if the noise injection is sufficient to capture all the uncertainty
- Can increase the memory usage in training (although for a big problem this memory increase is insignificant)

Probabilistic Architecture choices

ECMWF

Loss functions and entropy for probabilistic predictions

If the output is probabilistic (i.e. a distribution), then standard loss functions cannot be used.

It is common practice to instead minimise the KL-divergence loss function (difference between the real distribution and the predicted distribution)

$$D_{KL}(q||p) = \int_W q(W') \log\left(\frac{q(W')}{p\left(W'|D_{tr}\right)}\right) dW'$$

For probabilistic predictions, the entropy, H_i, can also be calculated which is a measure of the prediction uncertainty

$$H_i = -\sum_{j=1}^{N_i} p_{ij} \log(p_{ij})$$

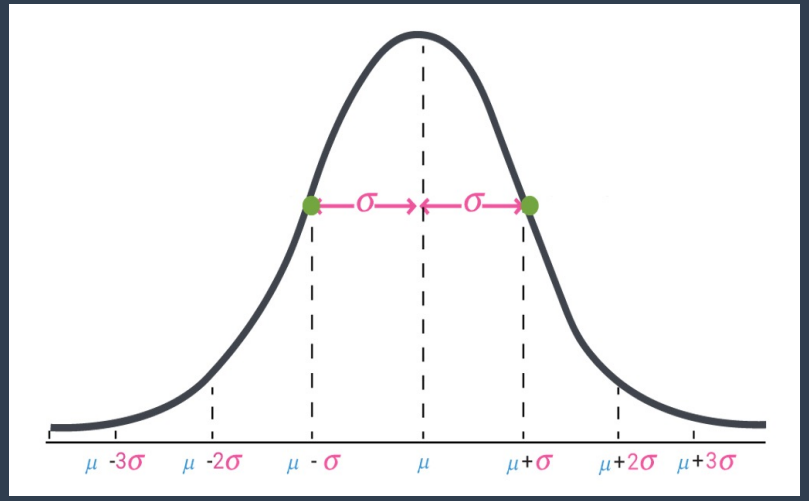
where N_i is the number of possible outcome and p_{ij} is the probability of each outcome j for sample i.

The larger the entropy the more uncertain the model is of the result

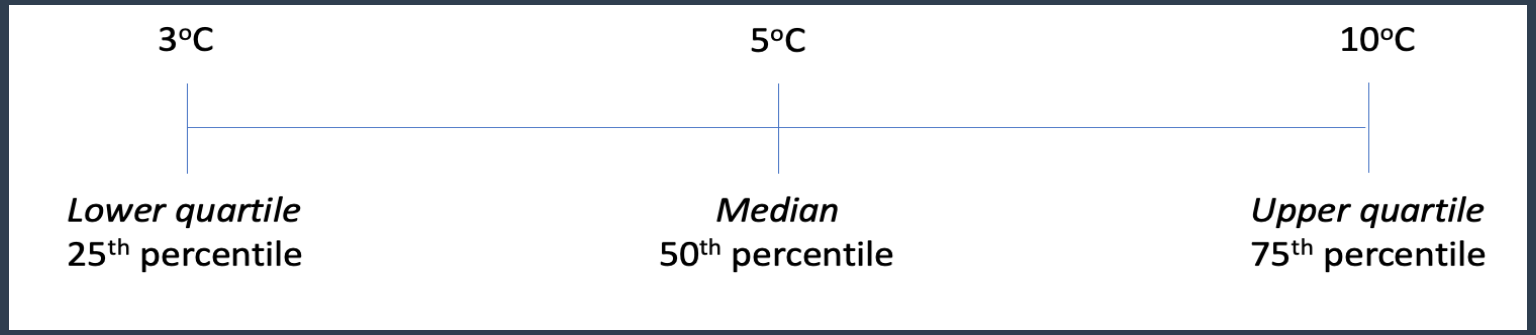
How about just predicting a gaussian?

Predict a parametric distribution

Instead of predicting a deterministic output, predict mean μ and variance σ that define a *normal distribution*



Or, for example, predict quantiles of a distribution



Predicting a parametric distribution

Advantages:

- Easy to train and training costs are roughly equivalent to those for a deterministic neural network
- Can be applied to any deep learning neural network method

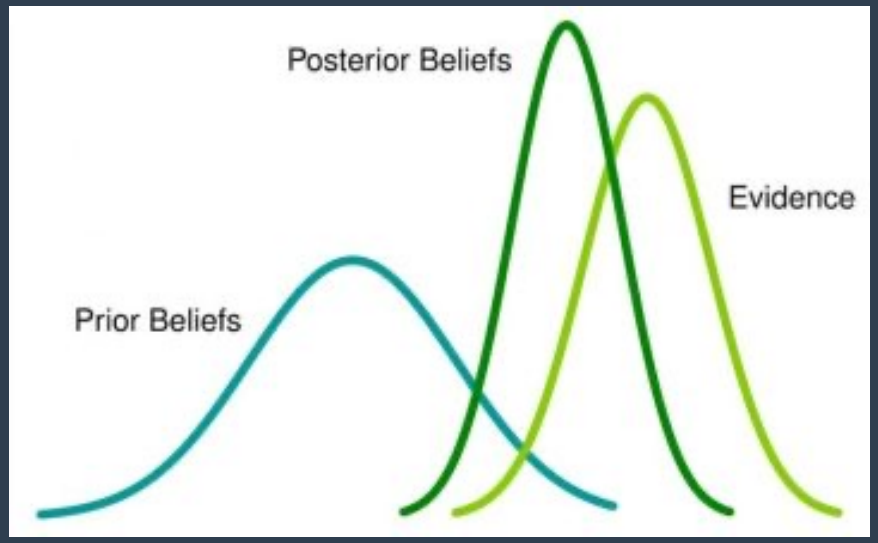
Disadvantages:

- Requires knowledge of the output distribution which is not always known and assumptions of its shape
- In some cases, we require ensemble members rather than distributions for downstream applications

But what about an arbitrary distribution?

Bayesian Inference

During Bayesian Inference, a distribution of prior beliefs is ”updated” to a distribution of posterior beliefs using evidence



Bayes Rule

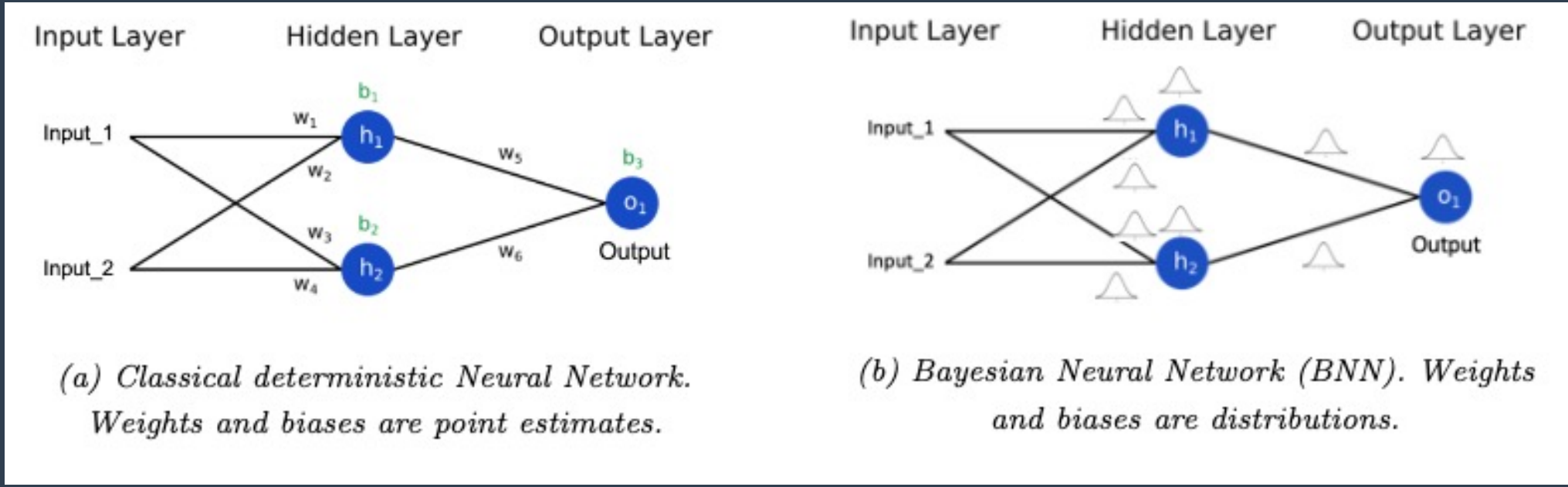
$$p\left(W|D_{tr}\right) = \frac{p(D_{tr}|W)p(W)}{p(D_{tr})}$$

where W are the network parameters, $D_{tr} = (x_n, y_n)$ the training data and $p(W)$ the prior distribution of the parameters.

Bayesian Neural Networks (BNNs)

A BNN is different from a deterministic neural network because the weights and bias are distributions rather than point estimates. These distributions are determined using Bayesian Inference

Below a simple example of a deterministic neural network is compared to a simple example of a BNN

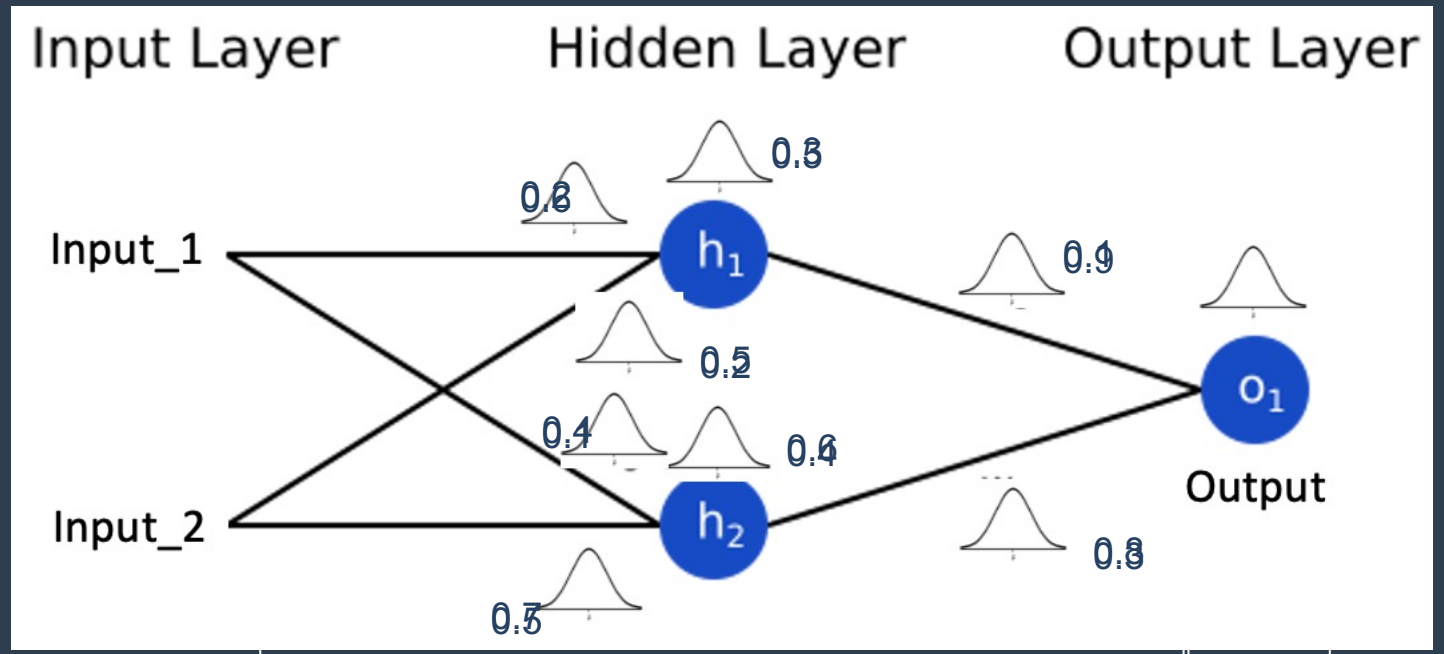


(a) Classical deterministic Neural Network. Weights and biases are point estimates.

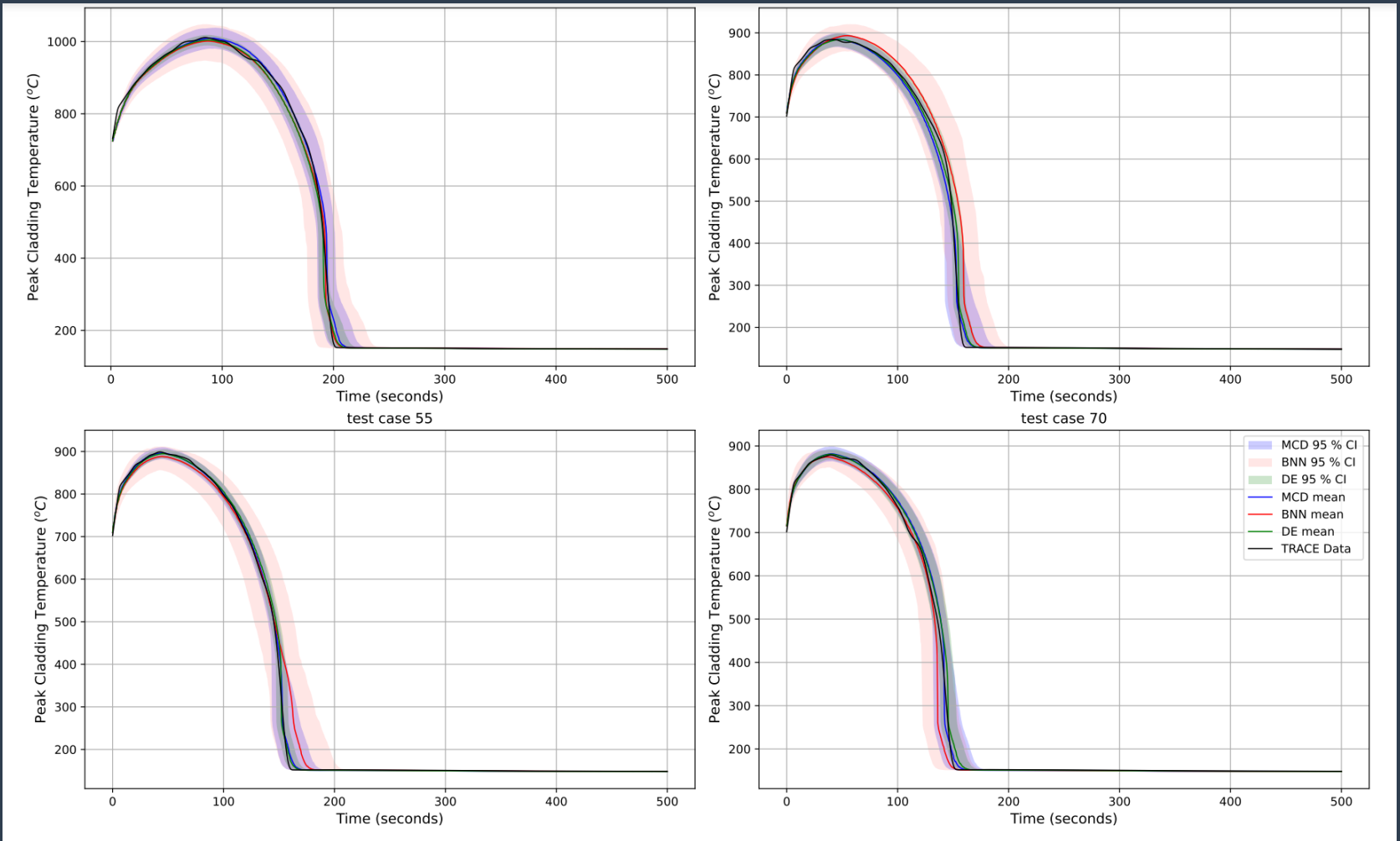
(b) Bayesian Neural Network (BNN). Weights and biases are distributions.

Bayesian Neural Networks (BNNs)

The distributions on the hidden layers make it easy to generate an ensemble of models using repeated sampling, as shown in the example below

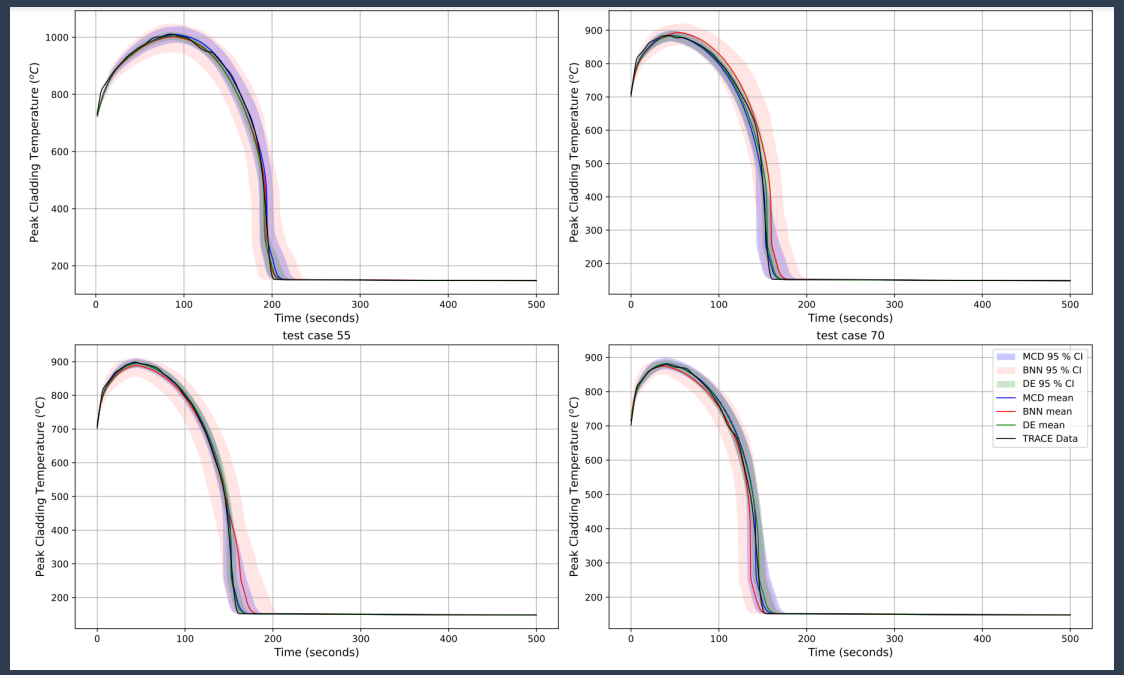


BNN vs Dropout vs Deep Ensemble



Yaseen, M., Xie, Z., & Wu, X. (2023, August). Uncertainty Quantification of Deep Neural Network Predictions for Time-dependent Responses with Functional PCA. In *In Proceedings of the 20th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-20), Washington, DC, USA*.

BNN vs Dropout vs Deep Ensemble



Yaseen et al. (2023) show that for their problem, BNNs generally provide more uncertainty than dropout or deep ensembles.
However the mean of the distribution can also be less accurate than the mean of deep ensembles

Yaseen, M., Xie, Z., & Wu, X. (2023, August). Uncertainty Quantification of Deep Neural Network Predictions for Time-dependent Responses with Functional PCA. In *In Proceedings of the 20th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-20), Washington, DC, USA*.

Bayesian Neural Networks (BNNs)

Advantages:

- Capture both aleatoric and epistemic uncertainty
- Costs only twice as much as training a deterministic neural network
- Can be used in conjunction with most deep learning methods

Disadvantages:

- Can be difficult to train
- Good support for BNNs exists in Tensorflow but not in PyTorch
- Have to be carefully tuned to ensure they do not predict too much uncertainty

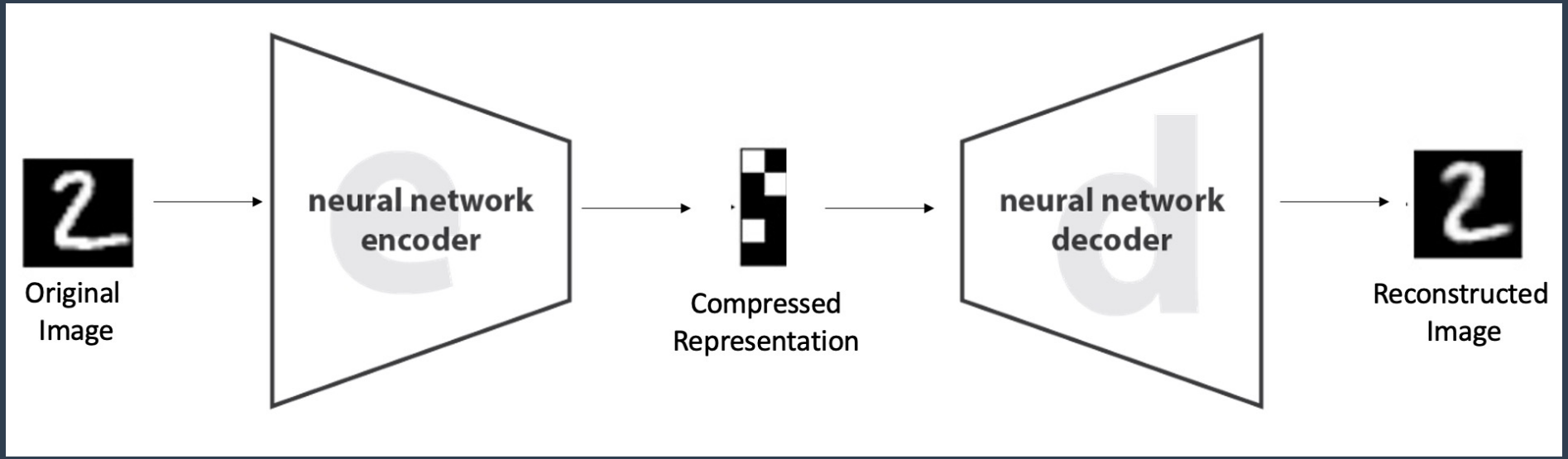
Can we capture uncertainty using generative models?

ECMWF

Variational Autoencoders

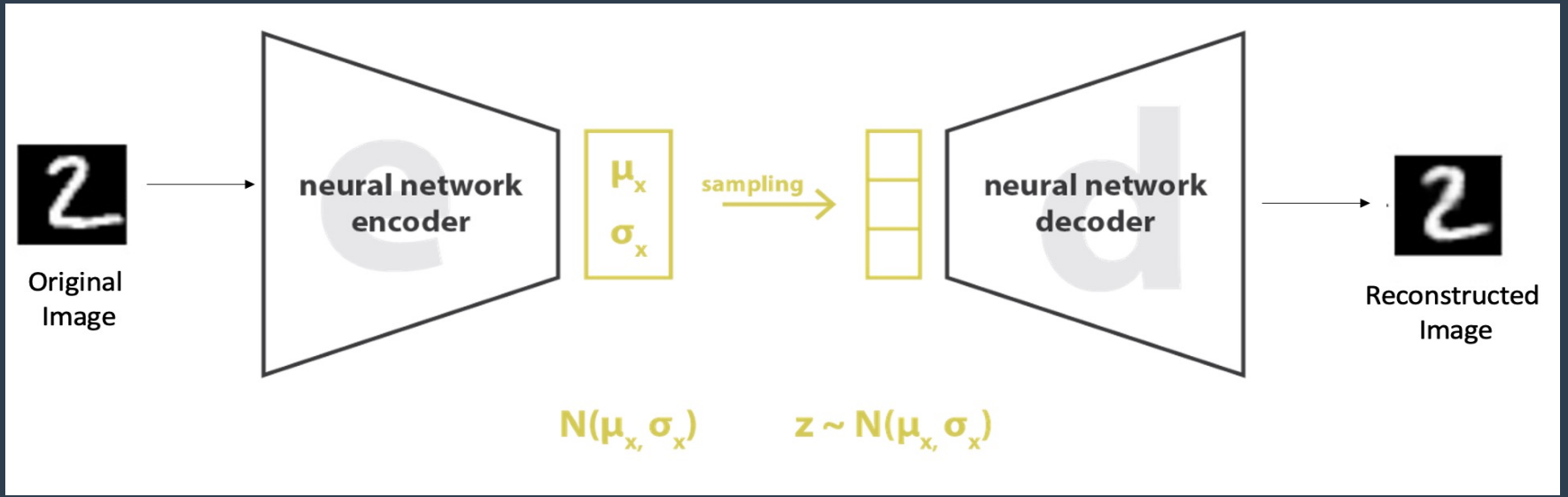
In an Autoencoder, an *encoder* encodes data from the old representation to the new one, and a *decoder* decodes the data to the old representation.

Both are neural networks trained to learn the optimum encoding and decoding



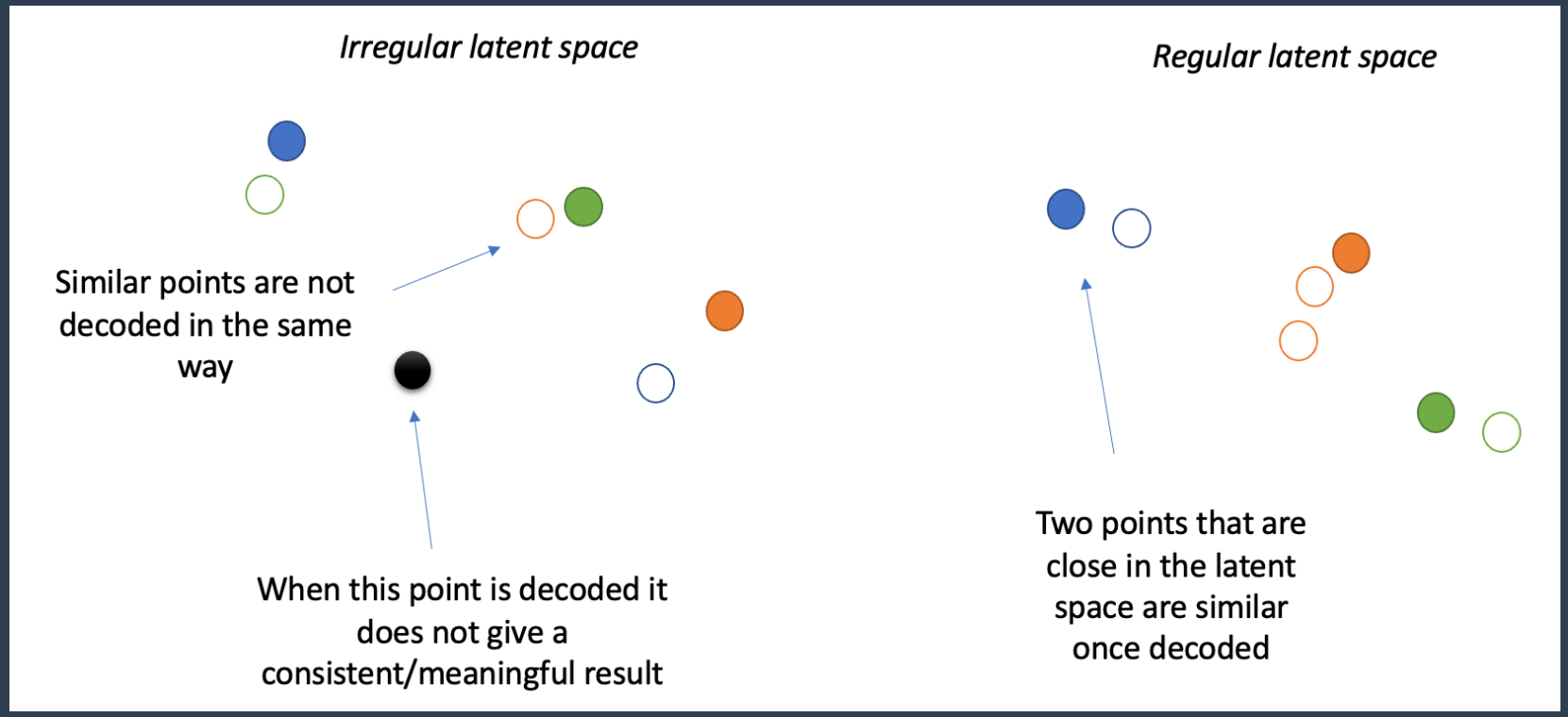
Variational Autoencoders

Variational autoencoders encode a distribution instead of a single point



Variational Autoencoders

Train the latent space so that two points close together are similar once decoded



Once trained, we can repeatedly sample from the latent space. Decoding the points using the decoder means we can then generate an ensemble

Variational Autoencoders

Advantages:

- Can generate new ensembles members that can be used in downstream applications
- Can use deterministic loss functions instead of having to consider probabilistic ones

Disadvantages:

- Like other generative methods, ensemble members generated may not be physical
- Training costs can be high

Summary

ECMWF

We have considered multiple different methods to quantify uncertainty:

1. Test-time augmentation
2. Dropout Layers
3. Deep Ensembles
4. SWAG
5. Predicting parametric distributions
6. Bayesian Neural Networks
7. Variational Autoencoders
8. Injecting and shaping noise

They each have advantages and disadvantages, which have to be considered before deciding which one(s) is the correct one for your problem