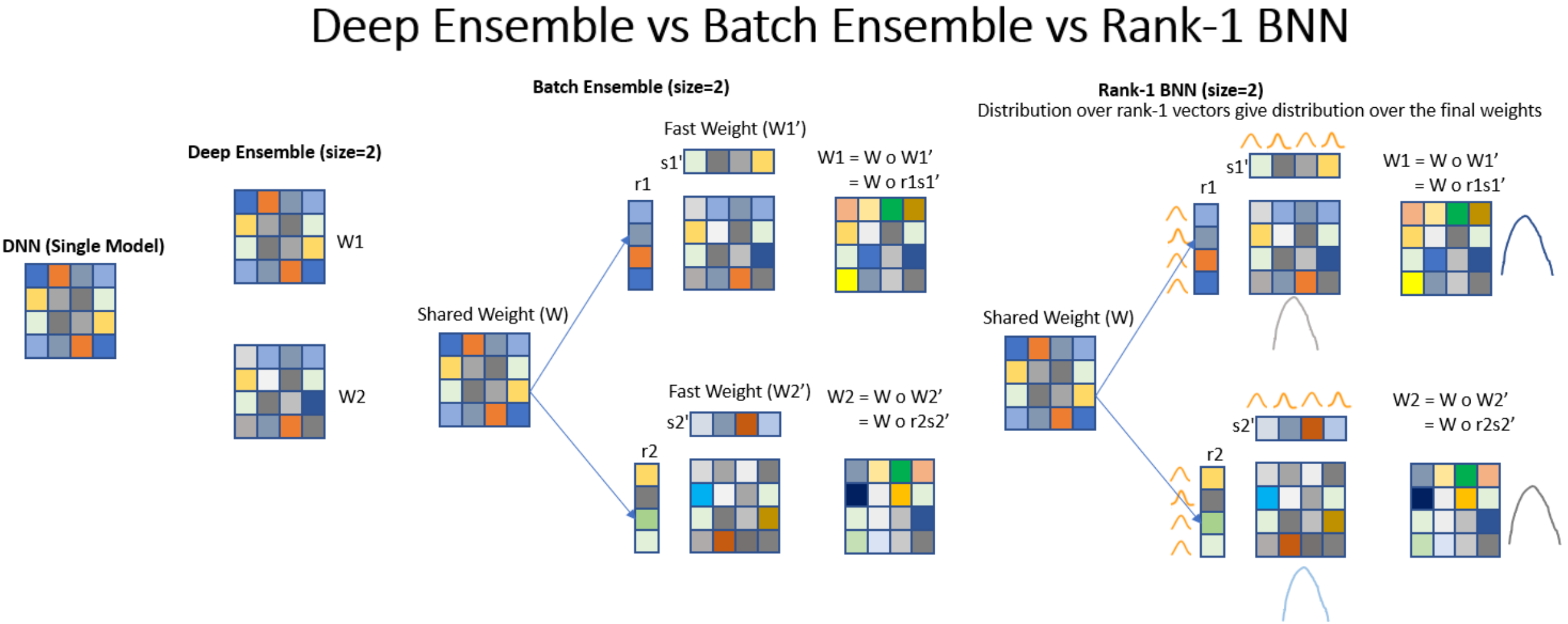
Understanding **weight matrices** and **parameters count** of different deep learning ensemble models used for **Uncertainty Estimation** – **Deep Ensemble, Batch Ensemble & Rank-1 BNN**

Anuj Shah

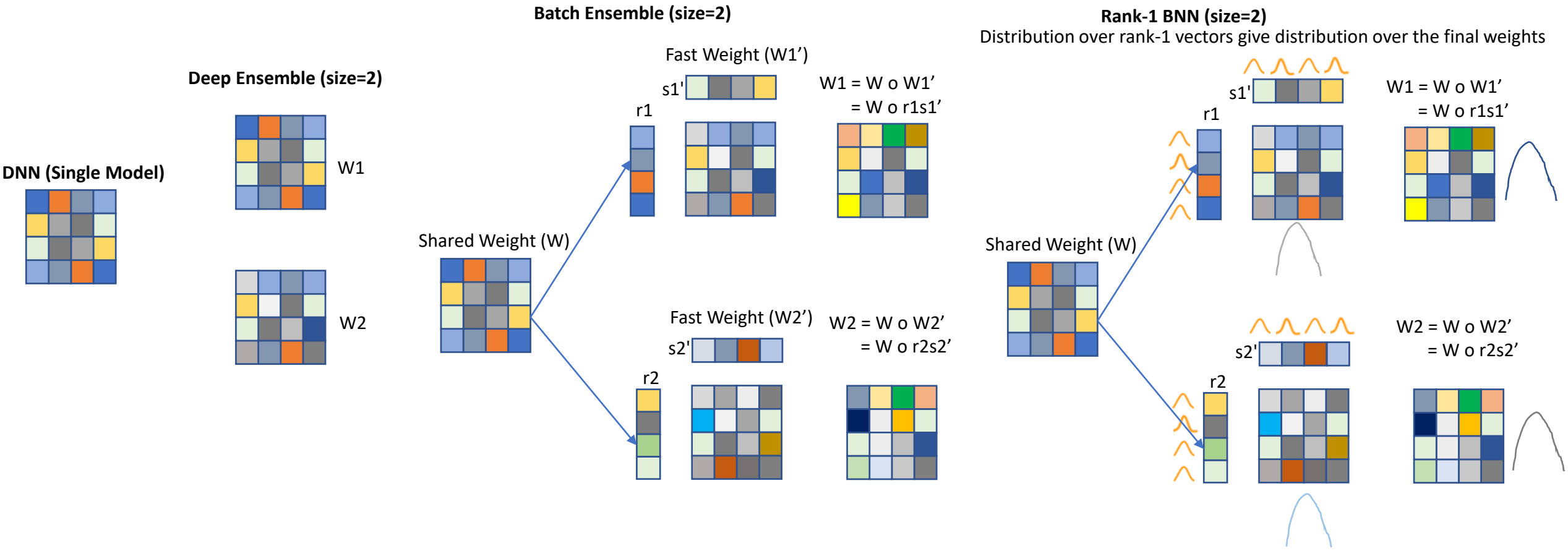
Part-1

Understanding **weight matrices** and **parameters count** of different deep learning ensemble models used for **Uncertainty Estimation**

Part-1: Theoretical understanding



Deep Ensemble vs Batch Ensemble vs Rank-1 BNN



Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles

Balaji Lakshminarayanan **Alexander Pritzel** **Charles Blundell**
DeepMind
{balajiln,apritzel,cblundell}@google.com

Abstract

Deep neural networks (NNs) are powerful black box predictors that have recently achieved impressive performance on a wide spectrum of tasks. Quantifying predictive uncertainty in NNs is a challenging and yet unsolved problem. Bayesian NNs, which learn a distribution over weights, are currently the state-of-the-art for estimating predictive uncertainty; however these require significant modifications to the training procedure and are computationally expensive compared to standard (non-Bayesian) NNs. We propose an alternative to Bayesian NNs that is simple to implement, readily parallelizable, requires very little hyperparameter

BatchEnsemble: An alternative approach to Efficient Ensemble and Lifelong Learning

Yeming Wen[1,2,3*], **Dustin Tran**[3] **& Jimmy Ba**[1,2]
[1]University of Toronto, [2]Vector Institute, [3]Google Brain

Abstract

Ensembles, where multiple neural networks are trained individually and their predictions are averaged, have been shown to be widely successful for improving both the accuracy and predictive uncertainty of single neural networks. However, an ensemble's cost for both training and testing increases linearly with the number of networks, which quickly becomes untenable.

In this paper, we propose BatchEnsemble[1], an ensemble method whose computational and memory costs are significantly lower than typical ensembles. BatchEnsemble achieves this by defining each weight matrix to be the Hadamard product of a shared weight among all ensemble members and a rank-one matrix per member. Unlike ensembles, BatchEnsemble is not only parallelizable across devices, where one device trains one member, but also parallelizable within a device, where multiple ensemble members are updated simultaneously for a given mini-batch. Across CIFAR-10, CIFAR-100, WMT14 EN-DE/EN-FR translation, and out-of-distribution tasks, BatchEnsemble yields competitive accuracy and uncertainties as typical ensembles; the speedup at test time is 3X and memory reduction is 3X at an ensemble of size 4. We also apply BatchEnsemble to lifelong

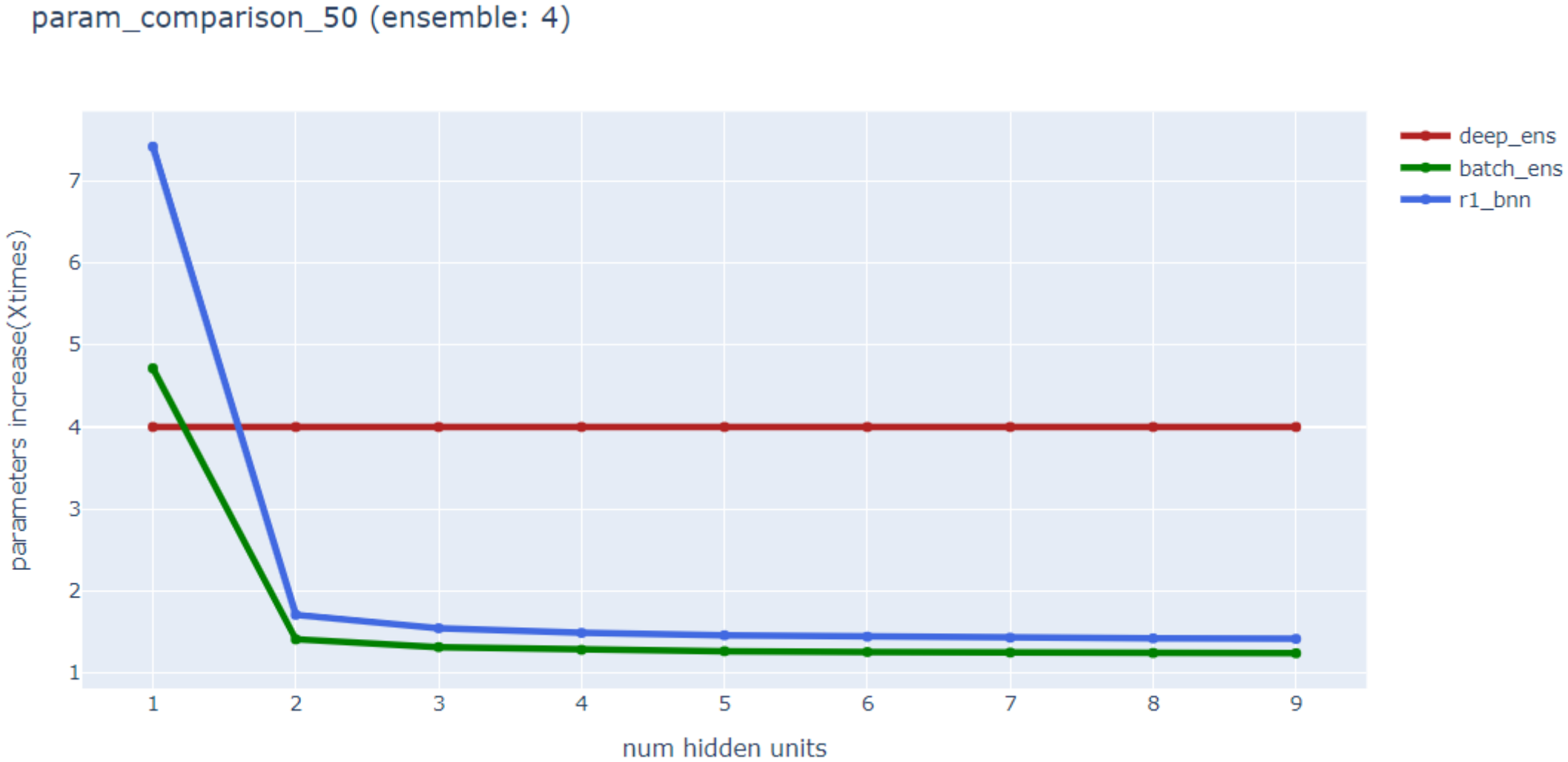
Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors

Michael W. Dusenberry [*†1] **Ghassen Jerfel** [*1 2] **Yeming Wen** [1 3] **Yi-An Ma** [1 4] **Jasper Snoek** [1]
Katherine Heller [1 2] **Balaji Lakshminarayanan** [1] **Dustin Tran** [1]

Abstract

Bayesian neural networks (BNNs) demonstrate promising success in improving the robustness and uncertainty quantification of modern deep learning. However, they generally struggle with underfitting at scale and parameter efficiency. On the other hand, deep ensembles have emerged as alternatives for uncertainty quantification that, while outperforming BNNs on certain problems, also suffer from efficiency issues. It remains unclear how to combine the strengths of these two deep learning. In principle, BNNs can permit graceful failure, signalling when a model does not know what to predict (Kendall & Gal, 2017; Dusenberry et al., 2019), and can also generalize better to out-of-distribution examples (Louizos & Welling, 2017; Malinin & Gales, 2018). However, there are two important challenges prohibiting their use in practice.

First, Bayesian neural networks often underperform on metrics such as accuracy and do not scale as well as simpler baselines (Gal & Ghahramani, 2016; Lakshminarayanan et al., 2017; Maddox et al., 2019). A possible reason is that the best configurations for BNNs remain unknown. What

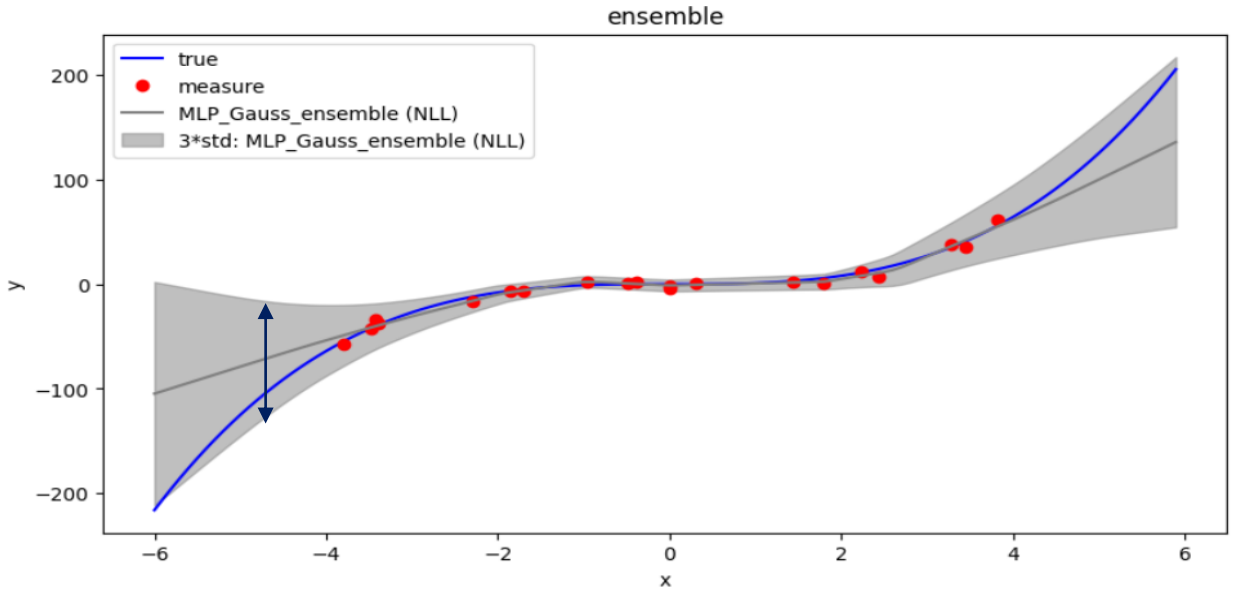

param_comparison_50 (ensemble: 4)
deep_ens
batch_ens
r1_bnn
parameters increase(Xtimes)
num hidden units
1
2
3
4
5
6
7
8
9

Uncertainty Estimation

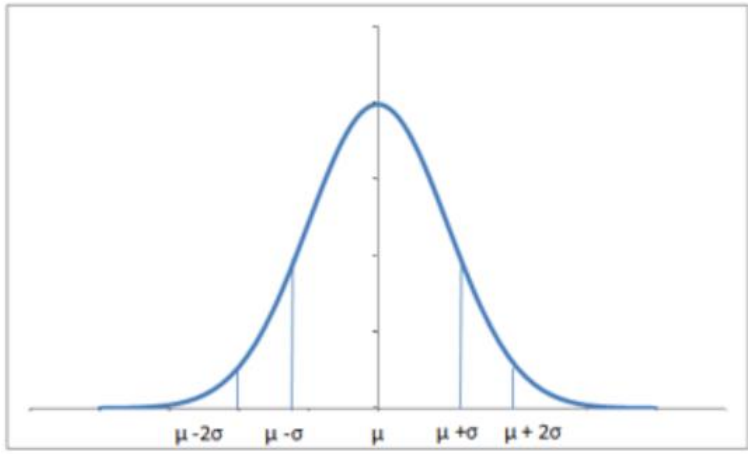
- In this video we are not going to cover in depth about the uncertainty estimation or the ensemble approaches
- We will see for the three approaches
  - On a high level how the weight matrices are defined
  - Code a simple MLP for deep ensemble , batch ensemble and rank-1 BNN
  - Compute the parameters for all the above
  - then convince us mathematically about the number of parameters in each of this paper make sense
  - and show the comparison through graph

Uncertainty Estimation

- Uncertainty estimation refers to quantifying the reliability of your prediction.
- A result can be uncertain even when it is highly confident

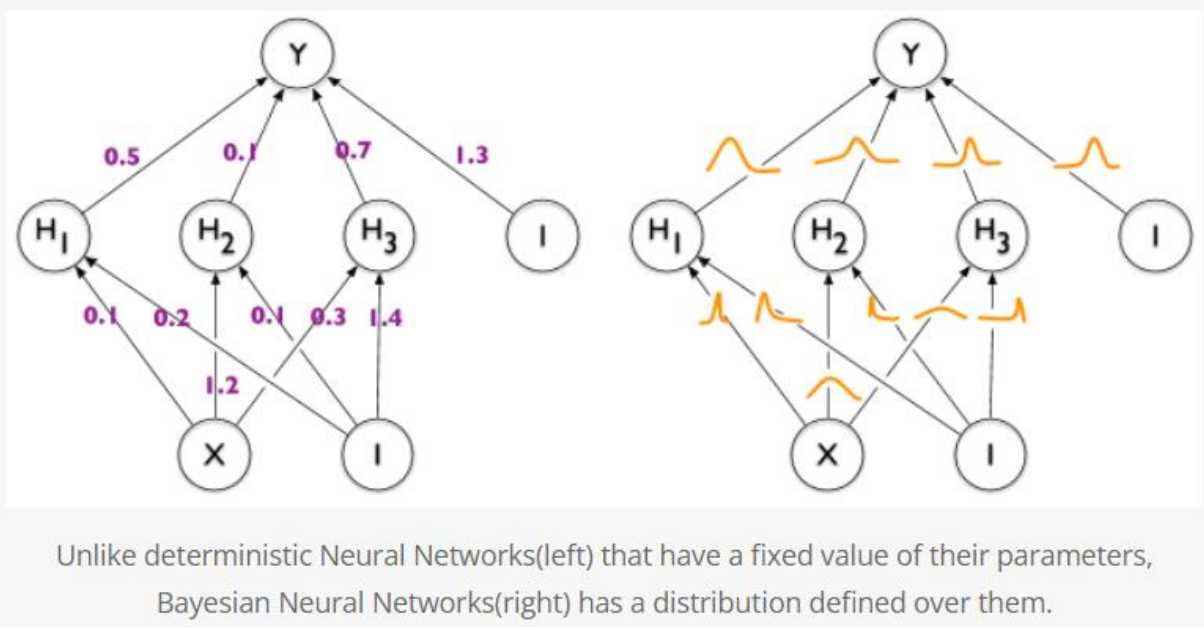


Prediction uncertainty refers to the variability in **prediction** due to plausible alternative input values.



Uncertainty Estimation

- Bayesian Neural networks can estimate predictive uncertainty but in practice BNN are harder to implement and train
- They are also computationally expensive

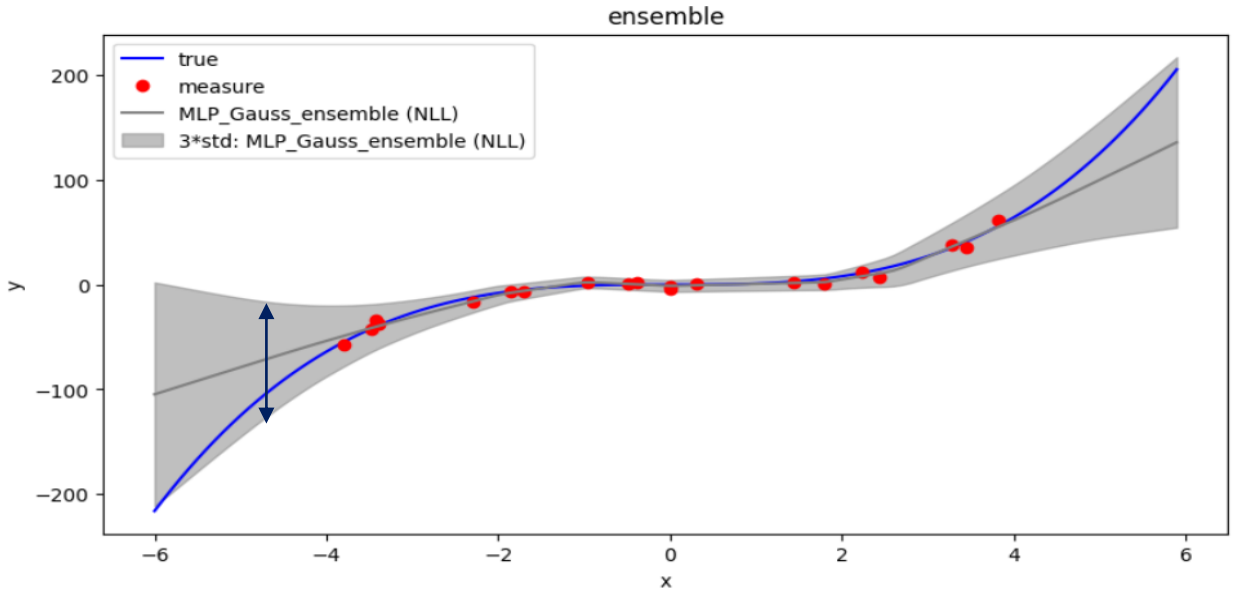


Unlike deterministic Neural Networks(left) that have a fixed value of their parameters, Bayesian Neural Networks(right) has a distribution defined over them.

Pic ref: https://sanjaykthakur.com/2018/12/05/the-very-basics-of-bayesian-neural-networks/

Uncertainty Estimation

- The Paper *"Simple & Scalable Predictive Uncertainty Estimation using Deep Ensembles – L Balaji",* showed that ensemble in deep learning is a good way of uncertainty estimation and can serve as an alternative to the Bayesian neural networks.
- Various papers followed this and proposes approaches for efficient ensemble. Two of which are – *"BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning – Yeming Wen"* & "Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors – M.W. Dusenberry & G Jerfel"

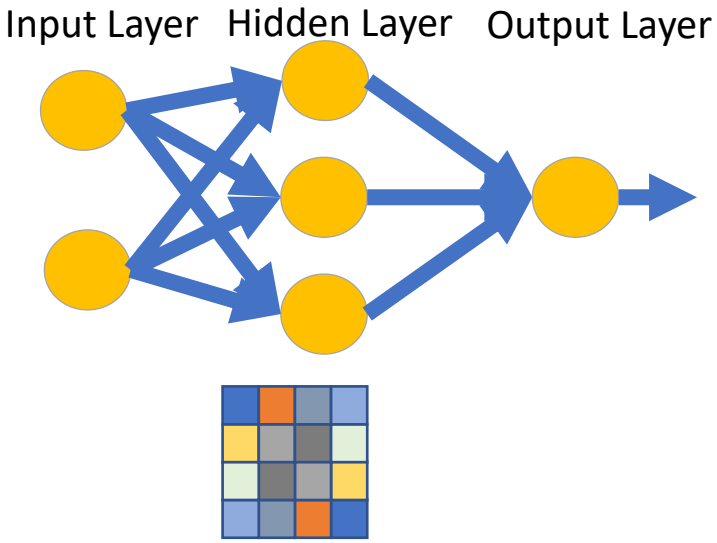


Preface

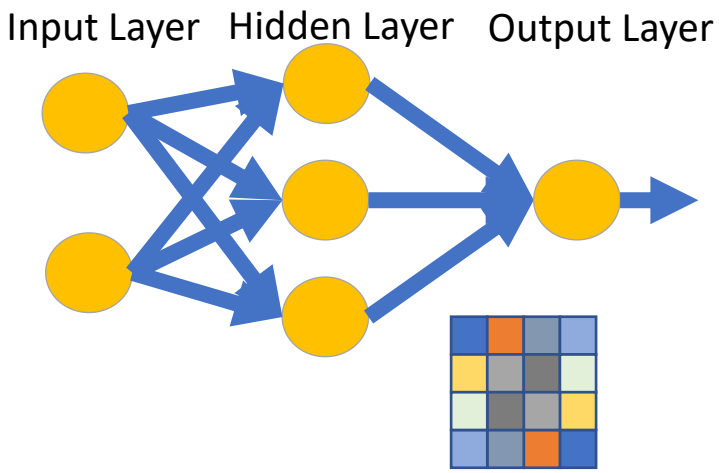
- Simple & Scalable Predictive Uncertainty Estimation using Deep Ensembles – L Balaji : **Deep Ensemble.**
- BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning – Yeming Wen : **Batch Ensemble.**
- Efficient and Scalable Bayesian Neural Nets with Rank-1 Factors – M.W. Dusenberry & G Jerfel : **Rank-1 BNN**.
- Ensemble Size in explanation -2
- Ensemble Size in code and parameter count -4

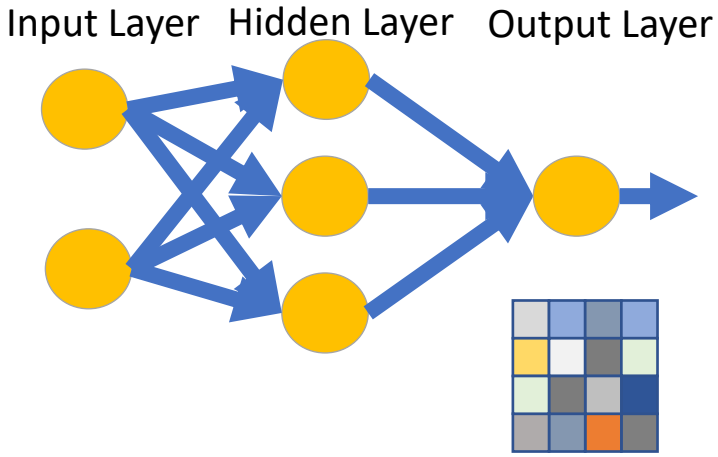
Deep Ensemble

DNN (Single Model)



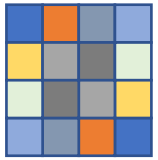
Deep Ensemble (size=2)



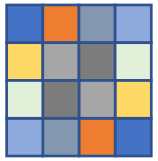


Deep Ensemble

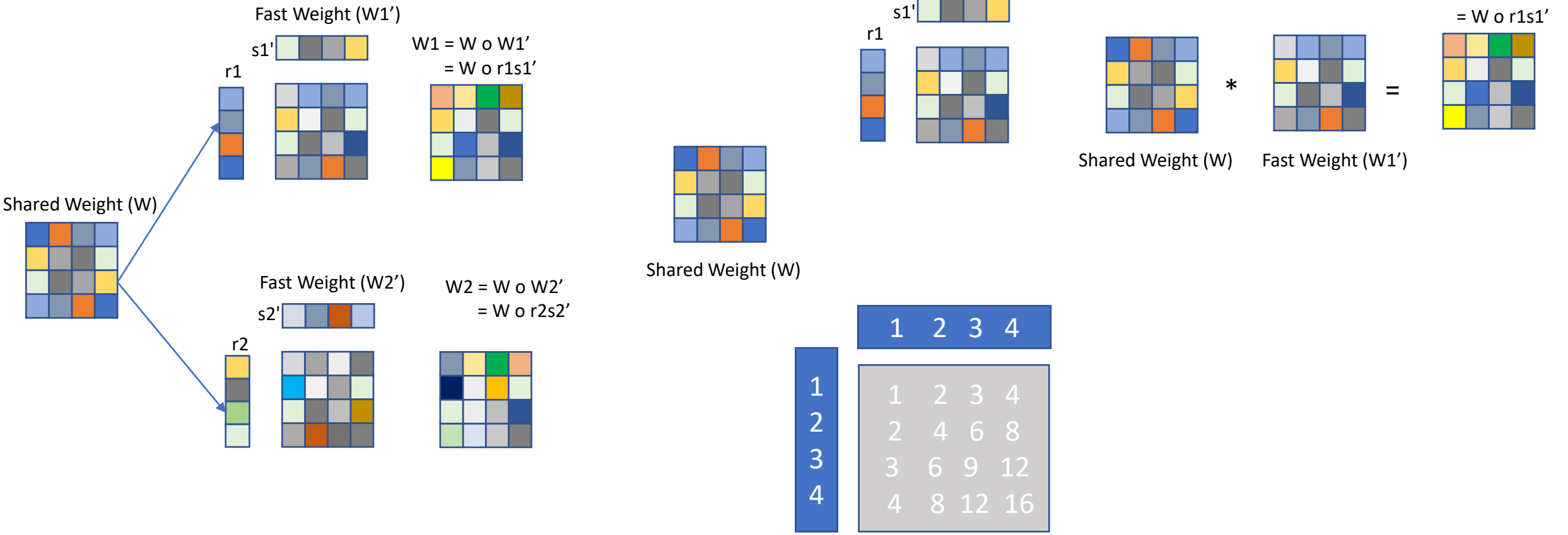
DNN (Single Model)



Deep Ensemble (size=2)

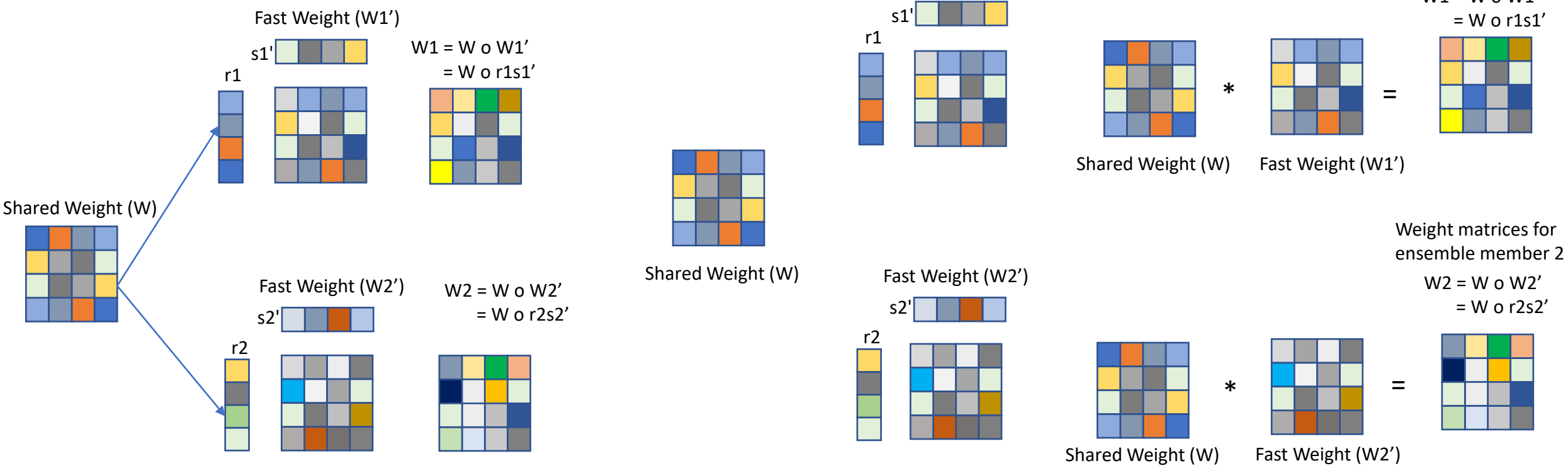


Batch Ensemble (size=2)



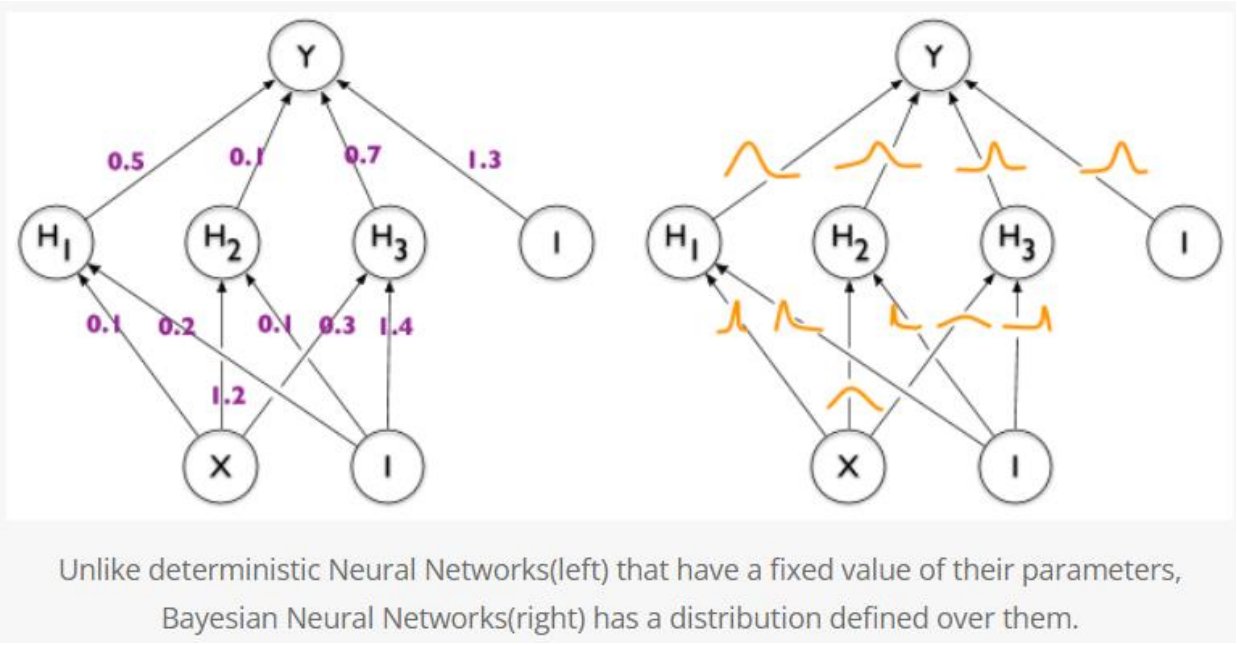
ri and si share same dimension as input and output – m & n respectively

Batch Ensemble (size=2)

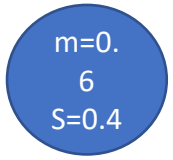


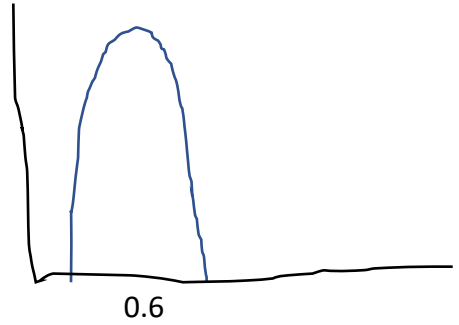
ri and si share same dimension as input and output – m & n respectively

Rank-1 BNN (size=2)



Unlike deterministic Neural Networks(left) that have a fixed value of their parameters, Bayesian Neural Networks(right) has a distribution defined over them.

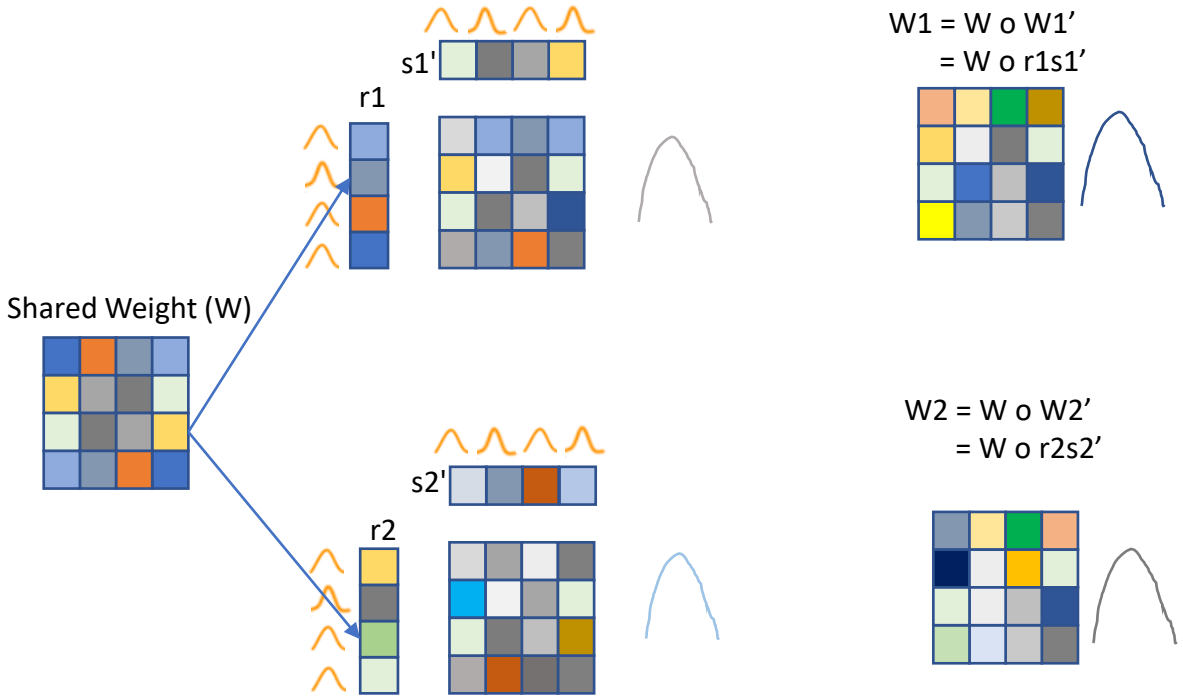




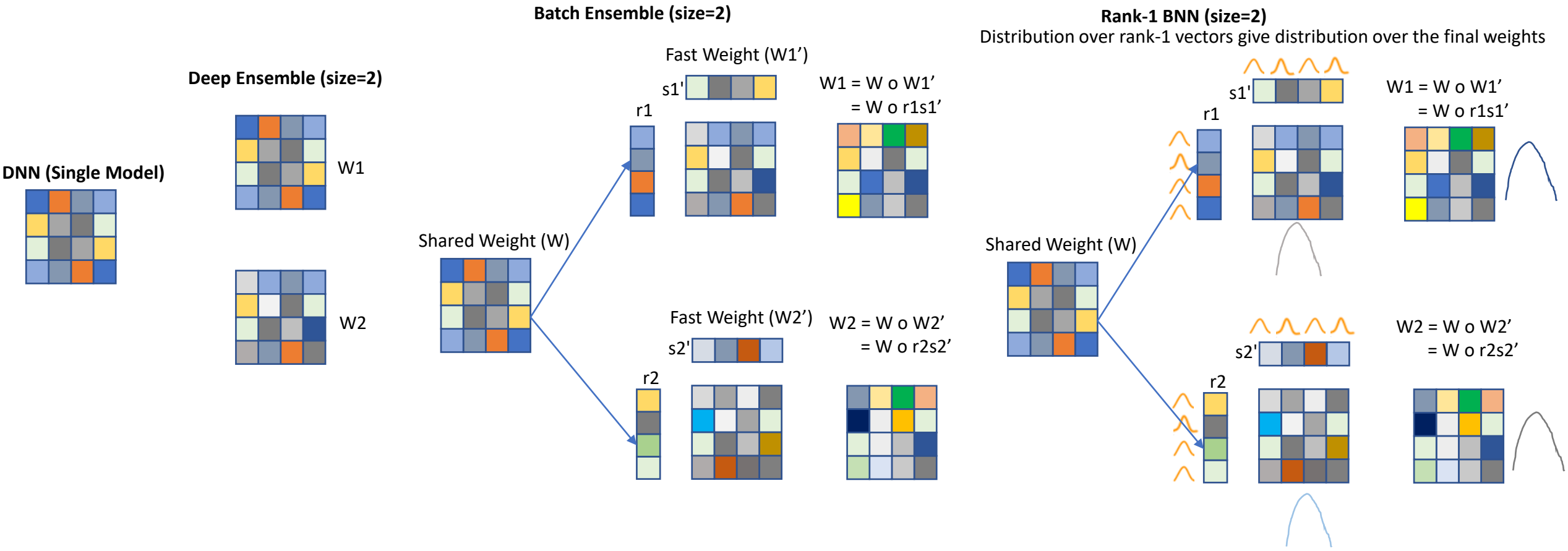
https://sanjaykthakur.com/2018/12/05/the-very-basics-of-bayesian-neural-networks/

Rank-1 BNN (size=2)

Distribution over rank-1 vectors give distribution over the final weights



Deep Ensemble vs Batch Ensemble vs Rank-1 BNN

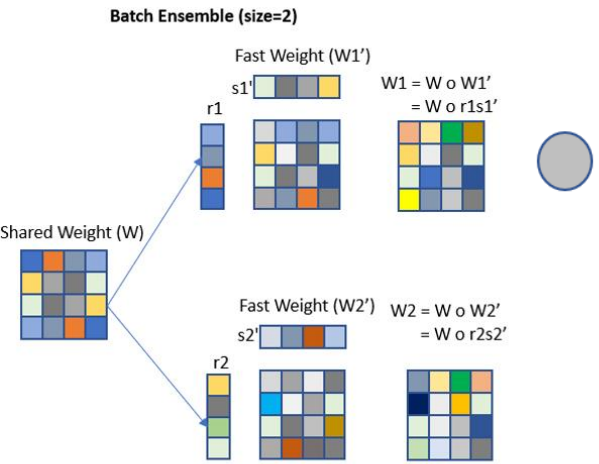


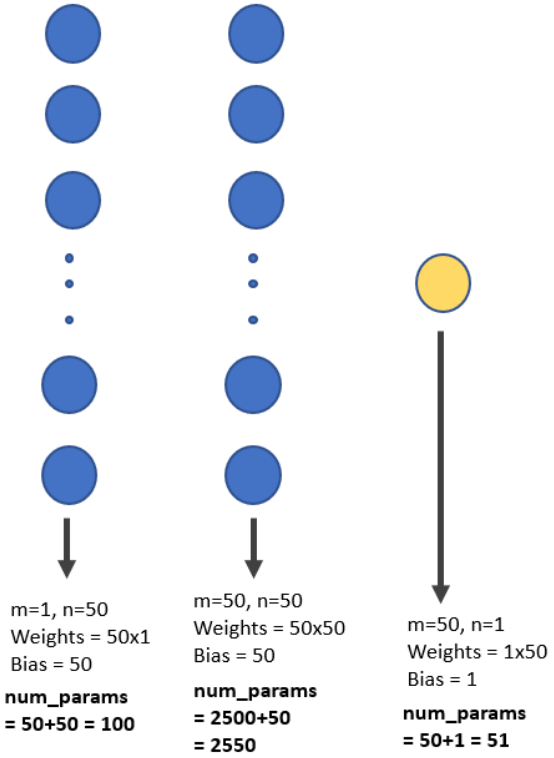
Part-2

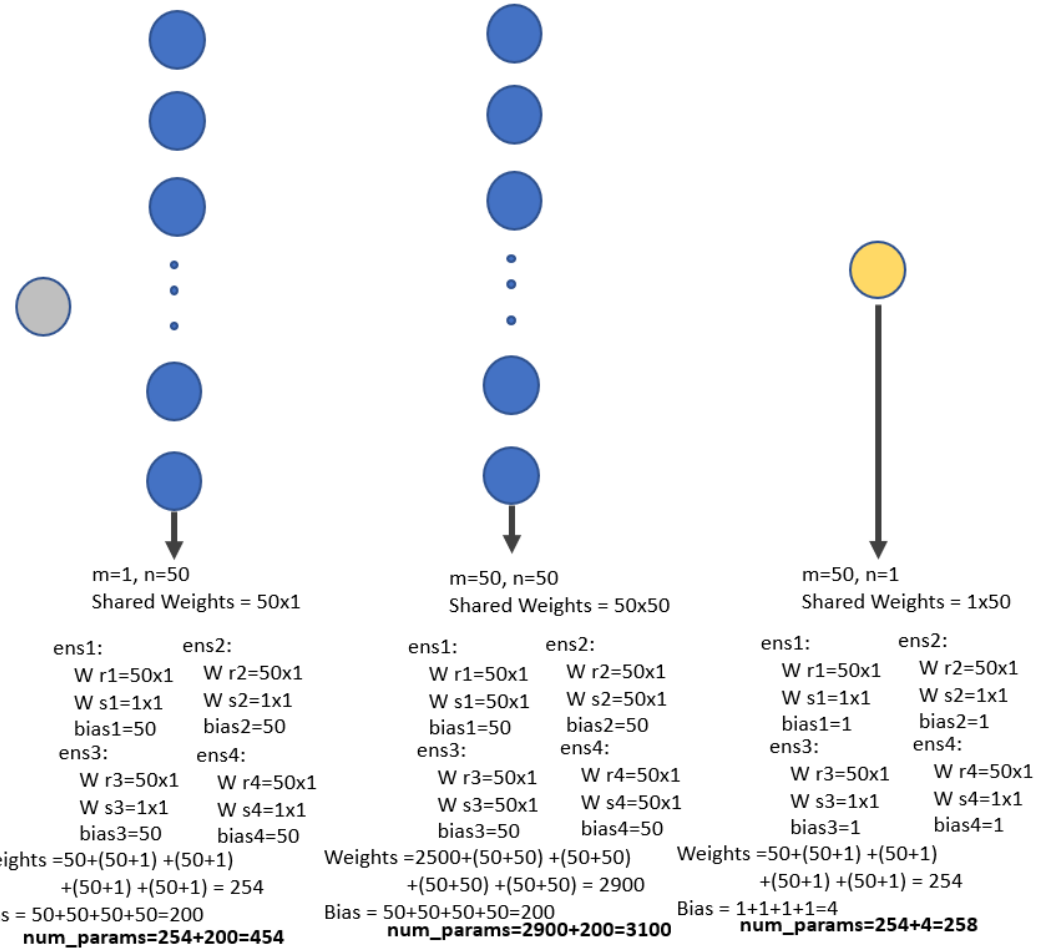
Understanding **weight matrices** and **parameters count** of different deep learning ensemble models used for **Uncertainty Estimation**

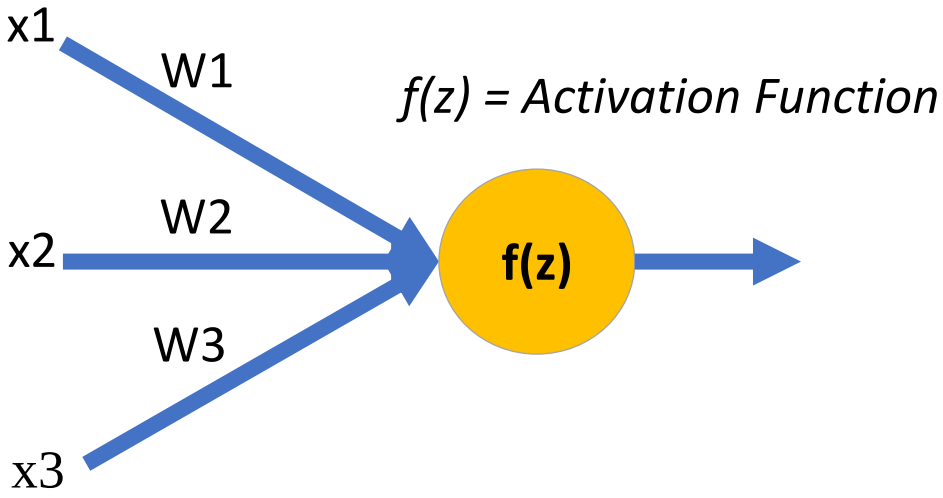
Part-2: Code and Computation

Deep Ensemble vs Batch Ensemble vs Rank-1 BNN

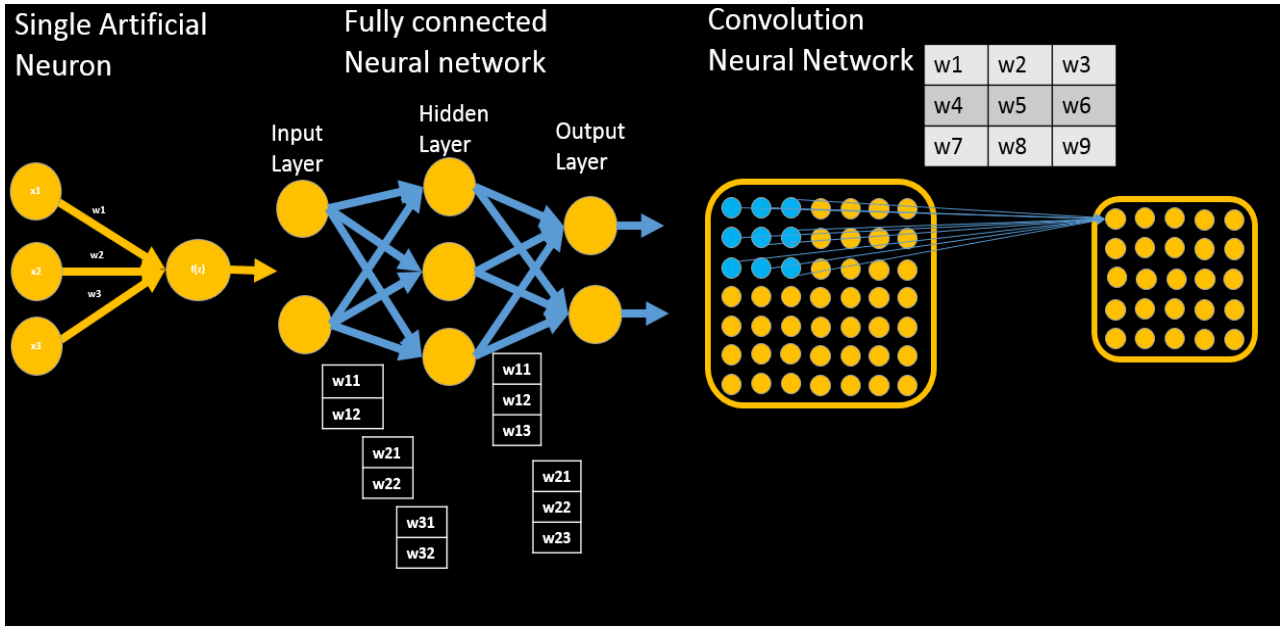




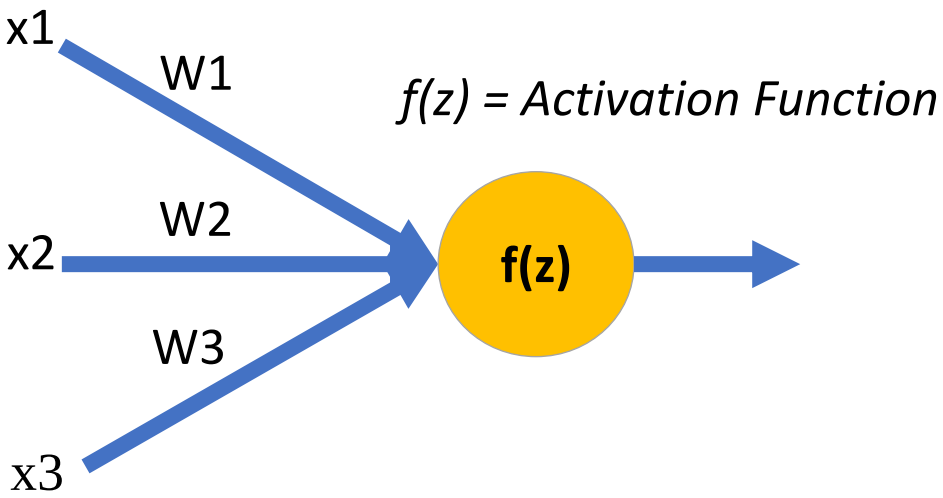


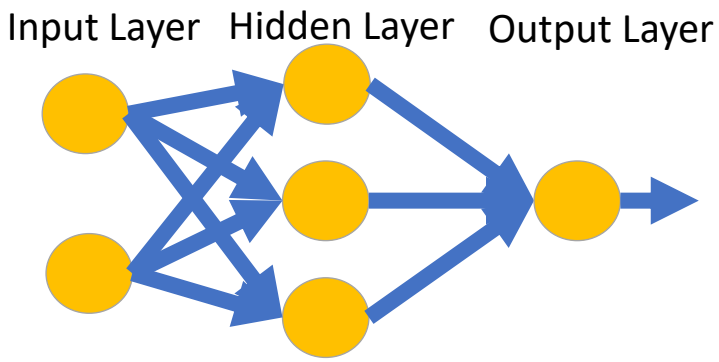


$$z = (w1*x1+w2*x2w3*x3)+b$$



Fully connected NN





$z = (w1*x1+w2*x2w3*x3)+b$

Num_params = weights + bias

For one output unit (neuron)

1. Weights = num of input unit
2. Bias = 1

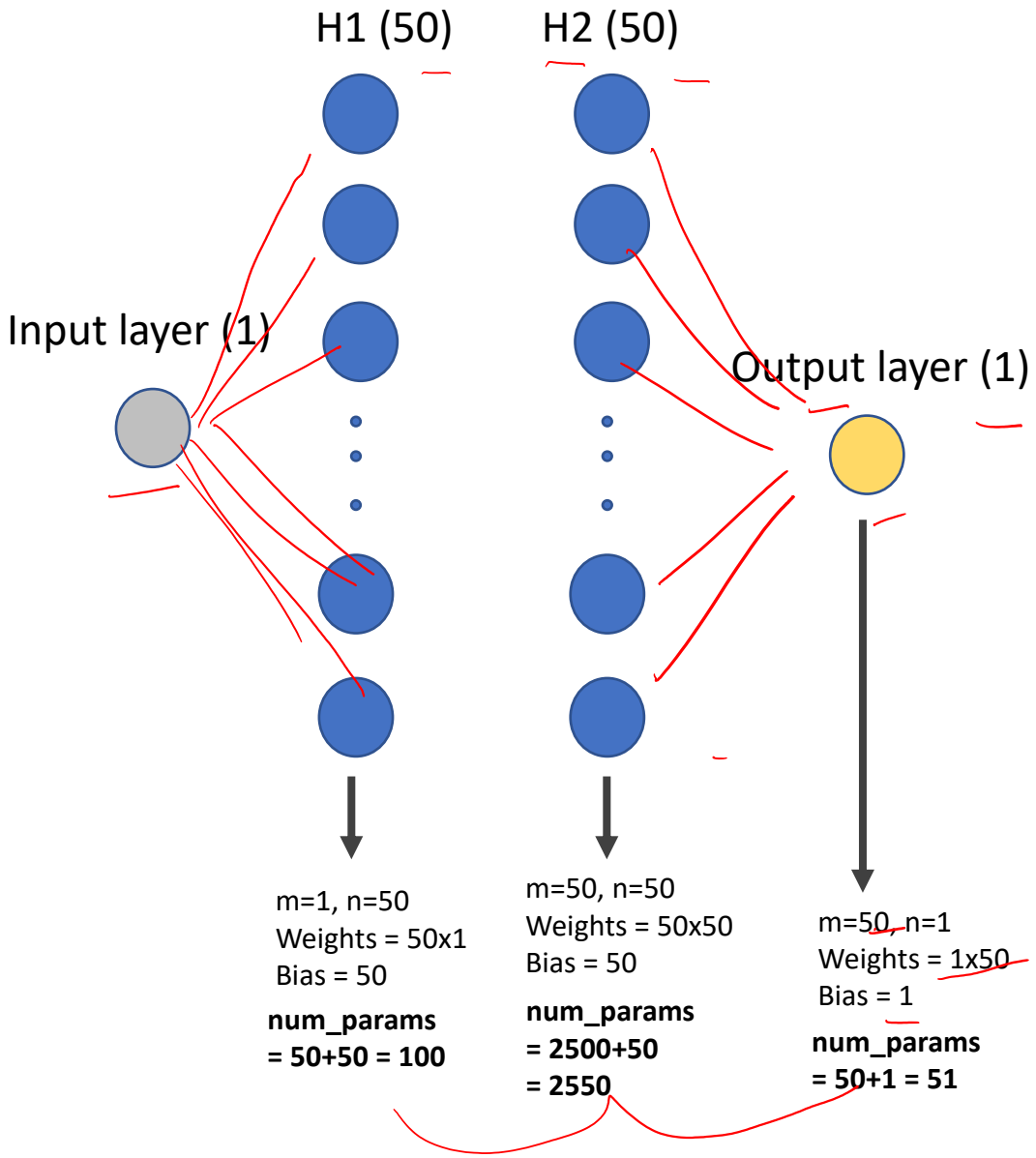
For any layer in NN

- m=num input nodes
- n=num output nodes

1. Weights = n*m
2. Bias = n

Num_param=weights+bias=n*m + n

Single model (Fully Connected NN)



For any layer in NN

- m=num input nodes
- n=num output nodes

1. Weights = n*m
2. Bias = n

Num_param=weights+bias=n*m + n

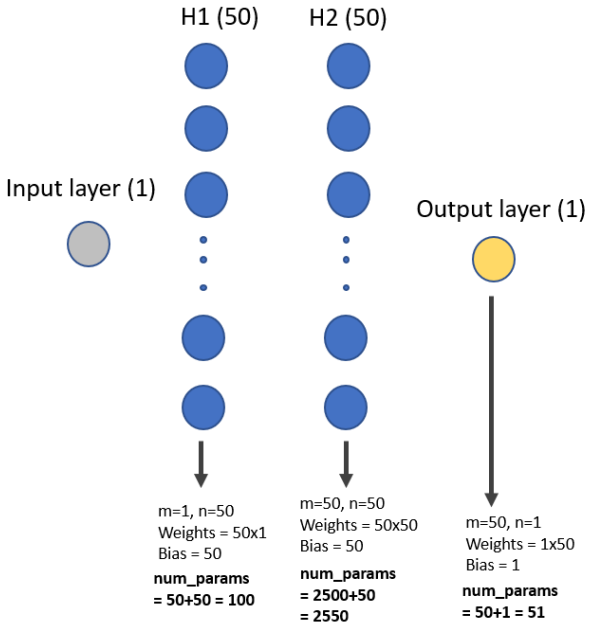
m = 1
n = 50
w = 50x1
b = 50
} 50+50 = 100

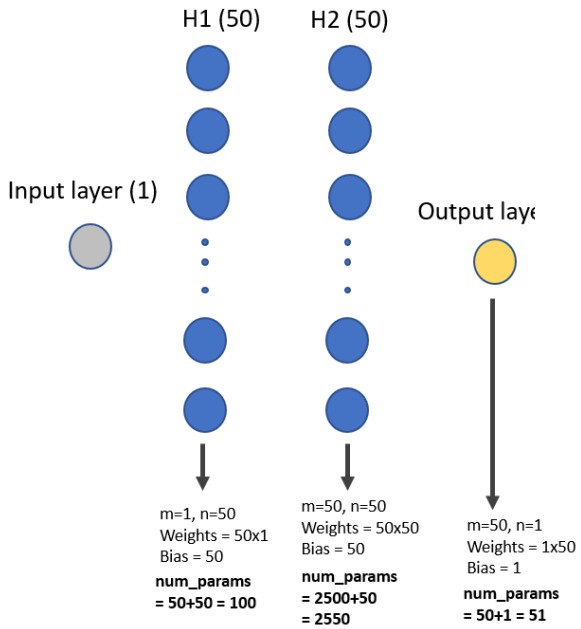
m = 50
n = 50
w = 50x50 = 2500
b = 50
} 2550

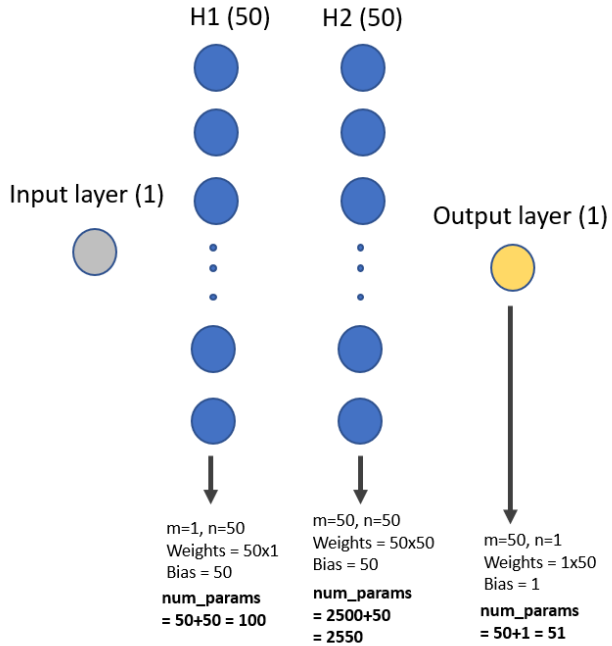
2550
1 51
2701

2701

Deep Ensemble (Ensemble size=4)





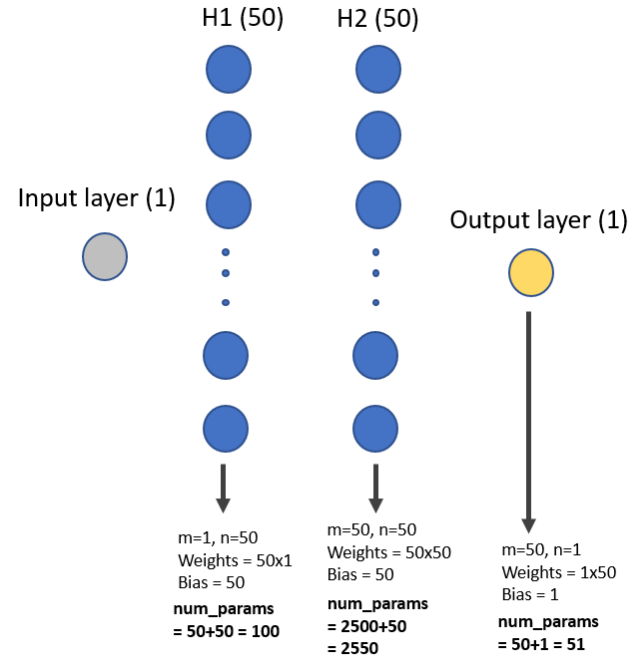


For any layer in NN

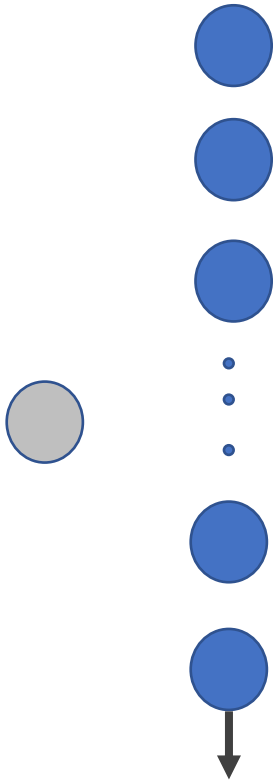
- m=num input nodes
- n=num output nodes

1. Weights = n*m
2. Bias = n

Num_param=weights+bias=n*m + n



Batch Ensemble (ensemble size = 4)



m=1, n=50
Shared Weights = 50x1

ens1:
W r1=1x1
W s1=50x1
bias1=50

ens2:
W r2=1x1
W s2=50x1
bias2=50

ens3:
W r3=1x1
W s3=50x1
bias3=50

ens4:
W r4=1x1
W s4=50x1
bias4=50

Weights =50+(50+1) +(50+1)
+(50+1) +(50+1) = 254
Bias = 50+50+50+50=200
num_params=254+200=454

m=50, n=50
Shared Weights = 50x50

ens1:
W r1=50x1
W s1=50x1
bias1=50

ens2:
W r2=50x1
W s2=50x1
bias2=50

ens3:
W r3=50x1
W s3=50x1
bias3=50

ens4:
W r4=50x1
W s4=50x1
bias4=50

Weights =2500+(50+50) +(50+50)
+(50+50) +(50+50) = 2900
Bias = 50+50+50+50=200
num_params=2900+200=3100

m=50, n=1
Shared Weights = 1x50

ens1:
W r1=50x1
W s1=1x1
bias1=1

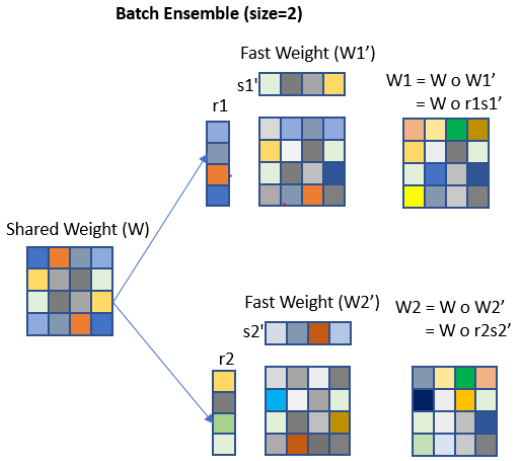
ens2:
W r2=50x1
W s2=1x1
bias2=1

ens3:
W r3=50x1
W s3=1x1
bias3=1

ens4:
W r4=50x1
W s4=1x1
bias4=1

Weights =50+(50+1) +(50+1)
+(50+1) +(50+1) = 254
Bias = 1+1+1+1=4
num_params=254+4=258

ri and si share same dimension as input and output – m & n respectively



For any layer in NN
- m=num input nodes
- n=num output nodes

1. Weights = n*m
2. Bias = n

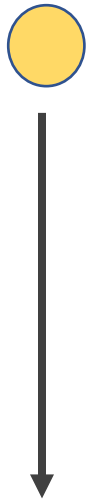
Num_param=weights+bias=n*m + n

m=1, n=50
Weights = 50x1
Bias = 50

num_params
= 50+50 = 100

m=50, n=50
Weights = 50x50
Bias = 50

num_params
= 2500+50
= 2550



m=50, n=1
Weights = 1x50
Bias = 1

num_params
= 50+1 = 51

m=1, n=50
Shared Weights = 50x1

ens1:
W r1=1x1
W s1=50x1
bias1=50

ens2:
W r2=1x1
W s2=50x1
bias2=50

ens3:
W r3=1x1
W s3=50x1
bias3=50

ens4:
W r4=1x1
W s4=50x1
bias4=50

Weights =50+(50+1) +(50+1)
+(50+1) +(50+1) = 254
Bias = 50+50+50+50=200
num_params=254+200=454

m=50, n=50
Shared Weights = 50x50

ens1:
W r1=50x1
W s1=50x1
bias1=50

ens2:
W r2=50x1
W s2=50x1
bias2=50

ens3:
W r3=50x1
W s3=50x1
bias3=50

ens4:
W r4=50x1
W s4=50x1
bias4=50

Weights =2500+(50+50) +(50+50)
+(50+50) +(50+50) = 2900
Bias = 50+50+50+50=200
num_params=2900+200=3100

m=50, n=1
Shared Weights = 1x50

ens1:
W r1=50x1
W s1=1x1
bias1=1

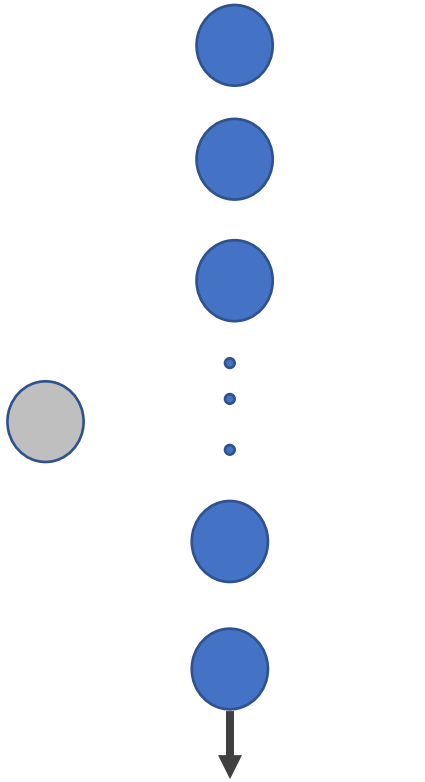
ens2:
W r2=50x1
W s2=1x1
bias2=1

ens3:
W r3=50x1
W s3=1x1
bias3=1

ens4:
W r4=50x1
W s4=1x1
bias4=1

Weights =50+(50+1) +(50+1)
+(50+1) +(50+1) = 254
Bias = 1+1+1+1=4
num_params=254+4=258

Rank-1 BNN for each of the rank1 vector we have a distribution that is represented by 2 value mu and sigma



m=1, n=50
Shared Weights = 50x1

ens1:
W r1=1x1
W s1=50x1
bias1=50

ens2:
W r2=1x1
W s2=50x1
bias2=50

ens3:
W r3=1x1
W s3=50x1
bias3=50

ens4:
W r4=1x1
W s4=50x1
bias4=50

Weights =50+(50+1)*2 +(50+1)*2
+(50+1)*2 +(50+1)*2 = 458

Bias = 50+50+50+50=200

num_params=458+200=658

m=50, n=50
Shared Weights = 50x50

ens1:
W r1=50x1
W s1=50x1
bias1=50

ens2:
W r2=50x1
W s2=50x1
bias2=50

ens3:
W r3=50x1
W s3=50x1
bias3=50

ens4:
W r4=50x1
W s4=50x1
bias4=50

Weights =2500+(50+50)*2 +(50+50)*2
+(50+50)*2 +(50+50)*2 = 3300

Bias = 50+50+50+50=200

num_params=3300+200=3500

m=50, n=1
Shared Weights = 1x50

ens1:
W r1=50x1
W s1=1x1
bias1=1

ens2:
W r2=50x1
W s2=1x1
bias2=1

ens3:
W r3=50x1
W s3=1x1
bias3=1

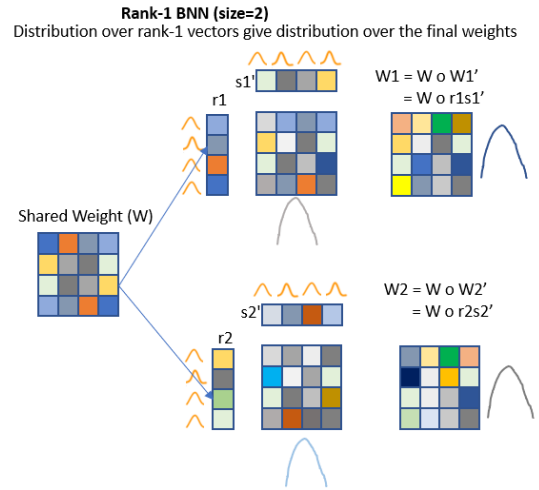
ens4:
W r4=50x1
W s4=1x1
bias4=1

Weights =50+(50+1)*2 +(50+1)*2
+(50+1)*2 +(50+1)*2 = 458

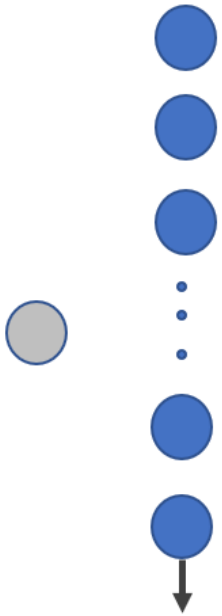
Bias = 1+1+1+1=4

num_params=458+4=462

ri and si share same dimension as input and output – m & n respectively



Rank-1



m=1, n=50
Shared Weights = 50x1

ens1:
W r1=1x1
W s1=50x1
bias1=50

ens2:
W r2=1x1
W s2=50x1
bias2=50

ens3:
W r3=1x1
W s3=50x1
bias3=50

ens4:
W r4=1x1
W s4=50x1
bias4=50

Weights =50+(50+1)*2 +(50+1)*2
+(50+1)*2 +(50+1)*2 = 458

Bias = 50+50+50+50=200

num_params=458+200=658

Batch Ensemble

m=1, n=50
Shared Weights = 50x1

ens1:
W r1=1x1
W s1=50x1
bias1=50

ens2:
W r2=1x1
W s2=50x1
bias2=50

ens3:
W r3=1x1
W s3=50x1
bias3=50

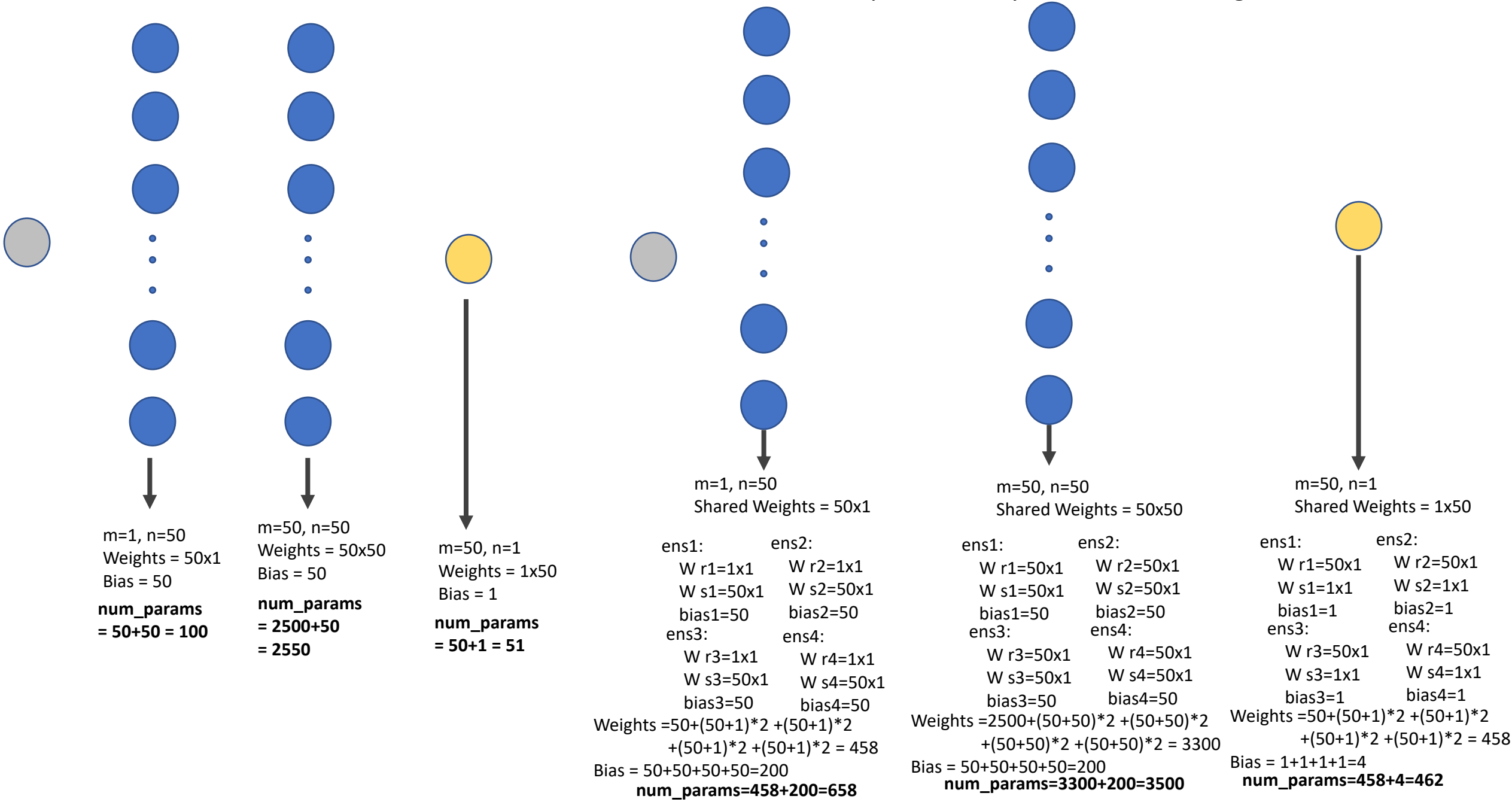
ens4:
W r4=1x1
W s4=50x1
bias4=50

Weights =50+(50+1) +(50+1)
+(50+1) +(50+1) = 254

Bias = 50+50+50+50=200

num_params=254+200=454

Rank-1 BNN for each of the rank1 vector we have a distribution that is represented by 2 value mu and sigma



m=1, n=50
Weights = 50x1
Bias = 50

num_params
= 50+50 = 100

m=50, n=50
Weights = 50x50
Bias = 50

num_params
= 2500+50
= 2550

m=50, n=1
Weights = 1x50
Bias = 1

num_params
= 50+1 = 51

m=1, n=50
Shared Weights = 50x1

ens1:
W r1=1x1
W s1=50x1
bias1=50

ens2:
W r2=1x1
W s2=50x1
bias2=50

ens3:
W r3=1x1
W s3=50x1
bias3=50

ens4:
W r4=1x1
W s4=50x1
bias4=50

Weights =50+(50+1)*2 +(50+1)*2
+(50+1)*2 +(50+1)*2 = 458
Bias = 50+50+50+50=200
num_params=458+200=658

m=50, n=50
Shared Weights = 50x50

ens1:
W r1=50x1
W s1=50x1
bias1=50

ens2:
W r2=50x1
W s2=50x1
bias2=50

ens3:
W r3=50x1
W s3=50x1
bias3=50

ens4:
W r4=50x1
W s4=50x1
bias4=50

Weights =2500+(50+50)*2 +(50+50)*2
+(50+50)*2 +(50+50)*2 = 3300
Bias = 50+50+50+50=200
num_params=3300+200=3500

m=50, n=1
Shared Weights = 1x50

ens1:
W r1=50x1
W s1=1x1
bias1=1

ens2:
W r2=50x1
W s2=1x1
bias2=1

ens3:
W r3=50x1
W s3=1x1
bias3=1

ens4:
W r4=50x1
W s4=1x1
bias4=1

Weights =50+(50+1)*2 +(50+1)*2
+(50+1)*2 +(50+1)*2 = 458
Bias = 1+1+1+1=4
num_params=458+4=462

Thank You