
SAE: Sequential Anchored Ensembles

Arnaud Delaunoy
University of Liège
a.delaunoy@uliege.be

Gilles Louppe
University of Liège
g.louppe@uliege.be

Abstract

Computing the Bayesian posterior of a neural network is a challenging task due to the high-dimensionality of the parameter space. Anchored ensembles approximate the posterior by training an ensemble of neural networks on anchored losses designed for the optima to follow the Bayesian posterior. Training an ensemble, however, becomes computationally expensive as its number of members grows since the full training procedure is repeated for each member. In this note, we present Sequential Anchored Ensembles (SAE), a lightweight alternative to anchored ensembles. Instead of training each member of the ensemble from scratch, the members are trained sequentially on losses sampled with high auto-correlation, hence enabling fast convergence of the neural networks and efficient approximation of the Bayesian posterior. SAE outperform anchored ensembles, for a given computational budget, on some benchmarks while showing comparable performance on the others and achieved 2nd and 3rd place in the light and extended tracks of the NeurIPS 2021 Approximate Inference in Bayesian Deep Learning competition.

1 Introduction

Accurate uncertainty quantification has become of high importance for machine learning tasks where incorrect prediction could have severe consequences. Bayesian deep learning is a popular framework for capturing those uncertainties. It consists in estimating the Bayesian posterior

$$p(\boldsymbol{\theta} | \mathbf{D}) = \frac{p(\mathbf{D} | \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{D})}, \quad (1)$$

where \mathbf{D} is a dataset and $\boldsymbol{\theta}$ the parameters of the neural network. Predictions are then made through marginalization over the parameters

$$p(\mathbf{y} | \mathbf{x}, \mathbf{D}) = \int p(\mathbf{y} | \mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta} | \mathbf{D})d\boldsymbol{\theta}, \quad (2)$$

where \mathbf{y} are the outputs and \mathbf{x} are the inputs. In practice, inferring the Bayesian posterior is challenging due to the high-dimensionality of the parameter space; scalable techniques to approximate this posterior are hence needed.

One of such techniques is called anchored ensembles [Pearce et al., 2020]. It builds an ensemble of neural networks with a randomized objective function designed for the optima to approximately follow the Bayesian posterior $p(\boldsymbol{\theta} | \mathbf{D})$. Under the assumption of a normal prior $p(\boldsymbol{\theta}) = \mathcal{N}(\mu_{\text{prior}}, \Sigma_{\text{prior}})$ and likelihood $p(\mathbf{D} | \boldsymbol{\theta})$, the optima $\boldsymbol{\theta}^*$ obtained by minimizing the anchored loss $-(\log p(\mathbf{D} | \boldsymbol{\theta}) + \log p_{\text{anc}}(\boldsymbol{\theta}))$, where $p_{\text{anc}} = \mathcal{N}(\boldsymbol{\theta}_{\text{anc}}, \Sigma_{\text{prior}})$, approximately follow the Bayesian posterior $p(\boldsymbol{\theta} | \mathbf{D})$ if $\boldsymbol{\theta}_{\text{anc}} \sim p(\boldsymbol{\theta})$.

Algorithm 1 Anchored Ensembling (AE)

```
for  $i$  in  $1, \dots, N$  do
   $\theta_{\text{anc},i} \sim p(\theta)$  ▷ Sample anchor
   $\theta_{\text{init},i} \leftarrow \text{INIT}()$  ▷ Initialize NN
   $\theta_i^* \leftarrow \text{TRAIN}(\theta_{\text{anc},i}, \theta_{\text{init},i})$  ▷  $\theta_i^* \leftarrow \arg \min_{\theta} -(\log p(\mathbf{D} | \theta) + \log p_{\text{anc}}(\theta))$ 
```

2 Sequential anchored ensembles

In this work, we propose Sequential Anchored Ensembles (SAE) which stem from the observation that if $\theta_{\text{anc},i-1}$ and $\theta_{\text{anc},i}$ are close so should be θ_{i-1}^* and θ_i^* . Therefore, training all the members of the ensemble from scratch is inefficient. We exploit this to reduce the computational cost of the training procedure by training the elements of the ensemble sequentially starting from the previous solution θ_{i-1}^* . This is illustrated in Figure 1 and summarized in Algorithm 2.

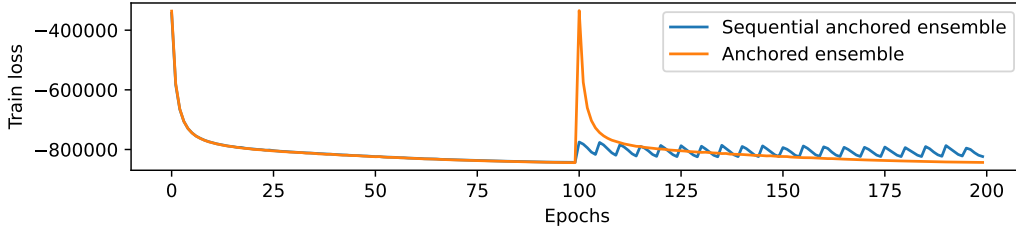


Figure 1: While anchored ensembles start the training from scratch at a high loss, sequential anchored ensembles start from the previously obtained solution and hence at a lower loss allowing to build an ensemble of 21 members in the time anchored ensembles built an ensemble of 2 members. Each peak corresponds to the start of a new training procedure, the fast convergence of sequential anchored ensembles allows to train much more ensemble members than anchored ensembles in the same time.

Let us consider the construction of an ensemble composed of N members $\theta_1^*, \dots, \theta_N^*$, with θ_i^* constructed based on the anchor $\theta_{\text{anc},i}$. A first anchor is sampled from the prior and the corresponding neural network is trained in a classical way. The next anchors are sampled such that two consecutive anchors are close to each other and the corresponding neural networks are trained starting from the previous optimum. As consecutive anchors are close, so are the solutions, and the training procedure is then expected to converge much faster. To sample consecutive anchors that are close to each other but eventually span the prior, we rely on an MCMC procedure. As the parameters are assumed independent under the prior, we run a separate chain for each parameter of the neural network so that some transitions are accepted by the Metropolis-Hastings algorithm at each step. The parameters of the anchors being independently normally distributed, the distribution is easy to navigate as opposed to an MCMC procedure directly performed on the posterior. To benefit from parallelization and decrease ensemble correlation, the algorithm can be run multiple times with different initializations.

Algorithm 2 Sequential Anchored Ensembling (SAE)

```
 $\theta_{\text{anc},1} \sim p(\theta)$  ▷ Sample first anchor  
 $\theta_{\text{init},1} \leftarrow \text{init}()$  ▷ Initialize NN  
 $d_1 \leftarrow \text{RANDCHOICE}(\text{SIZE}(\theta_{\text{anc},1}), \{-1, 1\})$  ▷ Direction  
 $\theta_1^* \leftarrow \text{TRAIN}(\theta_{\text{anc},1}, \theta_{\text{init},1})$  ▷ Long training  
for  $i$  in  $2, \dots, N$  do ▷  $i^{\text{th}}$  step of the SAE algorithm  
   $\theta_{\text{anc},i}, d_i \leftarrow \text{MH\_UPDATE}((\theta_{\text{anc},i-1}, d_{i-1}))$  ▷ Alg. 3  
   $\theta_{\text{init},i} \leftarrow \theta_{i-1}^*$  ▷ Start from previous optimum  
   $\theta_i^* \leftarrow \text{TRAIN}(\theta_{\text{anc},i}, \theta_{\text{init},i})$  ▷ Short training
```

Algorithm 3 Guided walk Metropolis-Hastings

```

function MH_UPDATE( $\theta_{\text{anc}}, \mathbf{d}$ )
   $\theta'_{\text{anc}} \leftarrow \text{EMPTY}(\text{SIZE}(\theta_{\text{anc}})), \mathbf{d}' \leftarrow \text{EMPTY}(\text{SIZE}(\mathbf{d}))$ 
  for  $(\theta_{\text{anc},j}, d_j)$  in  $(\theta_{\text{anc}}, \mathbf{d})$  do                                ▷  $j^{\text{th}}$  anchor's parameter
     $y \leftarrow \theta_{\text{anc},j} + d_j|z|, z \sim \mathcal{N}(0, \sigma_{\text{step}})$                                 ▷ Transition
     $\alpha \leftarrow \min\left(\frac{p(y)}{p(\theta_{\text{anc},j})}, 1\right)$                                 ▷ Acceptance probability
     $u \sim \mathcal{U}(0, 1)$ 
    if  $u < \alpha$  then                                                    ▷ Accept transition
       $\theta'_{\text{anc},j} \leftarrow y, d'_j \leftarrow d_j$                                 ▷ Keep same direction
    else                                                                    ▷ Reject transition
       $\theta'_{\text{anc},j} \leftarrow \theta_{\text{anc},j}, d'_j \leftarrow -d_j$                                 ▷ Invert direction
  return  $\theta'_{\text{anc}}, \mathbf{d}'$ 

```

The algorithm can be performed with any MCMC procedure, however, to be efficient, the MCMC procedure should eventually span the whole prior space so that the optima span the whole posterior while making small transitions for training to remain fast. We have found that a guided walk Metropolis-Hastings procedure with Gaussian transitions [Gustafson, 1998] performs well. This procedure is illustrated in Algorithm 3. The difference between a classical Metropolis-Hastings algorithm and its guided walk version is that the latter always performs transitions in the same direction until a rejection occurs. On rejection, the transition direction is inverted until the next rejection. As we know the prior to be normally distributed, rejection cannot occur until the point of maximal density is passed. The anchors will then consistently evolve in the same direction hence allowing to efficiently span the prior with short steps.

3 Experiments

To assess the efficiency of SAE, we compare how close the approximated predictive density $1/N \sum_{i=1}^N p(\mathbf{y} | \mathbf{x}, \theta_i^*)$ is to the true density $\int p(\mathbf{y} | \mathbf{x}, \theta)p(\theta | \mathbf{D})d\theta$ both for anchored ensembles and sequential anchored ensembles. For a given computational budget, the sequential anchored ensembles will be composed of more members than the anchored ensembles but each member will be trained for a shorter time and built sequentially. The true posterior used for comparison has been computed by a Hamiltonian Monte-Carlo procedure [Izmailov et al., 2021] in the context of the NeurIPS 2021 Approximate Inference in Bayesian Deep Learning competition [Wilson et al., 2021].

		Cifar10 Resnet-20		Cifar10(-C) Alexnet		IMDB		DermaMNIST		UCI-Gap
		Ag.	TV	Ag.	TV	Ag.	TV	Ag.	TV	W_2
1000 epochs	AE	0.849	0.201	0.726	0.262	0.892	0.109	0.877	0.104	0.148
	SAE	0.856	0.176	0.772	0.212	0.887	0.110	0.880	0.098	0.159
10,000 epochs	AE	0.862	0.199	0.746	0.236	0.926	0.086	0.897	0.089	0.137
	SAE	0.903	0.133	0.787	0.200	0.916	0.099	0.893	0.086	0.143

Table 1: Comparison of the performance of anchored ensembles and sequential anchored ensembles. While the computational budget is split uniformly between ensemble members for anchored ensembles, sequential anchored ensembles perform many short training procedures allowing to construct more ensemble members for the same computational budget. The reported values are the median performance over at least 20 runs. Cifar10(-C) corresponds to a dataset composed of 20% of uncorrupted samples and 80% of samples with a corruption level of 4.

Results for computational budgets of 1000 and 10,000 epochs are shown in Table 1. Additional results, experimental details and metrics definitions can be found in Appendix A. We observe that for a given computational budget, SAE performs better than AE on Cifar10 and Cifar10-C, slightly worse on UCI-Gap and shows similar performance on the other datasets. It shows that the sequential procedure is able to navigate the posterior and sometimes approximates the posterior density more

efficiently than traditional anchored ensembles. Interestingly, we observe that when both methods yield a similar agreement, SAE usually yields a lower total variation. SAE hence tends to provide better-calibrated posteriors even when classification performance is similar. The reason why it performs better on some datasets than others is still unclear. However, we hypothesize that some posteriors are easier to navigate leading to high performance using SAE while others are harder and benefit more from different initializations of the weights.

4 Related work

Building an ensemble in a sequential manner has already been proven successful in the context of non-anchored deep ensembles. Snapshot ensembles [Huang et al., 2017] and Fast Geometric Ensembling [Garipov et al., 2018] build a deep ensemble by recording weights during the training procedure. They make use of a cyclical learning rate schedule to increase diversity. Stochastic Weights Averaging [Izmailov et al., 2018] directly averages the weights in place of the predictions leading to lower memory requirements and prediction complexity. Sequential anchored ensembling follows similar ideas but uses anchors to create diversity in the ensemble following the Bayesian posterior. Improvements of those methods include low-precision training [Yang et al., 2019] and Simplicial Pointwise Random Optimization [Benton et al., 2021] which build simplexes of low loss. Such ideas could possibly be adapted for the sequential anchored ensembles setting, improving its performance.

Acknowledgments and Disclosure of Funding

The authors would like to thank Tim Pearce for his insightful comments and feedback. Arnaud Delaunoy would like to thank the National Fund for Scientific Research (F.R.S.-FNRS) for his scholarship. Gilles Louppe is recipient of the ULiège - NRB Chair on Big Data and is thankful for the support of the NRB.

References

- Gregory Benton, Wesley Maddox, Sanae Lotfi, and Andrew Gordon Gordon Wilson. Loss surface simplexes for mode connecting volumes and fast ensembling. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 769–779. PMLR, 18–24 Jul 2021.
- Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8803–8812, 2018.
- Paul Gustafson. A guided walk metropolis algorithm. *Statistics and computing*, 8(4):357–364, 1998.
- Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *International Conference on Learning Representations*, 2017.
- Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.
- Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Wilson. What are bayesian neural network posteriors really like? *arXiv preprint arXiv:2104.14421*, 2021.
- Tim Pearce, Felix Leibfried, and Alexandra Brintrup. Uncertainty in neural networks: Approximately bayesian ensembling. In *International conference on artificial intelligence and statistics*, pages 234–244. PMLR, 2020.
- Andrew Gordon Wilson, Pavel Izmailov, Matthew D Hoffman, Yarin Gal, Yingzhen Li, Melanie F Pradier, Sharad Vikram, Andrew Foong, Sanae Lotfi, and Sebastian Farquhar. Evaluating approximate inference in bayesian deep learning. 2021.

Guandao Yang, Tianyi Zhang, Polina Kirichenko, Junwen Bai, Andrew Gordon Wilson, and Chris De Sa. Swalp: Stochastic weight averaging in low precision training. In *International Conference on Machine Learning*, pages 7015–7024. PMLR, 2019.

A Additional experiments

In this section, we provide results on smaller budgets and all the experimental details, results are shown in Table 2. Details about the architectures can be found in Table 3. The allocation of the computational budget for the UCI-Gap dataset can be found in Table 5 while the others can be found in Table 4. The metrics used to compare the approximate posterior to the true one are, for classification, the agreement

$$\text{agreement}(p, \hat{p}) = \frac{1}{n} \sum_{i=1}^N I[\arg \max_j \hat{p}(y_j | x_i) = \arg \max_j p(y_j | x_i)] \quad (3)$$

and the total variation

$$\text{TV}(p, \hat{p}) = \frac{1}{n} \sum_{i=1}^n \frac{1}{2} \sum_j |p(y_j | x_i) - \hat{p}(y_j | x_i)|, \quad (4)$$

where p is the true predictive density and \hat{p} is the approximate one. For regression we use a point-wise Wasserstein-2 distance based on samples from the predictive distribution

$$W_2(p, \hat{p}) = \inf_I \sqrt{\sum_{i \in I, j} |p_i - \hat{p}_j|^2}. \quad (5)$$

Agreement	budget (epochs)	200	500	1000	10,000
Cifar10 Resnet-20	AE	0.804 ^{+0.010} _{-0.020}	0.833 ^{+0.009} _{-0.015}	0.849 ^{+0.006} _{-0.011}	0.862 ^{+0.005} _{-0.005}
	SAE	0.782 ^{+0.012} _{-0.025}	0.830 ^{+0.010} _{-0.017}	0.856 ^{+0.006} _{-0.010}	0.903 ^{+0.004} _{-0.002}
Cifar10-C Alexnet	AE	0.660 ^{+0.007} _{-0.010}	0.706 ^{+0.005} _{-0.007}	0.726 ^{+0.004} _{-0.003}	0.746 ^{+0.003} _{-0.007}
	SAE	0.738 ^{+0.006} _{-0.014}	0.762 ^{+0.003} _{-0.005}	0.772 ^{+0.005} _{-0.004}	0.787 ^{+0.002} _{-0.003}
IMDB	AE	0.829 ^{+0.008} _{-0.009}	0.865 ^{+0.007} _{-0.008}	0.892 ^{+0.004} _{-0.003}	0.926 ^{+0.001} _{-0.002}
	SAE	0.831 ^{+0.003} _{-0.005}	0.868 ^{+0.004} _{-0.006}	0.887 ^{+0.004} _{-0.007}	0.916 ^{+0.002} _{-0.001}
DermaMNIST	AE	0.826 ^{+0.014} _{-0.021}	0.865 ^{+0.011} _{-0.008}	0.877 ^{+0.007} _{-0.006}	0.897 ^{+0.005} _{-0.009}
	SAE	0.836 ^{+0.009} _{-0.027}	0.869 ^{+0.011} _{-0.008}	0.880 ^{+0.006} _{-0.012}	0.893 ^{+0.008} _{-0.014}

Total variation	budget	200	500	1000	10,000
Cifar10 Resnet-20	AE	0.225 ^{+0.017} _{-0.014}	0.211 ^{+0.015} _{-0.007}	0.201 ^{+0.012} _{-0.011}	0.199 ^{+0.003} _{-0.002}
	SAE	0.223 ^{+0.022} _{-0.012}	0.194 ^{+0.020} _{-0.012}	0.176 ^{+0.011} _{-0.007}	0.133 ^{+0.002} _{-0.003}
Cifar10-C Alexnet	AE	0.335 ^{+0.005} _{-0.008}	0.283 ^{+0.005} _{-0.007}	0.262 ^{+0.006} _{-0.003}	0.236 ^{+0.006} _{-0.002}
	SAE	0.245 ^{+0.014} _{-0.006}	0.221 ^{+0.004} _{-0.003}	0.212 ^{+0.004} _{-0.004}	0.200 ^{+0.004} _{-0.003}
IMDB	AE	0.180 ^{+0.007} _{-0.006}	0.131 ^{+0.005} _{-0.003}	0.109 ^{+0.003} _{-0.005}	0.086 ^{+0.001} _{-0.001}
	SAE	0.160 ^{+0.003} _{-0.003}	0.124 ^{+0.004} _{-0.003}	0.110 ^{+0.004} _{-0.003}	0.099 ^{+0.001} _{-0.002}
DermaMNIST	AE	0.153 ^{+0.012} _{-0.014}	0.117 ^{+0.006} _{-0.012}	0.104 ^{+0.007} _{-0.006}	0.089 ^{+0.005} _{-0.009}
	SAE	0.143 ^{+0.023} _{-0.014}	0.111 ^{+0.006} _{-0.010}	0.098 ^{+0.011} _{-0.007}	0.086 ^{+0.004} _{-0.001}

W_2	budget	200	500	1000	10,000
UCI-Gap	AE	0.174 ^{+0.357} _{-0.054}	0.152 ^{+0.083} _{-0.042}	0.148 ^{+0.142} _{-0.032}	0.137 ^{+0.039} _{-0.021}
	SAE	0.196 ^{+0.481} _{-0.072}	0.169 ^{+0.149} _{-0.059}	0.159 ^{+0.162} _{-0.068}	0.143 ^{+0.096} _{-0.041}

Table 2: Comparison of the performance of anchored ensembles and sequential anchored ensembles for various computational budgets. While the computational budget is split uniformly between ensemble members for anchored ensembles, sequential anchored ensembles perform many short training procedures allowing to construct more ensemble members for the same computational budget. The reported values are the median, minimal and maximal performance over 100 runs for the UCI-Gap dataset and over 20 runs for the other datasets.

Cifar10 Resnet-20	20-layers CNN
Cifar10 Alexnet	5-layers CNN
IMDB	1 Conv layer + LSTM cell
DermaMNIST	2-layers CNN
UCI-Gap	2-layers MLP

Table 3: Architectures used with each dataset.

Budget (epochs)	200	500	1000	10,000
AE	2 members of 100 epochs 2 members	5×100 5 members	10×100 10 members	100×100 100 members
SAE	1 chain of 100 first epochs 50 sequential trainings of 2 epochs 51 members	$2(100 + 75 \times 2)$ 152 members	$3(100 + 116 \times 2)$ 351 members	$10(100 + 450 \times 2)$ 4510 members

Table 4: Allocation of the computational budget for the Cifar10, Cifar10(-C), IMDB and DermaM-NIST datasets.

Budget (epochs)	200	500	1000	10,000
AE	2 members of 100 epochs 2 members	5×100 5 members	10×100 10 members	100×100 100 members
SAE	1 chain of 100 first epochs 10 sequential trainings of 10 epochs 11 members	$2(100 + 15 \times 10)$ 32 members	$3(100 + 23 \times 10)$ 72 members	$10(100 + 90 \times 10)$ 910 members

Table 5: Allocation of the computational budget for the UCI-Gap dataset.