

SAE: Single Architecture Ensemble Neural Networks

Martin Ferianc¹
martin.ferianc.19@ucl.ac.uk

Hongxiang Fan²
h.fan17@imperial.ac.uk

Miguel Rodrigues¹
m.rodrigues@ucl.ac.uk

¹ Department of Electronic and Electrical Engineering,
University College London,
London,
United Kingdom

² Department of Computing,
Imperial College London,
London,
United Kingdom

Abstract

Ensembles of separate neural networks (NNs) have shown superior accuracy and confidence calibration over single NN across tasks. To improve the hardware efficiency of ensembles of separate NNs, recent methods create ensembles within a single network via adding early exits or considering multi input multi output approaches. However, it is unclear which of these methods is the most effective for a given task, needing a manual and separate search through each method. Our novel Single Architecture Ensemble (SAE) framework enables an automatic and joint search through the early exit and multi input multi output configurations and their previously unobserved in-between combinations. SAE consists of two parts: a scalable search space that generalises the previous methods and their in-between configurations, and an optimisation objective that allows learning the optimal configuration for a given task. Our image classification and regression experiments show that with SAE we can automatically find diverse configurations that fit the task, achieving competitive accuracy or confidence calibration to baselines while reducing the compute operations or parameter count by up to $1.5\sim 3.7\times$.

1 Introduction

Ensemble of independently trained NNs achieves superior accuracy and confidence calibration over a single NN across tasks [19, 27]. In a naive ensemble, each NN is trained independently, which enables the ensemble members to learn independent features and make diverse predictions [14]. At evaluation time, each NN is queried with the same input, and the overall prediction is obtained by aggregating the individual predictions, leading to improved algorithmic performance. The computational cost of training and querying an NN ensemble increases linearly with N , the number of NNs in the ensemble as $\mathcal{O}(N)$. Therefore, as the number of NNs in the ensemble increases, the computational cost of training and querying the ensemble becomes prohibitive.

Recent advances have introduced more hardware-efficient solutions, encapsulating the ensemble within a singular network architecture. They do so by either adding early exits (EEs) to intermediate layers of the NN [1, 20, 25, 29], feeding the NN multiple inputs and simultaneously expecting multiple outputs (MIMO) [14], or a blend of both: multi input massive multi output (MIMMO) [8]. These techniques mimic the naive ensemble of NNs by simultaneously training multiple predictors in one training round and NN architecture and collecting their predictions in a single forward pass for the final predictions. These optimisations reduce the computational cost of training and inference to $\mathcal{O}(1)$, making the resultant ensembles hardware-efficient.

However, the current methodologies are fragmented and limited to their specific configurations e.g. only considering early exits - EE, multiple inputs and outputs - MIMO, or an extreme combination of both in MIMMO, and it is unclear which method is the most effective for a given task. In this work, we hypothesise that it is necessary to unify these methods and introduce a search space that encompasses them and their previously unexplored in-between configurations to find the optimal configuration for a given task. Therefore we introduce the novel Single Architecture Ensemble (SAE) framework consisting of a scalable search space and an optimisation objective that allows learning the optimal hardware-efficient ensemble configuration for a given task. We summarise our contributions as follows:

- 1.) A novel search space and problem formulation, encompassing the early exit, multiple inputs and outputs methods and their new in-between configurations.
- 2.) An optimisation objective that allows learning an optimal configuration for a given task, facilitating automatic and efficient exploration of the proposed search space.
- 3.) An empirical evaluation on image classification and regression tasks demonstrating that there is no one-size-fits-all hardware-efficient ensemble configuration out of the previous methods, confirming our hypothesis. However, with our novel SAE framework, we can find diverse configurations, achieving competitive accuracy or confidence calibration to baselines while reducing the compute operations or parameter count by up to $1.5\sim 3.7\times$.

2 Related Work

Modern NNs are becoming increasingly complex, with billions of parameters and hundreds of layers [14]. Nevertheless, it can be shown that most of these parameters are unused, indicating that the network is overparameterised [11]. Hardware-efficient NN ensembles utilise this overparameterisation to train a single NN that simulates the naive ensemble by providing the practitioner with diverse predictions in a single pass, thus improving the algorithmic performance while maintaining the hardware efficiency of a single NN. These approaches include early exit (EE) networks [1, 20, 25, 29], multi input multi output (MIMO) networks [14], and multi input massive multi output (MIMMO) networks [8].

We illustrate these approaches by considering a network consisting of $D \geq 1$ layers, and processing $N \geq 1$ inputs and outputs. A standard single exit (SE) NN processes a single input and output, $N = 1$, and gives a single prediction at the maximum depth of the network, D .

Early Exit (EE) EE networks [1, 20, 25, 29] introduce minimally-parametrised auxiliary exits at all or some intermediate layers within D layers of the NN. These exits predict the output at their respective depth, and the final prediction for a particular input is obtained by aggregating the predictions of all exits. The EE network leverages models of varying capacity by ensembling the exits' predictions at different network depths. For EE, the free parameter is the number and position of the exits within the network's depth D .

Multi Input Multi Output (MIMO) MIMO networks [14, 22, 26, 30, 33, 35, 36] process multiple inputs in a single architecture while being competitive with rank-1 Bayesian NNs and Batch Ensemble approaches [7, 38, 39]. A MIMO network processes N inputs and outputs simultaneously, such that N inputs are concatenated before the first layer into a single feature tensor. At the end of the network at depth D , the network gives N predictions, a separate prediction for each input. During training all the N input and target pairs are distinct, and at test time, the predictions for N repeated identical inputs are aggregated to give a single prediction. MIMO learns a representation that encodes the information from all inputs simultaneously, utilising the network’s capacity more efficiently. For MIMO, the free parameter is the number of inputs N that the network should process simultaneously.

Multi Input Massive Multi Output (MIMMO) MIMMO networks [8] combine MIMO and EE by adding exactly $D - 1$ reshaping connectors to the network, feeding in the intermediate layers’ outputs to the prediction head at the final layer. MIMMO processes N inputs and outputs for each exit, resulting in $N \times D$ predictions per forward pass through the network. MIMMO trains to match the predictions between all exits and targets, and at test time, the predictions are averaged across all exits for repeated inputs. While MIMMO utilises all the exits’ predictions within the network’s depth D , the number of inputs N is a free parameter.

The current methodologies in hardware-efficient NN ensembles, such as EE, MIMO, and MIMMO, are specialised and constrained to particular settings. For instance, EE focuses on the arrangement and number of exits within a network’s depth, MIMO on processing multiple inputs simultaneously, and MIMMO combines features of both, optimising the usage of all exits and the number of inputs. Our work argues that there is not a universal, optimal configuration or method that suits all problem settings, but it depends on the type and size of the NN and the complexity of the dataset. Therefore, it is necessary to define a scalable search space that encompasses these methods and introduces their novel in-between configurations, and an optimisation objective that allows its efficient exploration. In the Supplementary Material, we provide an even more detailed comparison of the generalised methods, and baseline methods compared to in the experiments.

3 Single Architecture Ensemble (SAE)

To address the limitations of the current methodologies, we introduce the SAE framework. It consists of a scalable search space and an optimisation objective that allows learning the optimal hardware-efficient ensemble configuration for a given task.

3.1 Search Space of SAE

We first consider a naive generalisation of the EE, MIMO, and MIMMO methods and their novel in-between configurations by considering a network with D layers and N inputs and outputs per exit. If we consider that a network processes N inputs and can have up to D exits, we can represent the search space of the generalised methods and their in-between configurations as $(2^D - 1)^N$, where $(2^D - 1)$ exits are turned on or off for each input in N . Manual exploration of this search space is impractical, and it is necessary to introduce a simplification that allows automatic and efficient exploration of the search space.

We simplify the search space by considering the number of inputs N and the *maximum* number of exits $1 \leq K \leq D$ that can be active for each input in N . The new efficient search

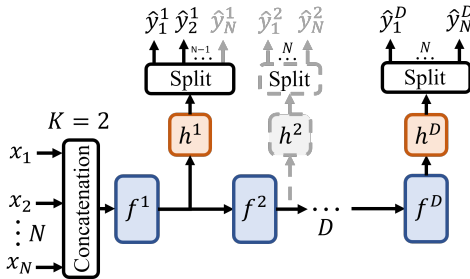


Figure 1: The Single Architecture Ensemble (SAE). The filled rectangles stand for learnable layers $\{f^j(\cdot)\}_{j=1}^D$ and prediction heads $\{h^j(\cdot)\}_{j=1}^D$, while the empty rectangle represents a non-parametric operation. The D is the network depth, N is the number of separate inputs in the ensemble, and K is the maximum number of active exits during evaluation for each input. The $\{x_i\}_{i=1}^N$ are the N inputs and $\{\hat{y}_i^j\}_{i,j=1}^{N,D}$ represent the predictions from the D exits for N inputs. The arrows represent the flow of information. The dashed and greyed boxes and arrows represent exits that were active during training but inactive during evaluation because of top K exits identified during training.

space is reduced to $N \times D \ll (2^D - 1)^N$ where a practitioner needs to try $N \times D$ configurations via setting N and K . By setting N and K , we recover the generalised methods as:

- if $N = 1, K = 1$, this is a Standard Single exit (SE) NN;
- if $N = 1, K \geq 2$, this is an EE NN;
- if $N \geq 2, K = 1$, this is a MIMO NN;
- if $N \geq 2, K = D$, this is a MIMMO NN;
- if $N \geq 2, 1 < K < D$, this is a previously unexplored In-Between (I/B) configuration.

The SAE search space can be illustrated as a network architecture in Figure 1. The network processes N inputs via a widened input layer, after which the refined features of the D layers are processed by exits at different depths. During training, all the exits are active, and the network produces $N \times D$ predictions. However, during evaluation, the network produces $N \times K$ predictions by selecting the top K exits for each input. Next, we present the optimisation objective that allows learning which K exits to use for each input in N and the network weights at the same time.

3.2 Optimisation Objective of SAE

The optimisation is built on jointly learning the network weights and the depth distribution of the K most suitable exits for each input in N via variational inference [4]. Through defining the preferences for the exits for each input as learnable variables, we avoid having to manually search through the search space of $(2^D - 1)^N$ configurations, and instead the depth preference is learned during training, similarly to learning operations in differentiable neural architecture search (DNAS) [23]. In contrast to DNAS, the final configuration does not need to be discretised and retrained. To fully explore the search space, the user needs to run the training only $N \times D$ times, where up to D exits can be active for each input, making the traversal of the search space more efficient.

We define the SAE’s optimisation objective, initially considering N independent ensemble inputs. Denote the dataset $\mathcal{D} = \{x_i, y_i\}_{i=1}^{|D|}$, where x_i is the input and y_i is the ground truth

for input i and $|\mathcal{D}|$ is the size of the dataset \mathcal{D} . We assume initially that our model processes N inputs $\{x_i\}_{i=1}^N$ and produces N matching outputs $\{\hat{y}_i\}_{i=1}^N$. During training, we assume that the N inputs $\{x_i\}_{i=1}^N$ and targets $\{y_i\}_{i=1}^N$ are sampled independently from the dataset \mathcal{D} in \mathcal{N} , hence assuming independence between $p(\{y_i\}_{i=1}^N | \{x_i\}_{i=1}^N, \mathbf{w}) = \prod_{i=1}^N p(y_i | \{x_z\}_{z=1}^N, \mathbf{w})$ [14], where \mathbf{w} are the learnable weights. By independence, we mean that the pairs (x_i, y_i) are sampled independently from the other $N - 1$ pairs. There is no mixing of inputs and targets and y_i is the ground truth associated with x_i . We denote $\mathbf{x}_i = \{x_z\}_{z=1}^N$ as the input for i , containing x_i and $N - 1$ random sampled inputs from the dataset \mathcal{D} , matching the $\{y_i\}_{i=1}^N$.

With the independence assumption, we introduce the exits, which enable the NN to make a prediction for each input i at D different depths as $p(y_i | \mathbf{x}_i, d_i = j, \mathbf{w}) = p(y_i | \hat{y}_i^j)$; $\hat{y}_i^j = h^j(f^j(f^{j-1}(\dots f^1(\mathbf{x}_i))))$ where $1 \leq j \leq D$ and $f^j(\cdot)$ is the j -th layer and $h^j(\cdot)$ is the prediction head for the j -th layer. The \hat{y}_i^j is the prediction for input i at depth j and d_i is the latent variable for D exits for input i . The computation is visualised in Figure 1.

The evidence of the data for a single tuple is $p(y_i | \mathbf{x}_i, \mathbf{w}) = \sum_{j=1}^D p(y_i | \mathbf{x}_i, d_i = j, \mathbf{w}) p(d_i = j)$ for a categorical prior for the latent variable d_i , $p(d_i)$ where $p(d) = \prod_{i=1}^N p(d_i)$. We introduce a categorical variational distribution $q(d | \theta) = \prod_{i=1}^N q(d_i | \theta_i)$ with parameters $\theta = \{\theta_i^j\}_{i,j=1}^{N,D}$, where θ_i are the parameters of the weightings of all the exits for input i . Our aim is to learn θ to determine the K active exits for each input during evaluation together with \mathbf{w} .

Antorán et al. [1] demonstrated for $N = 1$ that by choosing the prior and the variational distribution to be categorical, it is possible to jointly optimise \mathbf{w} and θ and maximise the evidence lower bound of the data via stochastic gradient ascent. To derive the evidence lower bound for all N independent predictors, we minimise the Kulback-Leibler (KL) divergence [15] between the variational distribution $q(d | \theta)$ and the posterior distribution $p(d | \mathcal{D}, \mathbf{w}) = \prod_{i=1}^N p(d_i | \mathcal{D}, \mathbf{w})$ as:

$$\begin{aligned} KL(q(d | \theta) || p(d | \mathcal{D}, \mathbf{w})) &= \sum_{i=1}^N KL(q(d_i | \theta_i) || p(d_i | \mathcal{D}, \mathbf{w})) \\ &= \sum_{i=1}^N \mathbb{E}_{q(d_i | \theta_i)} \left[\log q(d_i | \theta_i) - \log \frac{p(Y_i | \mathbf{X}_i, d_i, \mathbf{w}) p(d_i)}{p(Y_i | \mathbf{X}_i, \mathbf{w})} \right] \\ &= \underbrace{\sum_{i=1}^N \mathbb{E}_{q(d_i | \theta_i)} [-\log p(Y_i | \mathbf{X}_i, d_i, \mathbf{w})]}_{\text{Data-fit in } -\mathcal{L}(\mathbf{w}, \theta)} + \underbrace{KL(q(d_i | \theta_i) || p(d_i))}_{\text{Regulariser in } -\mathcal{L}(\mathbf{w}, \theta)} + \underbrace{\sum_{i=1}^N \log p(Y_i | \mathbf{X}_i, \mathbf{w})}_{\text{Evidence}} \quad (1) \end{aligned}$$

The KL divergence in Equation 1 is decomposed into the expectation of the log-likelihood, or the data-fit term, of the data $Y_i = \{y_i^{(j)}\}_{j=1}^{|\mathcal{D}|}$, given input tuples $\mathbf{X}_i = \{\mathbf{x}_i^{(j)}\}_{j=1}^{|\mathcal{D}|}$ for dataset size $|\mathcal{D}|$, the exits d_i and the weights \mathbf{w} , the KL divergence between the variational distribution $q(d_i | \theta_i)$ and the prior distribution $p(d_i)$, or the regulariser term, and the evidence of the data. The KL divergence is non-negative, and the evidence of the data is maximised when the $\mathcal{L}(\mathbf{w}, \theta)$ is maximised as $\mathcal{L}(\mathbf{w}, \theta) \leq \sum_{i=1}^N \log p(Y_i | \mathbf{X}_i, \mathbf{w})$. The objective $\mathcal{L}(\mathbf{w}, \theta)$ can be further decomposed with respect to all the inputs N and the whole dataset \mathcal{D} as in Equation 2.

$$\begin{aligned} \mathcal{L}(\mathbf{w}, \theta) &= \sum_{i=1}^N \mathbb{E}_{q(d_i | \theta_i)} [\log p(Y_i | \mathbf{X}_i, d_i, \mathbf{w})] - \sum_{i,j=1}^{N,D} \theta_i^j \log \frac{\theta_i^j}{D-1} \\ &= \sum_{i,b=1}^{N,|\mathcal{D}|} \mathbb{E}_{q(d_i | \theta_i)} [\log p(y_i^{(b)} | \mathbf{x}_i^{(b)}, d_i, \mathbf{w})] - \sum_{i,j=1}^{N,D} \theta_i^j \log \frac{\theta_i^j}{D-1} \quad (2) \end{aligned}$$

The b is the index of the input tuple $\mathbf{x}_i^{(b)}$ and the target $y_i^{(b)}$ sampled from the dataset \mathcal{D} .

We assume a uniform categorical prior, meaning that the KL divergence between the variational distribution $q(d|\theta)$ and the prior distribution $p(d)$ can be computed in closed form. Equation 3 approximates the evidence lower bound, where B is the batch size.

$$\mathcal{L}(\mathbf{w}, \theta) \approx \frac{|\mathcal{D}|}{B} \sum_{i,b=1}^{N,B} \mathbb{E}_{q(d_i|\theta_i)} \left[\log p(y_i^{(b)} | \mathbf{x}_i^{(b)}, d_i, \mathbf{w}) \right] - \sum_{i,j=1}^{N,D} \theta_i^j \log \frac{\theta_i^j}{D-1} \quad (3)$$

We propose to sample the depth variable d_i from the variational distribution $q(d_i|\theta_i)$ using a top K sampling strategy [18] which enables us to optimise the θ towards the top K exits per input. We use one Monte Carlo sample per batch to approximate the expectation, the sum of the D log-likelihoods multiplied by the sampled probabilities of d_i . The θ are implemented as zero-initialised logits divided by a temperature T to which softmax was applied to normalise the logits as $\theta_i^j = \frac{\exp(l_i^j/T)}{\sum_{k=1}^D \exp(l_i^k/T)}$ and l_i^j is the j -th exit logit for input i .

We empirically observed that the training of the SAE might be compromised by overregularisation, which can be caused by the regulariser term in Equation 3 or by the size of the dataset $|\mathcal{D}|$. We propose to multiply the regulariser term by a factor $\alpha(t)$, where $0 \leq \alpha(t) \leq 1$ is a linear interpolation depending on the training step t , and replace the $\frac{|\mathcal{D}|}{B}$ with 1 in Equation 3. Furthermore, we propose to linearly interpolate between starting and ending values of the temperature $0 < T(t) \leq 1$ and the input repetition factor $0 \leq i(t) \leq 1$ during training, where the starting and ending values can be optimised. The $T(t)$ determines the sharpness of the probabilities over the auxiliary exits during sampling, which enables the framework to gradually focus on the K most important auxiliary exits for each input in the architecture. The $i(t)$ determines a portion of the batch B where the same input is repeated across all the N inputs, relaxing the independence assumption [14]. The linear interpolation of hyperparameters aims to enable the framework to iterate over multiple configurations during training, trading-off exploration and exploitation of the search space. We empirically observed that the linear hyperparameter schedules were not too disruptive to the training process, however, other strategies can be considered.

During the evaluation, the SAE framework produces $N \times K$ predictions $\{\hat{y}_i^j\}_{i,j=1}^{N,K}$, where K is the number of exits used for each input in N . It does so through only keeping the top K largest logits and setting the rest to negative infinity to give $\theta^* = \{\theta_i^{*j}\}_{i,j=1}^{N,K}$. If no inputs select an exit at some level j , it does not need to be implemented as shown in Figure 1 in grey. The input sample is repeated N times as $\mathbf{x}^* = \{\mathbf{x}_i\}_{i=1}^N$ and the predictions from the active exits are collected. The final prediction is obtained by averaging the predictions from the exits and their θ_i^{*j} as $\hat{y}^* = \frac{1}{N} \sum_{i,j=1}^{N,K} \hat{y}_i^j \theta_i^{*j}$ which approximates the marginal likelihood.

In summary, the training objective $\mathcal{L}(\mathbf{w}, \theta)$ allows us to learn the exit preferences θ for each input $i \in N$ and each exit $j \in D$ along the network’s weights \mathbf{w} via gradient ascent. A practitioner can set the $N \geq 1$ and $1 \leq K \leq D$ to explore the search space of the generalised methods and their novel in-between configurations by optimising the \mathbf{w} and θ via the $\mathcal{L}(\mathbf{w}, \theta)$. The resultant \mathbf{w} and θ^* define the optimal configuration for a given task, maximally exploiting the network’s capacity through the search space exploration. In the Supplementary Material, we provide an Algorithm summarising the training and evaluation as well as the implementation details for the input and early exits per architecture type.

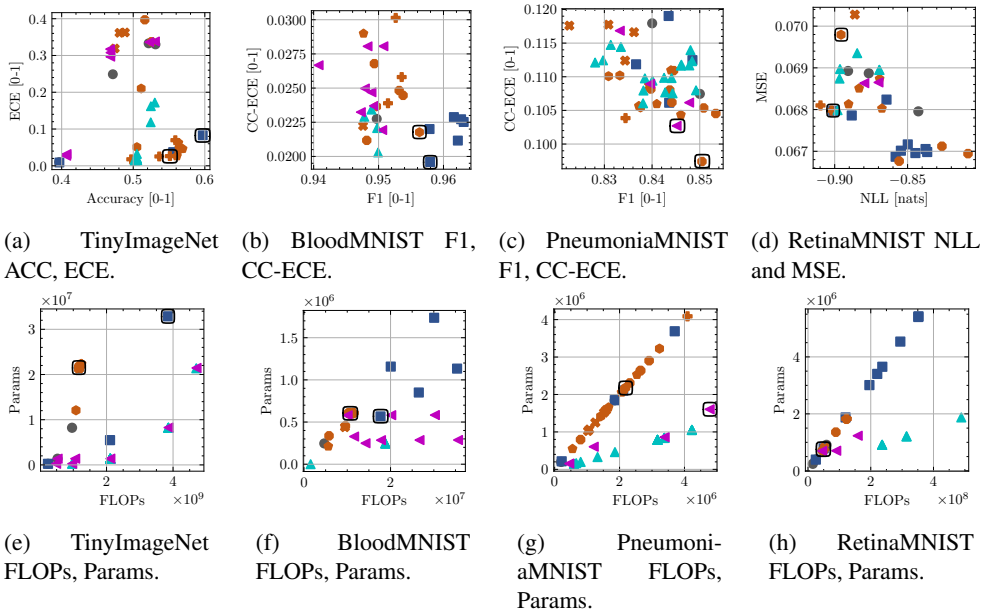


Figure 2: Comparison on ID test sets, with respect to **Standard NN** ●, **NN Ensemble** ■, **SAE: I/B: $N \geq 2, 2 \leq K < D$** +, **EE: $N = 1, K \geq 2$** ●, **MIMMO: $N \geq 2, K = D$** ◆, **MIMO: $N \geq 2, K = 1$** ◇, **SE NN: $N = 1, K = 1$** ✦, **MCD** ▲, **BE** ◀. The black outlines denote the configurations compared in the text.

4 Experiments

We perform experiments on four datasets: TinyImageNet [14], BloodMNIST, PneumoniaMNIST and RetinaMNIST, [10] for both classification and regression tasks. Our architecture backbones are ResNet [15] for TinyImageNet, ViT [9] for RetinaMNIST, VGG [16] for BloodMNIST, and a residual fully connected net with batch normalisation [17] and ReLU activations (FC) for PneumoniaMNIST. We compare approaches under SAE with Monte Carlo Dropout (MCD) [18], where we insert dropout [19] layers before each linear or convolutional layer. We also compare to Batch Ensemble (BE) [20], where the BE layers replace all linear and convolutional layers in the network. The algorithmic lower bound is a standard NN. The algorithmic upper bound is a naive ensemble of NNs. We employ multi-objective Bayesian optimisation (MOBO) to perform hyperparameter optimisation (HPO) for SAE and: $1 \leq N \leq 4, 1 \leq K \leq D$ the input repetition, alpha and temperature start and end values and the width of the network for all the evaluated metrics simultaneously on the validation dataset. For all the other methods, we enumerate networks of all widths, depths, or N for BE and NN ensembles. Exceptions are MCD, for which we set $N = 4$ to make it competitive and we fix the network width for TinyImageNet. We use MOBO to search for the MCD or SAE’s parameters. We minimise unweighted negative log-likelihood (NLL) loss for all tasks except for SAE. For classification, we measure F1 score [$\uparrow 0, 1$], Accuracy (ACC) [$\uparrow 0, 1$], Expected Calibration Error (ECE) [$\downarrow 0, 1$], Class Conditional ECE (CC-ECE) [$\downarrow 0, 1$] and the NLL [$\downarrow 0, \infty$] as evaluation metrics. For regression, we use Gaussian NLL [$\downarrow -\infty, \infty$] and mean squared error (MSE) [$\downarrow 0, \infty$] as evaluation metrics. From the hardware perspective, we measure the number of floating point operations (FLOPs) [$\downarrow 0, \infty$] and the number of pa-

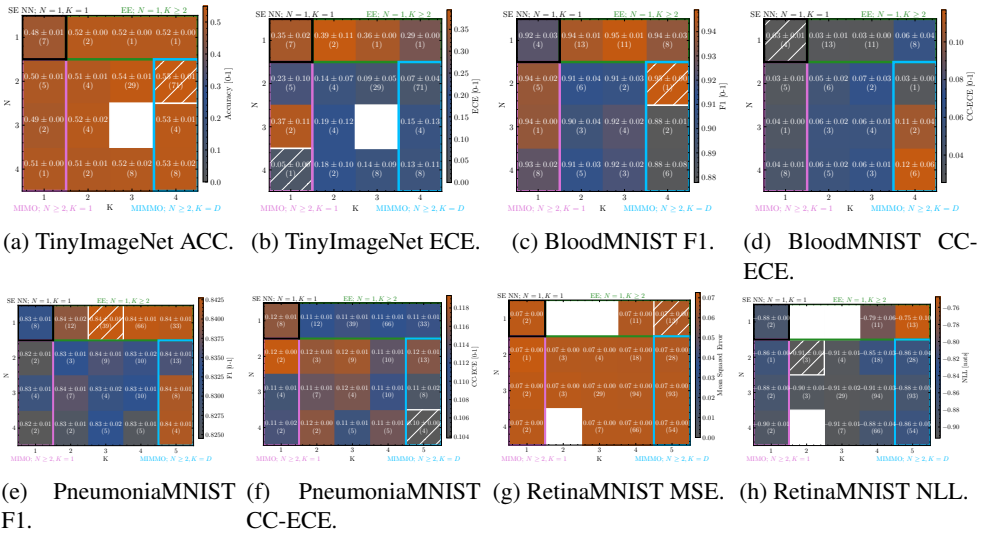


Figure 3: Varying N, K on across ID test sets. The upper number is the average performance over N, K combinations. The number in brackets is the number of sampled configurations by HPO. White box means no configurations sampled for that N, K . Pattern signals best average performance. The coloured outlines signal the special cases for the generalised methods.

rameters $\lfloor \downarrow 0, \infty$). We test on ID and OOD data created by applying augmentations to the test set, such as Gaussian noise, motion blur, and contrast changes [16]. We chose versatile architectures paired with datasets of varying complexity to demonstrate the SAE’s effectiveness across different tasks and datasets. The Supplementary Material details architectures, datasets, metrics, HPO runs, additional results on the correlation of hyperparameters, including $N, K, \alpha(t), i(t)$ and $T(t)$ and the performance, and OOD experiments.

4.1 Baseline Comparison

In Figure 2, we compare Pareto optimal configurations identified via HPO across all metrics and datasets. In the upper row, we show the results on ACC/F1/MSE and ECE/CC-ECE/NLL and in the lower row, we show the results on FLOPs and parameters. We split the methods in SAE into 5 categories based on the N and K : **I/B**: $N \geq 2, 2 \leq K < D$ \oplus , **EE**: $N = 1, K \geq 2$ \bullet , **MIMMO**: $N \geq 2, K = D$ \heartsuit , **MIMO**: $N \geq 2, K = 1$ \blacklozenge , **SE NN**: $N = 1, K = 1$ \blacklozenge . The black outlines around the markers in the Figures denote the configurations compared in the text. The results are averaged across 4 random seeds. The Supplementary Material contains numerical results for OOD datasets, the dataset’s best configurations and NLL comparison.

The overall results show that SAE’s Pareto points cluster around the best algorithmic and hardware performance, while providing trade-offs between the two. For TinyImageNet, in Figures 2a and 2e, it can be seen that the **ensemble** \blacksquare achieves the best accuracy. However, SAE can find an **I/B** \oplus configuration which is within 3% of accuracy, but 4% better ECE than the best **ensemble** \blacksquare with $3.2\times$ fewer FLOPs and $1.5\times$ fewer parameters. For BloodMNIST, in Figures 2b and 2f, it can be seen that the **ensemble** \blacksquare achieves the best algorithmic performance. At the same time, the SAE can find an **I/B** \oplus configuration which is within 1% of the F1 score or CC-ECE with a comparable number of parameters but approximately $1.5\times$ fewer FLOPs. For PneumoniaMNIST, in Figures 2c and 2g, it can be

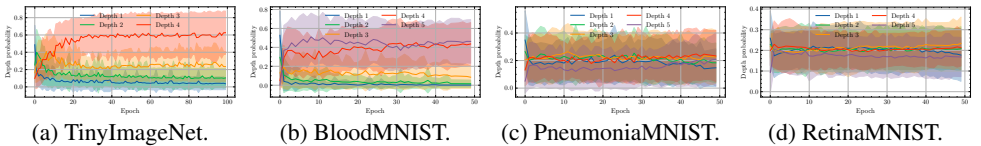






Figure 4: Depth preference during training when averaging all N and K for all datasets. The lines denote the mean trend, and the shaded regions denote the standard deviation across configurations.

seen that the **EE**  configuration is the best-performing method with approximately half FLOPs than the closest **BE** . For RetinaMNIST, in Figures 2d and 2h, it can be seen that in terms of NLL the **MIMO**  configuration is comparable to the **ensemble**  with marginally fewer FLOPs and parameters.

In summary, SAE can find a set of competitive or better configurations than the baselines across all datasets, tasks, architectures with various capacities, and evaluation metrics using fewer or comparable FLOPs and parameters. We hypothesise that this is due to the regularisation effect of the proposed objective, which prevents overfitting and instead focuses on fitting multiple sub-networks, each learning diverse representations, thus maximally utilising the capacity of the network. Combination of the representations, reflected via the diverse predictions ultimately leads to better algorithmic performance. In the Supplementary Material, we rank the methods based on their mean performance across all their Pareto optimal points, datasets and metrics and show that SAE achieves best or competitive ranks across the algorithmic metrics. SAE does not need custom operations as in BE, random number generators as in MCD, or more training rounds than a standard NN, MCD, or BE, it only needs to add compute-efficient early exits and an enlarged input layer capable of processing multiple inputs. The Supplementary Material provides a detailed hardware cost breakdown of SAE per each architecture type.

4.2 Ablations

In Figure 3, we investigate the influence of N and K on the algorithmic performance across ID test sets across all datasets. The Figures show the average performance across all N and K combinations across the individual HPO runs for each dataset.

As seen across the Figures, the best combinations of N and K can vary for the same dataset but different metrics, requiring a detailed search based on the specific use case. Figures 3a and 3b show that for TinyImageNet, $N = 2, 4$ and $K = 4, 1$ are the best configurations for accuracy and ECE, respectively. Figures 3c and 3d show that for BloodMNIST, $N = 2, 1$ and $K = 4, 1$ are the best configurations for F1, CC-ECE, respectively. Figures 3e and 3f show that for PneumoniaMNIST, $N = 1, 4$ and $K = 3, 5$ are the best configurations for F1, CC-ECE, respectively. Lastly, Figures 3g and 3h show that for RetinaMNIST, $N = 1, 2$ and $K = 5, 2$ are the best configurations for MSE and NLL, respectively.

An interesting observation is that $N > 1$ leads to better algorithmic performance, which differs from [8, 24] where $N > 1$ consistently leads to worse performance. On one hand, we observed that for easier problems, for example, PneumoniaMNIST the model’s capacity is large in comparison to the complexity of the problem, hence the network was able to learn representations for multiple predictors in the same architecture resulting in $N > 1$ and $K > 1$ configurations being beneficial. On the other hand, for harder problems, for example,

TinyImageNet, the model’s capacity was not sufficient to learn the representations for too many predictors, hence the performance peaks at a certain N and K combination.

Furthermore, we look at the learnt depth preference during training across all N and K for all datasets in Figure 4. As discussed in the previous paragraph, if the model’s capacity is large in comparison to the complexity of the problem, the depth preference is more uniform across the depths, as seen in Figures 4b, 4c, and 4d for BloodMNIST, PneumoniaMNIST, and RetinaMNIST, respectively. However, if the model’s capacity is not sufficient to learn the representations for too many predictors, the depth preference is strongly biased towards deeper depths, as seen in Figure 4a for TinyImageNet. These observations further validate our initial claim that there is no single best multi input multi output or early exit configuration across all datasets, tasks, and architectures, necessitating the need for a thorough and automatic search depending on the use case.

In terms of the backbone architectures and their impact on performance, we compare BloodMNIST and RetinaMNIST which have a similar complexity but use different architectures: VGG and ViT, respectively. We notice that for both problems the algorithmic performance improvements are similar across the datasets. However as discussed in the previous paragraphs, the optimal N and K , along with other hyperparameters, as well as the depth preference differ between the two datasets. Despite the different backbone architectures, the SAE can find competitive configurations for both datasets and architectures. The Supplementary Material contains more results on changing N , K or and depth preference.

5 Discussion

In this work we proposed the Single Architecture Ensemble (SAE). This unified framework melds the strengths of diverse hardware-efficient ensemble methods, outperforming traditional baselines in confidence calibration, accuracy and versatility across varied datasets, tasks, and architectures. Our findings underscore the necessity of SAE due to the lack of a universally superior method across all datasets, tasks, and architectures. Our method demonstrated that different network depths were used in different architectures and problem complexities, necessitating the need to learn the appropriate depth usage. Furthermore, we showed the different requirements for different architectures and the usefulness of adaptive hyperparameter scheduling.

There are multiple limitations of our study. In order to gain full benefits from SAE, the early exits, which introduce branching in the architecture, need to be implemented efficiently. An inefficient implementation can lead to a significant increase in the real-world hardware cost both during training and evaluation. Furthermore, we assumed an independence assumption between the inputs, which might not hold in practice. A theoretical investigation is needed to understand more complex assumptions and their impact on the performance and the trade-offs between the network’s representation capacity and quality of the learned representations. Lastly, we focused on image classification or regression tasks commonly solved in the community, which could fit our computational budget.

Therefore, in future work, we plan to investigate the theoretical properties of the SAE, such as the impact of the independence assumption on the performance and the trade-offs between the network’s representation capacity and the quality of the learned representations. Furthermore, to minimise the number of sub-optimal configurations, we plan to investigate the automatic learning of N and K , standing for the number of inputs and exits, respectively on tasks beyond image classification and regression.

Acknowledgements

Martin Ferianc was sponsored through a scholarship from the Institute of Communications and Connected Systems at UCL. Martin wants to thank syne-tune and team Matthias Seeger for their work in maintaining and improving the repository and advice regarding HPO.

References

- [1] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. *Advances in neural information processing systems*, 32, 2019.
- [2] Javier Antorán, James Allingham, and José Miguel Hernández-Lobato. Depth uncertainty in neural networks. *Advances in neural information processing systems*, 33: 10620–10634, 2020.
- [3] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.
- [4] Imre Csiszár. I-divergence geometry of probability distributions and minimization problems. *The annals of probability*, pages 146–158, 1975.
- [5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [7] Michael W Dusenberry, Ghassen Jerfel, Yeming Wen, Yi-an Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable Bayesian neural nets with rank-1 factors. In *ICML*, 2020.
- [8] Martin Ferianc and Miguel Rodrigues. Mimmo: Multi-input massive multi-output neural network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4563–4568, 2023.
- [9] Martin Ferianc and Miguel Rodrigues. Yamle: Yet another machine learning environment. *arXiv preprint arXiv:2402.06268*, 2024.
- [10] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.
- [11] Yarín Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.
- [12] Ellen R Girden. *ANOVA: Repeated measures*. Number 84. sage, 1992.

- [13] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.
- [14] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M Dai, and Dustin Tran. Training independent subnetworks for robust prediction. *arXiv preprint arXiv:2010.06610*, 2020.
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [16] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.
- [17] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.
- [18] Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *International Conference on Machine Learning*, pages 3499–3508. PMLR, 2019.
- [19] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.
- [20] Stefanos Laskaridis, Alexandros Kouris, and Nicholas D Lane. Adaptive inference through early-exit networks: Design, challenges and directions. In *Proceedings of the 5th International Workshop on Embedded and Mobile Deep Learning*, pages 1–6, 2021.
- [21] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.
- [22] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. *Advances in neural information processing systems*, 31, 2018.
- [23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.
- [24] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.
- [25] Yoshitomo Matsubara, Marco Levorato, and Francesco Restuccia. Split computing and early exiting for deep learning applications: Survey and research challenges. *ACM Computing Surveys*, 55(5):1–30, 2022.
- [26] Vishvak Murahari, Carlos E Jimenez, Runzhe Yang, and Karthik Narasimhan. Data-mux: Data multiplexing for neural networks. *arXiv preprint arXiv:2202.09318*, 2022.

- [27] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model’s uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019.
- [28] Biswajit Paria, Kirthevasan Kandasamy, and Barnabás Póczos. A flexible framework for multi-objective bayesian optimization using random scalarizations. In *Uncertainty in Artificial Intelligence*, pages 766–776. PMLR, 2020.
- [29] Lorena Qendro, Alexander Campbell, Pietro Lio, and Cecilia Mascolo. Early exit ensembles for uncertainty quantification. In *Machine Learning for Health*, pages 181–195. PMLR, 2021.
- [30] Alexandre Ramé, Rémy Sun, and Matthieu Cord. Mixmo: Mixing multiple inputs for multiple outputs via deep subnetworks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 823–833, 2021.
- [31] David Salinas, Matthias Seeger, Aaron Klein, Valerio Perrone, Martin Wistuba, and Cedric Archambeau. Syne tune: A library for large scale hyperparameter tuning and reproducible research. In *International Conference on Automated Machine Learning, AutoML 2022*, 2022. URL <https://proceedings.mlr.press/v188/salinas22a.html>.
- [32] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [33] Masoumeh Soflaei, Hongyu Guo, Ali Al-Bashabsheh, Yongyi Mao, and Richong Zhang. Aggregated learning: A vector-quantization approach to learning neural network classifiers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5810–5817, 2020.
- [34] 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.
- [35] Rémy Sun, Clément Masson, Nicolas Thome, and Matthieu Cord. Adapting multi-input multi-output schemes to vision transformers. In *CVPR 2022 workshop on Efficient Deep Learning for Computer Vision (ECV)*, 2022.
- [36] Rémy Sun, Alexandre Ramé, Clément Masson, Nicolas Thome, and Matthieu Cord. Towards efficient feature sharing in mimo architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2697–2701, 2022.
- [37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [38] Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020.

- [39] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. In *Neural Information Processing Systems*, 2020.
- [40] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1):41, 2023.
- [41] Jerrold H Zar. Spearman rank correlation. *Encyclopedia of biostatistics*, 7, 2005.
- [42] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A Survey of Large Language Models. *arXiv preprint arXiv:2303.18223*, 2023.