On Joint Regularization and Calibration in Deep Ensembles

Anonymous authors
Paper under double-blind review

Abstract

Deep ensembles are a powerful tool in machine learning, improving both model performance and uncertainty calibration. While ensembles are typically formed by training and tuning models individually, evidence suggests that jointly tuning the ensemble can lead to better performance. This paper investigates the impact of jointly tuning weight decay, temperature scaling, and early stopping on both predictive performance and uncertainty quantification. Additionally, we propose a partially overlapping holdout strategy that relaxes the need for a common holdout set, thereby increasing ensemble diversity. Our results demonstrate that jointly tuning the ensemble matches or improves performance across all conditions, with significant variation in effect size. We highlight the trade-offs between individual and joint optimization in deep ensemble training, with the overlapping holdout strategy offering an attractive practical solution. We believe our findings provide valuable insights and guidance for practitioners looking to optimize deep ensemble models.

1 Introduction

Deep ensembles are a simple and practical method that combines multiple independently trained models to enhance predictive accuracy, improve robustness, and provide uncertainty estimates (Lakshminarayanan et al.). Their effectiveness relies on having diverse members that have uncorrelated errors, which reduces variance and minimizes the impact of individual model mistakes (Hansen & Salamon; Krogh & Sollich).

While individual models in an ensemble may differ in architecture, training set, and other factors, a common practice is to train ensembles using the same model architecture, with the only differences being the initializations and the order in which the training examples are presented. This also offers a simple and effective method for selecting regularization hyperparameters such as weight decay and dropout: These settings can be optimized for a single model, typically through grid search, and then used to train the ensemble members independently. Similarly, if post-hoc calibration or early stopping is used, it is often applied to each ensemble member independently. This approach simplifies the tuning process, but while an ensemble of well-regularized and well-calibrated models will generally perform well, it may not be the optimal strategy. We refer to the possible mismatch between an ensemble of optimally tuned models and the optimally tuned ensemble as the *ensemble optimality gap*.

Previous work has shown that allowing individual models within an ensemble to overfit to a certain extent can lead to improvements in both prediction accuracy (Sollich & Krogh) and calibration (Wu & Gales). In practice, however, this is often disregarded because tuning the complete ensemble by holdout or cross-validation can be a considerable computational expense or does not seamlessly fit into existing workflows. Hyperparameter tuning (such as grid search) requires training an entire ensemble for each parameter combination, scaling the computational cost with the ensemble size. However, methods like early stopping can be validated during parallel training with minimal added cost, whereas post hoc techniques like temperature scaling can be evaluated on the ensemble without additional expense.

In this paper, we systematically explore the ensemble optimality gap across three key aspects of deep ensemble training and calibration: weight decay tuning, temperature scaling, and early stopping. Our objective is to assess the magnitude of this effect in common settings and demonstrate how it can be mitigated by optimizing for ensemble performance. In particular, we examine how the optimality gap influences model accuracy, uncertainty calibration, and predictive likelihood, as well as investigate its impact on ensemble diversity.

To enhance ensemble diversity, a well-known approach is to train the ensemble using a k-fold cross-validation strategy, where each ensemble member is validated on separate holdout data. While this approach can improve the estimation of generalization error for a single model, it prevents direct validation of the full ensemble performance, as a common validation data must be held out for all ensemble members. This leads to a choice between increasing ensemble diversity and having the ability to tune the ensemble as a whole, both of which strategies have been demonstrated to lead to improved performance. We explore a strategy that balances these factors by using partially overlapping holdout sets across ensemble members.

Finally, when training standard deep ensembles is impractical, techniques like batch ensembles or multiple-input multiple-output (MIMO) ensembles offer viable alternatives. In these approaches, the ensemble is formed by *sub-models* with partially shared parameters within a single, larger model that is trained in one run. We demonstrate how our overlapping holdout strategy can be applied in the batch ensemble setting and compare its performance across different initialization strategies.

In summary, our work addresses the following aspects:

- We demonstrate several settings in which the *ensemble optimality gap* is significant, and show to which extent it can be mitigated by jointly tuning the ensemble.
- We propose a novel *overlapping holdout* validation strategy that sits between using a common shared holdout set and using independent holdouts as in k-fold cross-validation.
- We present a case study based on *batch ensembling* that demonstrates how an ensemble can be jointly trained and tuned in a single run with the overlapping holdout validation strategy.

We validate our study empirically using two well-established benchmark tasks: one in image classification and the other in graph classification. Our results demonstrate clear benefits from validating the ensemble jointly, especially for early stopping and temperature scaling, compared to validating individual models, while joint weight decay tuning shows more nuanced effects predominantly related to calibration. We assess the utility of the overlapping holdout strategy in different settings and also provide key insights regarding initialization choices for efficient batch ensembles. Collectively, these findings offer concrete guidance for practitioners on navigating the trade-offs between individual and joint optimization when training and calibrating deep ensembles.

2 Background

2.1 Ensemble Methods: Foundations and Diversity

Ensemble methods improve prediction and robustness by combining multiple models (Dietterich). The core principle relies on combining outputs from diverse members with uncorrelated errors, thus reducing variance and improving generalization (Hansen & Salamon; Krogh & Sollich). In this paper, we focus on deep ensembles (Lakshminarayanan et al.), a simple and effective technique for deep neural networks. For classification tasks, we consider the common approach where the ensemble prediction is the arithmetic mean of the softmax probabilities from individual members,

$$\bar{\boldsymbol{p}}(y|x) = \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{p}(y|x, \theta_m),$$

corresponding to a uniform mixture over the ensemble members (see e.g. Tassi et al. for a discussion of the pros and cons of this strategy).

While effective, standard deep ensembles incur substantial costs, as training, storing, and running M independent models leads roughly to an M-fold increase in computation time at training and inference. This scalability challenge has spurred the development of more efficient ensemble methods. These approaches often reduce the computational or parameter costs through techniques like parameter sharing (e.g., BatchEnsemble (Wen et al., b)), creating implicit ensembles (e.g., MIMO (Havasi et al.), Early Exits (Qendro et al.)), leveraging stochastic inference (e.g., MC Dropout (Gal & Ghahramani)), or developing efficient Bayesian approximations (e.g., Rank-1 BNNs (Dusenberry et al.)).

Implicit and Explicit Diversity. Standard Deep Ensembles typically use identical architectures trained independently. Diversity is achieved *implicitly*, primarily through different random weight initializations, which serve as the main source of functional diversity, although using distinct stochastic batches during training also contribute to a lesser extent (Fort et al.). Data resampling techniques such as Bagging (Breiman) can promote beneficial diversity for traditional models but are often detrimental for deep networks (Lee et al.; Lakshminarayanan et al.). This is largely because deep models are sensitive to training data size, and the reduction in data per bagged member significantly weakens the individual predictors. Furthermore, regularization techniques applied during training (such as weight decay or early stopping), while improving individual model generalization, may also implicitly constrain the diversity among ensemble members. Although this implicit diversity (influenced by initialization, data splits, stochastic batches, and regularization) is often sufficient, explicit diversity-enhancing techniques can also lead to improvements, e.g., by modifying training losses or adding regularization (see e.g., Liu & Yao; Pagliardini et al.; Jain et al.), but their necessity and benefit, especially for large models, is debated (Abe et al., b;a). In some cases, joint training methods can lead to *learner collusion* (Jeffares et al.), a phenomenon where the ensemble members increase their diversity in a way that does not improve generalization.

Quantifying Diversity. Quantifying the diversity among ensemble members provides key insights into their collective behavior and prediction characteristics. For probabilistic predictive models, a useful information-theoretic metric for ensemble diversity is the difference between the entropy of the average predictive distribution, $\bar{\boldsymbol{p}}_i$, and the average entropy of individual member predictions, $\boldsymbol{p}_i^m$. This is equivalent to the average KL divergence $D_{\mathrm{KL}}(\boldsymbol{p}_i^m || \bar{\boldsymbol{p}}_i)$ (see Appendix B for details):

$$D_i = H(\bar{\boldsymbol{p}}_i) - \frac{1}{M}\sum_{m=1}^{M} H(\boldsymbol{p}_i^m). \tag{1}$$

In the classification setting with $\bar{\boldsymbol{p}}_i$ defined as the geometric mean, this expression has a natural interpretation in the form of a bias, variance, diversity decomposition of the expected loss (Wood et al.); however, in this work, we use the arithmetic mean, as it is more commonly applied. For an overview of alternative diversity metrics, see, e.g., Kuncheva & Whitaker; Heidemann et al..

2.2 Calibration of Deep Learning Models and Ensembles

Beyond predictive accuracy, the reliability of a model's confidence estimates is crucial for dependable decision-making, particularly in risk-sensitive applications (Niculescu-Mizil & Caruana). A model is considered well-calibrated if its predicted probabilities accurately reflect the true likelihood of correctness (e.g., predictions made with 80% confidence are correct 80% of the time). While modern deep neural networks achieve high accuracy, they are often found to be poorly calibrated, typically exhibiting overconfidence in their predictions (Guo et al.).

Calibration is commonly evaluated using metrics such as the expected calibration error (ECE), which measures the discrepancy between confidence and accuracy across prediction bins (Naeini et al.), and the negative log-likelihood (NLL) of the true classes. NLL is a proper scoring rule, meaning it is uniquely minimized when predicted probabilities match the true underlying probabilities, thus rewarding both accuracy and calibration (Gneiting & Raftery).

Temperature Scaling. A simple yet effective post-hoc technique for improving calibration is temperature scaling (Guo et al.). It involves rescaling the model's output logits $\boldsymbol{z}(\boldsymbol{x})$ by a single positive scalar parameter, the temperature T, before applying the softmax function according to the formula

$$\boldsymbol{p}(\boldsymbol{x};T) = \text{softmax}\left(\frac{\boldsymbol{z}(\boldsymbol{x})}{T}\right).$$

A temperature $T > 1$ softens the probability distribution (increasing entropy, reducing confidence), while $T < 1$ sharpens it. The optimal T is typically found as the value that minimizes some calibration metric on a held-out validation dataset $\mathcal{D}_{\text{val}}$. Using the NLL as the metric, the optimal temperature is given by

$$\arg\min_{T>0} \sum_{(\boldsymbol{x}_j, y_j)\in\mathcal{D}_{\text{val}}} -\log \boldsymbol{p}(\boldsymbol{x}_j;T)_{y_j}$$

where $\boldsymbol{p}(\boldsymbol{x}_j;T)_{y_j}$ denotes the predicted probability for the true class y_j. Since T only rescales logits before the softmax, it does not change individual models' accuracies.

Individual vs. Joint calibration When applying temperature scaling to an ensemble of M models, two main strategies arise, differing primarily in how the temperature parameter(s) are optimized and applied. It is important to note that *any* strategy involving temperature scaling applied before averaging the outputs of the non-linear softmax function can potentially alter the final classification outcome (i.e., the $\arg\max$ of the averaged probabilities) compared to averaging unscaled probabilities. The two main implementation strategies are:

- **Individual Temperature Scaling:** A separate temperature T_m is optimized for each ensemble member m, typically using its own validation set $\mathcal{D}_{\text{val}}^{(m)}$. The final ensemble prediction is the average of these individually calibrated probability vectors, given by

$$\bar{\boldsymbol{p}}^{\text{individual}}(\boldsymbol{x}) = \frac{1}{M}\sum_{m=1}^{M} \text{softmax}\left(\frac{\boldsymbol{z}_m(\boldsymbol{x})}{T_m}\right).$$

  Here, the potential impact on the classification outcome is influenced by the use of different scaling factors T_m across members.

- **Joint Temperature Scaling:** A single, shared temperature T_{joint} is optimized for the entire ensemble using a suitable joint validation set $\mathcal{D}_{\text{val}}^{\text{joint}}$. This shared temperature T_{joint} is applied to the logits $\boldsymbol{z}_m(\boldsymbol{x})$ of each member before the softmax activation, and the resulting calibrated probabilities are then averaged according to the formula

$$\bar{\boldsymbol{p}}^{\text{joint}}(\boldsymbol{x}) = \frac{1}{M}\sum_{m=1}^{M} \text{softmax}\left(\frac{\boldsymbol{z}_m(\boldsymbol{x})}{T_{\text{joint}}}\right).$$

  Although the scaling T_{joint} is uniformly applied (preserving the $\arg\max$ of individual members), the ensemble class prediction can change as averaging happens after the probability distributions are tempered. The optimal T_{joint} is found by minimizing a calibration metric (such as NLL) of this final averaged prediction $\bar{\boldsymbol{p}}^{\text{joint}}(\boldsymbol{x})$ on a joint validation set $\mathcal{D}_{\text{val}}^{\text{joint}}$.

The relationship between individual member calibration and overall ensemble calibration is non-trivial. Importantly, Wu & Gales showed that ensembling individually calibrated models does not guarantee a well-calibrated ensemble and can lead to under-confidence, advocating instead for calibration strategies that consider the ensemble effect. Their work focused specifically on optimizing temperature scaling parameters by minimizing ECE, whereas optimization based on proper scoring rules like NLL represents an alternative calibration objective commonly used for model training and evaluation.

2.3 Hyperparameter Tuning for Ensembles

Selecting appropriate hyperparameters, such as regularization strengths (e.g., weight decay) or learning parameters, is critical for training performant deep learning models. For ensembles, this presents a fundamental choice regarding the optimization objective: should hyperparameters be tuned to optimize the performance of individual members, or the performance of the ensemble as a whole?

A common practice, largely due to simplicity and significantly lower computational cost, involves tuning hyperparameters for a single model (e.g., via grid search or random search (Bergstra & Bengio)) and then applying the selected configuration uniformly to all ensemble members during their independent training (Lakshminarayanan et al.). Directly tuning for the ensemble objective, by contrast, would necessitate training and evaluating the *entire* ensemble for *each* candidate hyperparameter setting, incurring a computational cost that typically scales with the ensemble size and is often prohibitively expensive.

Beyond searching for a single optimal setting to apply uniformly, alternative strategies exist that leverage the models generated during hyperparameter exploration or explicitly use hyperparameter diversity. For instance, Wenzel et al. proposed *hyper-deep ensembles*, a method that explicitly combines models resulting from different hyperparameter settings (found via random search and greedy selection) and different random weight initializations. This combination of diversity sources was shown to improve robustness and uncertainty quantification compared to ensembles relying solely on random initialization. Similarly, Jin & Wu construct ensembles from models saved during learning rate schedule tuning runs, arguing this efficiently reuses computational effort and enhances diversity, reporting strong performance. A limitation of directly using models from tuning runs, however, is that the validation data used for hyperparameter selection is not incorporated into the training data for the final models, differing from standard workflows where models are typically retrained on combined data after tuning.

While these alternative construction methods exist, the common practice remains to tune a single configuration for standard deep ensembles. This standard practice implicitly assumes that hyperparameters optimal for a single model are also (close to) optimal for the ensemble, which might not hold true in practice, an issue also noted by Gorishniy et al.. However, there appears to be limited research directly comparing individual versus ensemble-based hyperparameter tuning for standard deep ensembles.

2.4 Early Stopping Ensembles

Early stopping is another widely used regularization technique that prevents overfitting by monitoring performance on a validation set and terminating training when performance on this set ceases to improve (Prechelt). Alternatively, the stopping point can be guided by estimators of generalization error that do not require held-out data, such as those derived from bootstrap ensembles (Hansen et al.). When applied to ensembles, the main approaches are:

- **Individual Early Stopping:** Each ensemble member m is trained and stopped independently based on its own performance, monitored on a validation set $\mathcal{D}_{val}^{(m)}$. Training for member m halts when its validation metric fails to improve for a predefined patience period.

- **Joint Early Stopping:** The performance of the *entire ensemble* (e.g., NLL of the average prediction) is monitored on a joint validation set $\mathcal{D}_{\text{val}}^{\text{joint}}$. Training for *all* members stops simultaneously when the *ensemble's* performance has not improved for the specified patience.

While individual early stopping optimizes each member in isolation, early ensemble theory suggests that allowing individual members to overfit slightly may be beneficial for the final ensemble performance (Sollich & Krogh). Joint early stopping might naturally facilitate this by potentially allowing longer training compared to the point where individual models start overfitting on their respective validation sets. Despite its conceptual appeal, the comparative benefits of individual versus joint stopping strategies for modern deep ensembles appear less explored. The feasibility and specific mechanism for evaluating ensemble performance for joint early stopping depend critically on the chosen validation data strategy (Section 3.2). Importantly, unlike potentially costly joint hyperparameter grid searches, joint early stopping can often be implemented with minimal computational overhead compared to individual early stopping, especially when members are trained in parallel.

3 Methodology

This section provides an overview of the experimental setup used to evaluate the impact of individual versus joint optimization strategies for deep ensembles. Full details regarding hyperparameters, specific training configurations, and architectures are provided in Appendix A. Error bars and shaded regions presented in figures correspond to 1.96× standard error of the mean calculated across multiple random seeds.

3.1 Datasets and Base Models

To assess the generality of our findings on ensemble optimization, we conducted experiments across two distinct and commonly used benchmark domains, differing significantly in data modality, task complexity, data scale, and model size:

- **Image Classification** (CIFAR-10 / WRN-16-4): Our first domain uses the CIFAR-10 dataset (Krizhevsky), a widely-used 10-class image classification benchmark with 50,000 training images. We pair this with a Wide ResNet-16-4 (WRN-16-4) model (Zagoruyko & Komodakis), a common CNN architecture for this task containing approximately 2.7 million parameters. This combination represents a setting where a high-capacity model operates on a moderately sized dataset, suggesting a significantly overparameterized regime. This characteristic makes it particularly suitable for studying the interplay between ensemble methods and factors like regularization (e.g., weight decay, early stopping) and multi-class calibration.
- **Graph Classification** (NCI1 / GCN): As a contrasting setting with structured data, we used the NCI1 graph classification benchmark (Shervashidze et al.; Wale et al.), a binary classification task with significantly fewer data points (4,110 graphs total, 3,288 used for training), with a four-layer Graph Convolutional Network (GCN). This pairing allows us to test our hypotheses on a different data modality and an architecture with substantially fewer parameters (24,204). Despite the vastly different scale, regularization remains important for the GCN's generalization on this dataset, allowing us to examine ensemble optimization effects in a different modeling context. Furthermore, the limited data availability may emphasize the impact of data allocation in different validation holdout strategies.

3.2 Validation Data Strategies for Ensemble Evaluation

How validation data is assigned to ensemble members impacts training and evaluation, particularly when considering joint ensemble objectives versus individual member training needs. We define $\mathcal{D}'$ as the available data excluding the final test set, $\mathcal{D}_{\text{val}}^{(m)}$ as the validation set for member m, and $\mathcal{D}_{\text{train}}^{(m)}$ as its training set. We consider three primary strategies for an ensemble of size M:

- **Shared Holdout:** All members use the same split: $\mathcal{D}_{\text{train}}^{(m)} = \mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{val}}^{(m)} = \mathcal{D}_{\text{val}}$ for all m. $\mathcal{D}_{\text{val}}$ is not used by any member during training. This allows direct evaluation of the full ensemble on $\mathcal{D}_{\text{val}}$.
- **Disjoint Holdout:** Each ensemble member m uses its own unique validation set, $\mathcal{D}_{\text{val}}^{(m)} \subset \mathcal{D}'$, of a predefined size. These validation sets are mutually disjoint, and their combined size cannot exceed the total available data $|\mathcal{D}'|$. Since member m trains on all data except its on validation set (i.e., $\mathcal{D}_{\text{train}}^{(m)} = \mathcal{D}' \setminus \mathcal{D}_{\text{val}}^{(m)}$), the data points used to validate it are necessarily included in the training data for all other members, maximizing training data utilization across the ensemble. However, this structure prevents direct evaluation of the full ensemble.
- **Overlapping Holdout:** Each model m gets a unique validation set $\mathcal{D}_{\text{val}}^{(m)}$ composed of two distinct parts (*halves*). Each half is shared with one neighboring model in a cyclical manner. For instance, half of Model 2's validation data is shared with Model 1, the other half with Model 3. This allows pairwise joint evaluation on the shared halves. (Formally, using M data portions S_k that partition

$\mathcal{D}'$, then $\mathcal{D}_{\text{val}}^{(m)} = S_m \cup S_{m+1 \pmod M}$ and $\mathcal{D}_{\text{train}}^{(m)} = \mathcal{D}' \setminus \mathcal{D}_{\text{val}}^{(m)}$). While this strategy does not permit estimating the validation performance of the entire ensemble on held-out data, it allows joint estimation of model pairs as an approximation.

Choosing among these strategies involves balancing trade-offs between data utilization (highest for disjoint holdout) and ability to perform joint ensemble evaluation (easiest with shared, possible pairwise with overlapping).

4 Experiments

4.1 Hyperparameter Tuning (Weight Decay)

Purpose This experiment aims to investigate how model performance is affected by different strategies for tuning the weight decay hyperparameter. Specifically, we compare two approaches: optimizing weight decay individually for a single model versus optimizing it to maximize the ensemble performance. This comparison aims to shed light on whether tuning for ensemble-level performance yields better generalization or uncertainty quantification.

Method Optimal weight decay was determined via grid search, minimizing validation NLL using a shared holdout set. We compared selecting the best value based on the average individual model NLL versus the NLL of the ensemble's average prediction. Models were trained using stochastic gradient descent (SGD) with momentum and cosine annealing (experimental parameters are detailed in Appendix Table 1).

Results Figure 1 shows the performance of ensembles of size 1–4 optimized for the joint NLL. Across both datasets we observe that there is a clear optimum for the NLL metric, and that the optimal weight decay parameter shifts marginally downward as the ensemble size increases. This is consistent with the idea that a small degree of overfitting of the individual models is beneficial for the ensemble. However, the effect size is small compared to the benefit of the ensemble itself. For CIFAR-10, the classification error follows a similar trend to the NLL; however, the optimal weight decay for calibration (measured by ECE) shifts significantly with ensemble size. This suggests that, if optimizing for ECE, it is crucial to tune weight decay based on ensemble performance in this case. For NCI1, the classification error and ECE reveal a clear trade-off between accuracy and calibration. Again, the largest effect of joint ensemble tuning is seen in ECE.

In Figure 2 we directly compare optimizing the weight decay for a single model versus for ensemble performance. For CIFAR-10 there is no benefit on classification error and NLL, but a strong benefit on ECE. For NCI1 there is a strong benefit on classification error but only little benefit on NLL and no benefit on ECE.

Conclusions The comparison suggests that tuning the weight decay specifically for the ensemble consistently leads to test performance that is either improved or on par with using the weight decay optimized for a single model across all metrics, though the improvements are generally modest. Whether the improvement shows as better classification error or improved calibration depends on the model and dataset, and the specific trade-off observed here is shaped by our decision to use NLL as the optimization criterion.

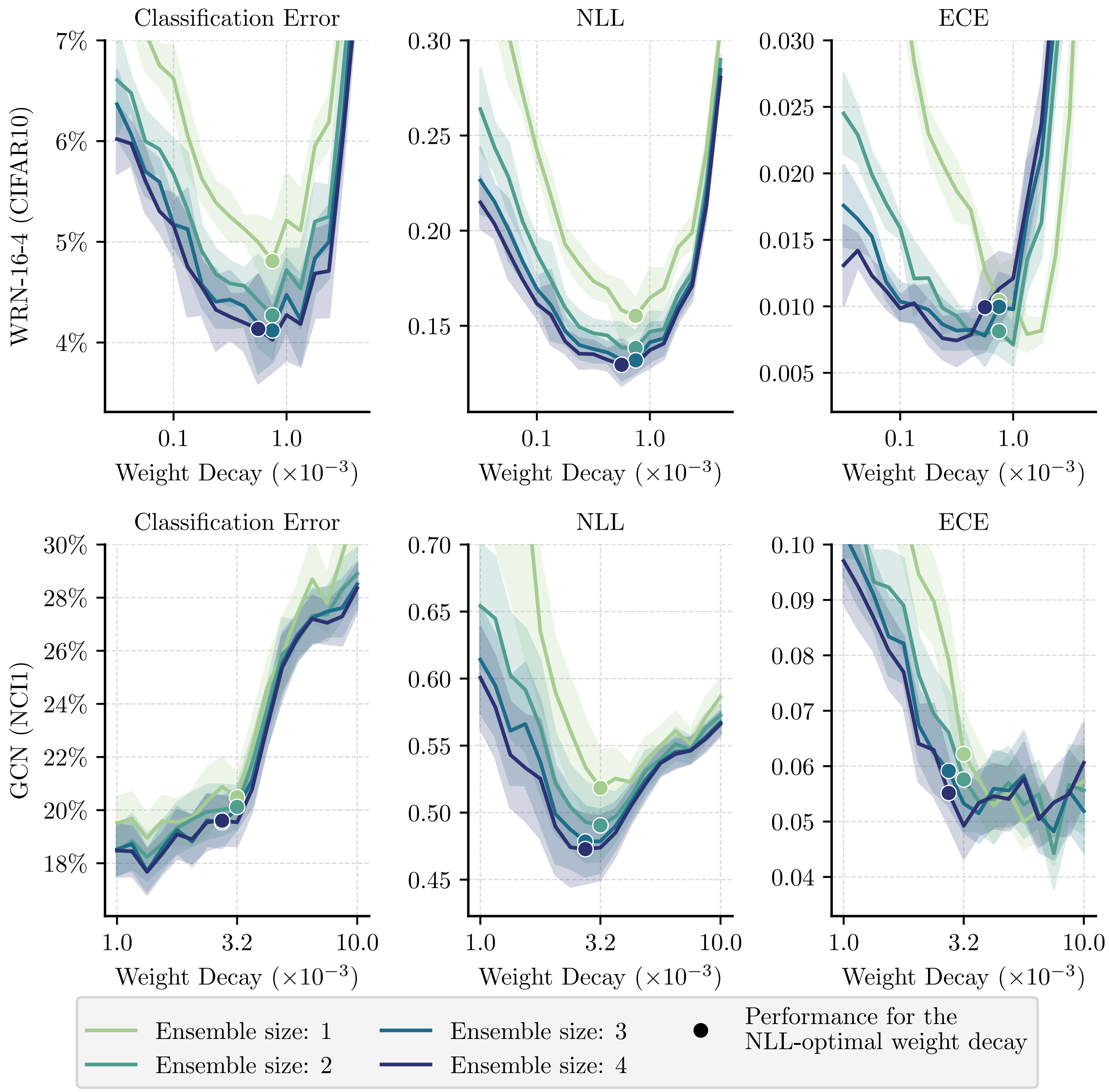

Figure 1: Validation performance across varying weight decay values for WRN-16-4 on CIFAR-10 and GCN on NCI1 for ensemble sizes 1 to 4. The optimal weight decay for each ensemble size is selected based on the lowest average NLL. (WRN: Wide ResNet; GCN: Graph Convolutional Network; NLL: negative log-likelihood; ECE: expected calibration error).

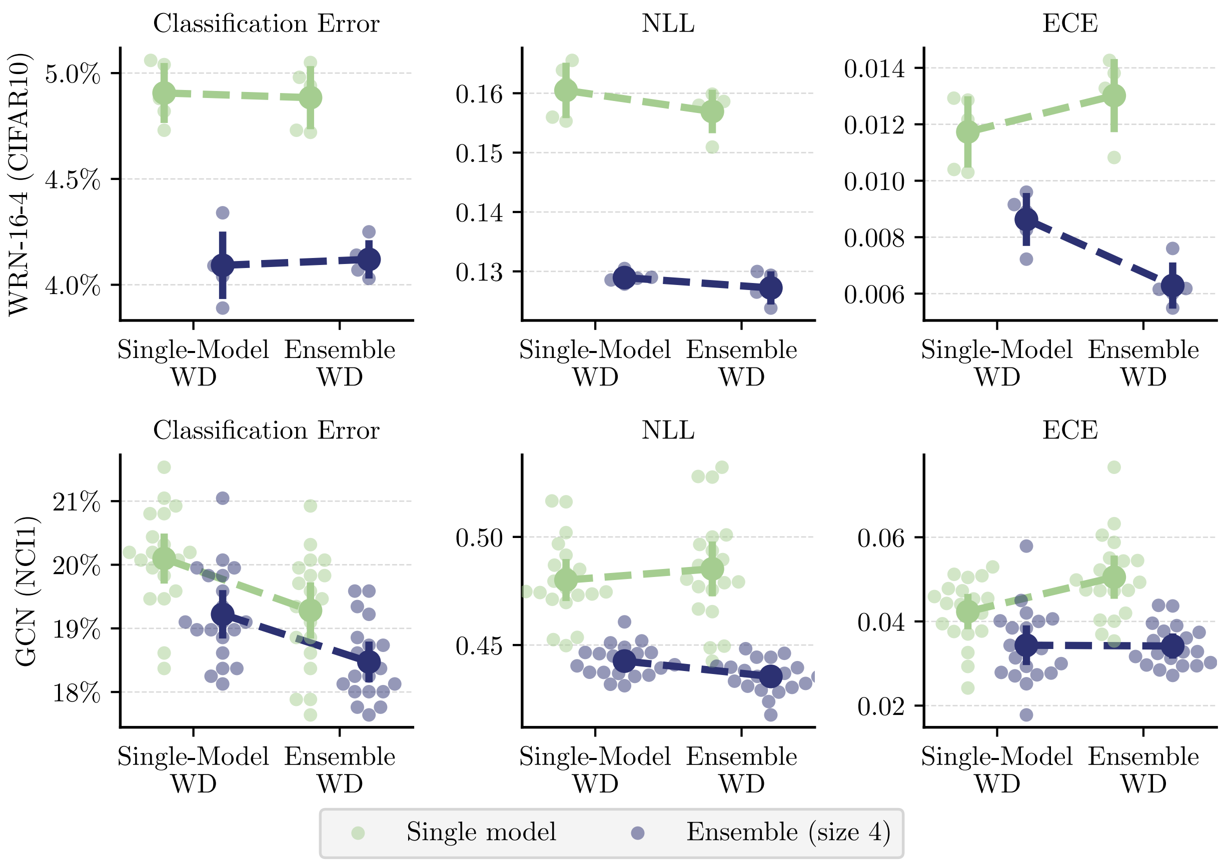

Figure 2: Test performance of single models and ensembles, trained using the optimal weight decay value tuned for single models (Single-Model WD) and the optimal weight decay value tuned for ensembles (Ensemble WD), as determined in Figure 1. (WRN: Wide ResNet; GCN: Graph Convolutional Network; NLL: negative log-likelihood; ECE: expected calibration error; WD: weight decay).

4.2 Temperature Scaling for Calibration

Purpose This experiment investigates how different temperature scaling strategies impact the calibration and overall performance of deep ensembles. We specifically compare optimizing temperature(s) for individual models versus optimizing a single temperature for joint ensemble prediction, evaluating these approaches under different validation holdout strategies and varying validation set sizes.

Method Base models were first trained using parameters from single-model weight decay tuning. Subsequently, to compare individual and joint temperature scaling strategies across shared and overlapping validation holdouts, we optimized the temperature(s) using L-BGFS to minimuze validation NLL (experimental parameters are detailed in Appendix Table 2).

Results Figure 3 presents the temperature scaling results. Across both model and dataset combinations, a trade-off emerges regarding the validation set size: increasing the percentage tends to decrease overall performance (due to reduced training data), while using very small percentages seems insufficient for robust temperature estimation. For WRN-16-4 on CIFAR-10, individual temperature scaling improves individual model calibration but degrades the ensemble's calibration (ECE). In contrast, joint temperature scaling generally improves ensemble ECE. This observation, obtained through NLL optimization, aligns with prior findings by Wu & Gales who optimized for ECE, suggesting the benefit of joint scaling is apparent regardless of which of these two metrics is optimized. This holds for both shared and overlapping holdouts. For GCN on NCI1, temperature scaling via NLL optimization showed limited ECE improvement, except at the 50% validation level. Notably, under this high percentage, the overlapping holdout strategy helped maintain ensemble performance compared to the shared holdout strategy, indicating its potential utility when data is scarce.

Conclusions The comparison indicates that joint temperature scaling based on the ensemble prediction is preferable, as individual scaling can worsen ensemble calibration. Joint scaling generally improved calibration (especially on CIFAR-10/WRN), consistent with findings from previous work using ECE optimization (Wu & Gales). The validation set size involves a critical trade-off, where reserving too much data harms overall performance. The overlapping holdout strategy shows promise, particularly in data-limited settings like NCI1.

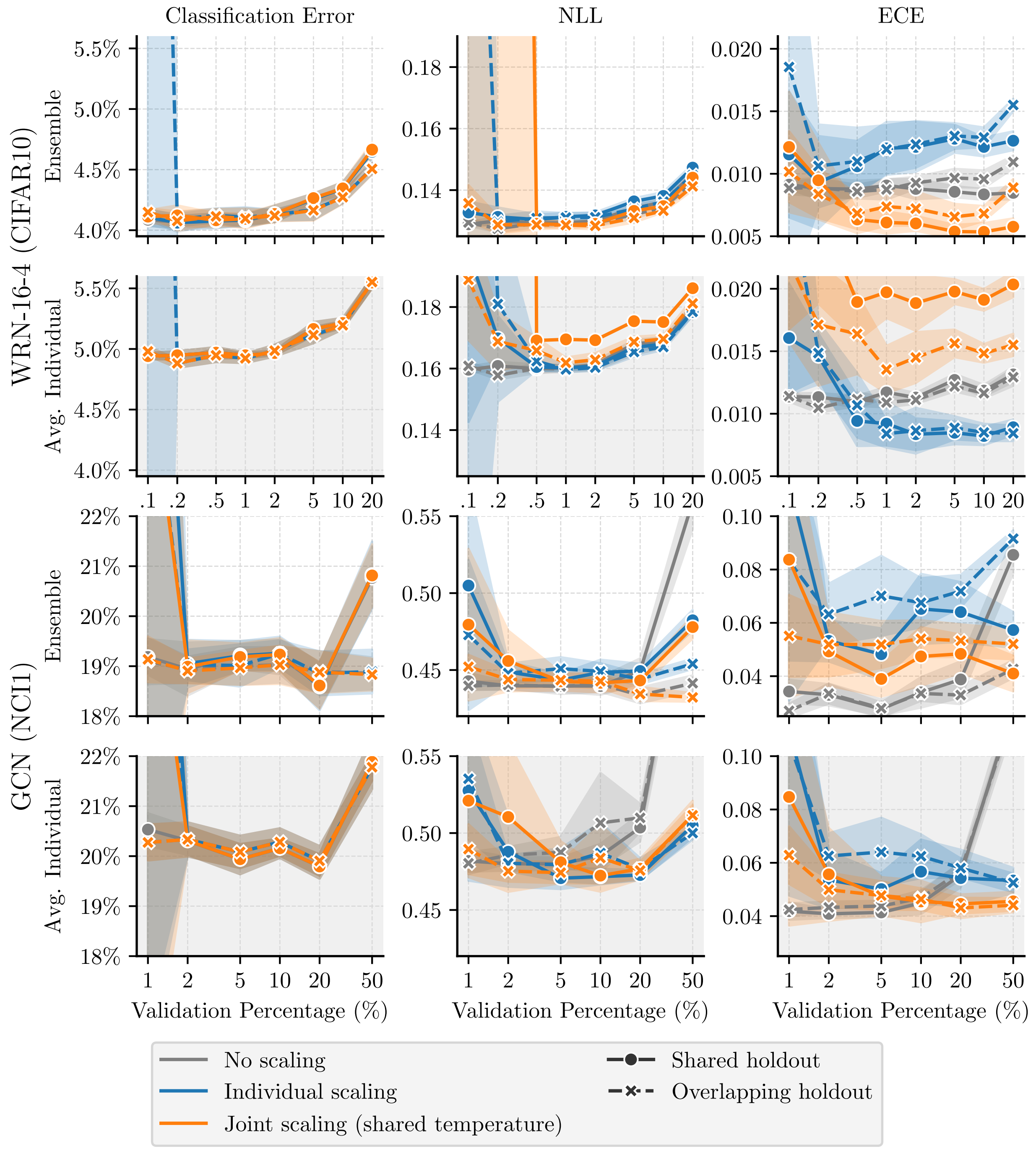

Figure 3: Temperature scaling results for ensembles ($M = 4$), comparing different temperature scaling approaches across varying validation percentages and holdout strategies. For each model/dataset we show both the metrics for the ensembles and average metrics for the individual models. (WRN: Wide ResNet; GCN: Graph Convolutional Network; NLL: negative log-likelihood; ECE: expected calibration error).

4.3 Early Stopping

Purpose This experiment aims to compare the effectiveness of individual versus joint early stopping strategies for deep ensembles. We assess their impact on key metrics including NLL, classification error, ECE, training duration, and ensemble diversity, utilizing different validation holdout methods and validation set sizes.

Method We compared individual and joint early stopping based on validation NLL using the Adam optimizer without weight decay and a patience of 10 epochs. Shared, disjoint, and overlapping holdout strategies were compared across various validation set percentages. To allow fair comparison of training duration across experiments with different validation set sizes, we report stopping times in step-normalized epochs (experimental parameters are detailed in Appendix Table 3).

Results Figure 4 presents the main performance metrics for early stopping, while Figure 5 provides additional details on stopping time and diversity. Across both WRN and GCN, joint stopping yields lower ensemble NLL compared to individual stopping, whereas individual stopping results in lower NLL for the individual models. Similarly, both classification error and ECE are improved by joint stopping. Figure 5 shows joint stopping leads to longer training durations (higher normalized stopping epochs). This extended training lowers classification error for both individual models and ensembles compared to individual stopping. However, the impact on ECE differs: individual models tend to become less calibrated (higher ECE) with longer training, while the ensembles under joint stopping either improve calibration (lower ECE on WRN-16-4) or maintain similar calibration levels (GCN on NCI1) compared to using the individual stopping strategy. Furthermore, additional results in Appendix Figure 8 show that shared holdout with joint stopping generally outperforms disjoint holdouts with individual stopping. For GCN on NCI1 at the 50% validation split (Figure 4), ensembles using overlapping holdouts perform comparably to shared holdouts, seemingly mitigating the impact of reduced training data seen in individual models, possibly linked to increased diversity (Figure 5).

Conclusions The findings indicate a clear advantage for employing joint early stopping based on the overall ensemble's validation performance. This strategy allows individual members to potentially train beyond their individual optimum, benefiting collective ensemble generalization and demonstrating the ensemble optimality gap for early stopping. While longer training can introduce an accuracy-calibration trade-off, joint stopping led to better overall ensemble performance in our experiments. Overlapping holdouts appear valuable when data limitations make a shared holdout costly. Monitoring ensemble performance for early stopping decisions is recommended.

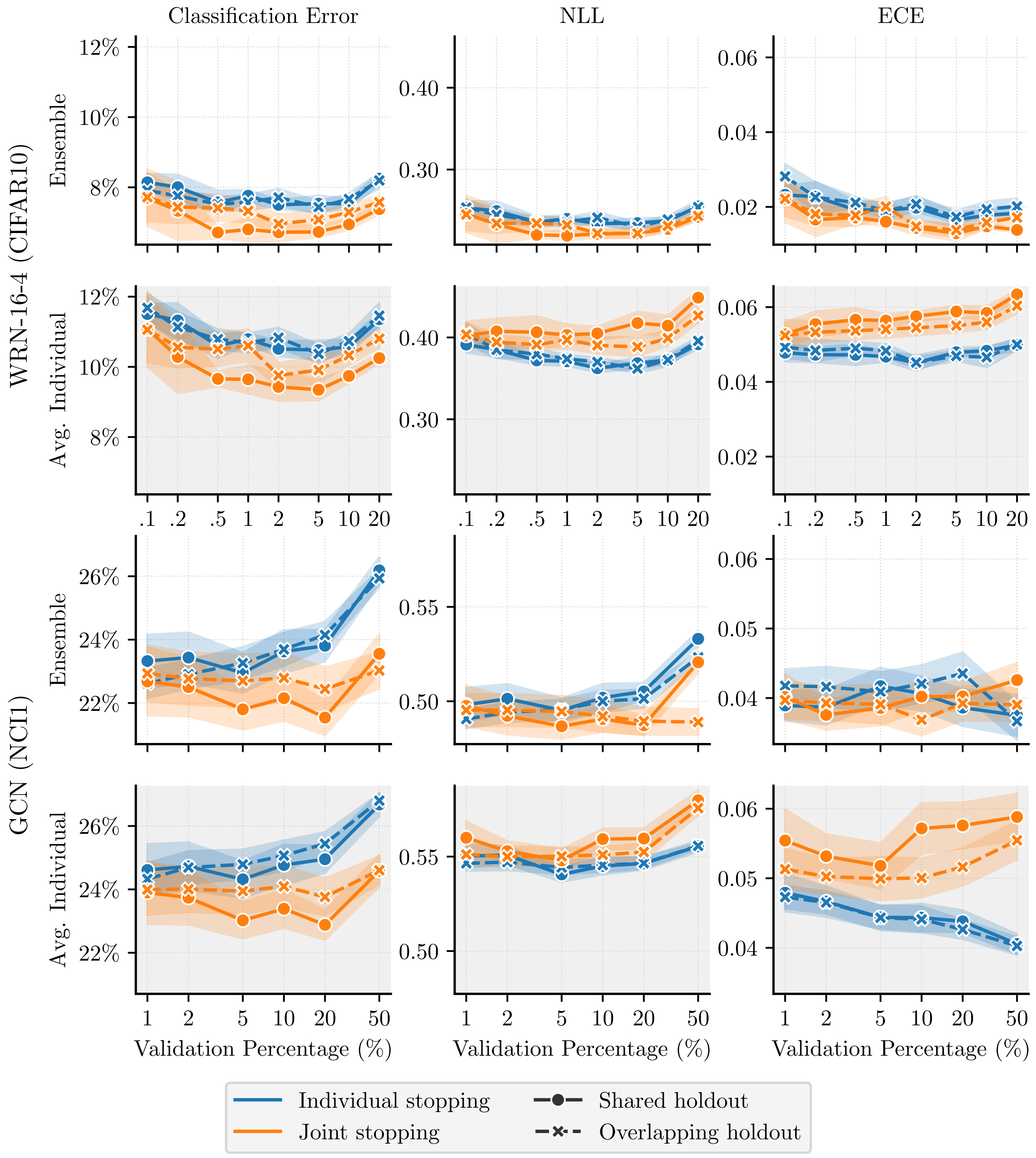

Figure 4: Early stopping performance for ensembles ($M = 4$), comparing different early stopping strategies across different validation percentages and holdout strategies. For each model/dataset we show both the metrics for the ensembles and average metrics for the individual models. (WRN: Wide ResNet; GCN: Graph Convolutional Network; NLL: negative log-likelihood; ECE: expected calibration error).

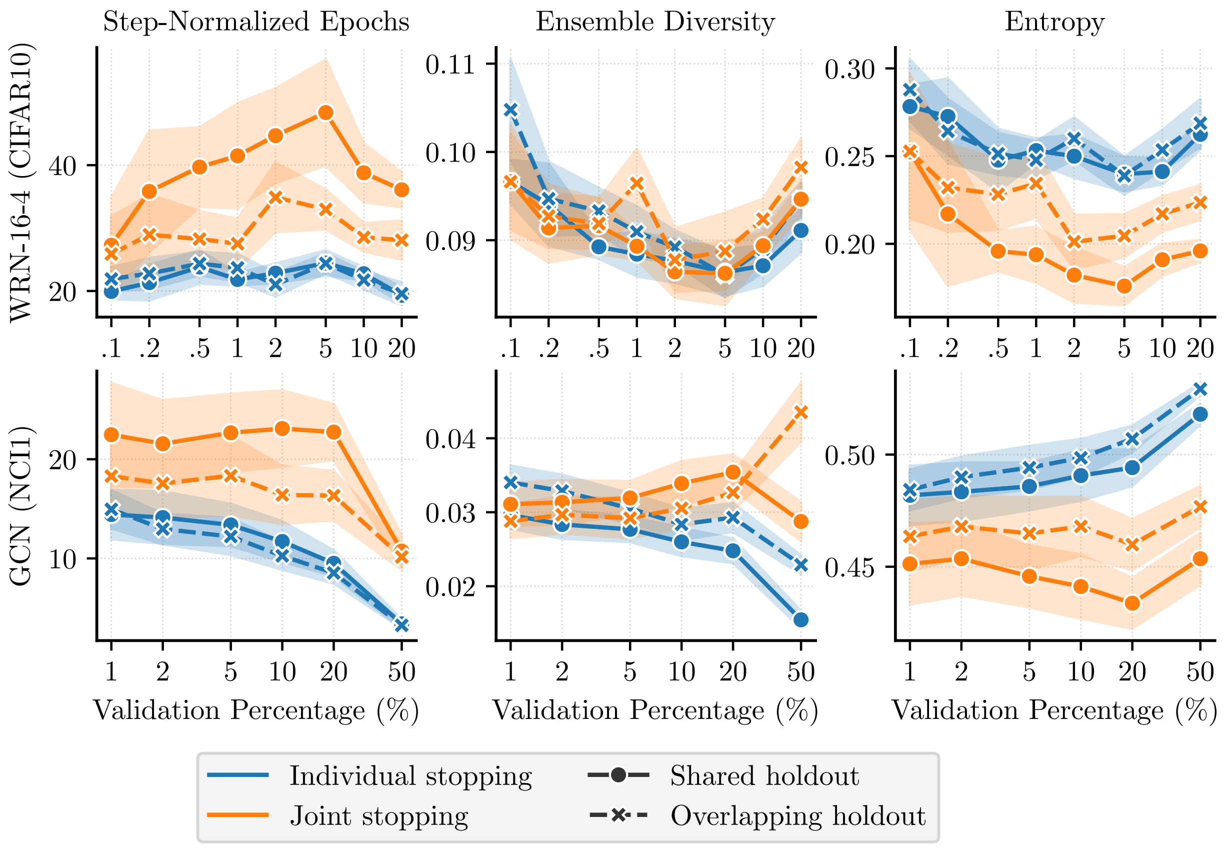

Figure 5: Additional insights into the early stopping strategies for WRN-16-4 on CIFAR-10 and GCN on NCI1. We show the stopping epoch (normalized by the number of training steps), the ensemble diversity, and the entropy of the predictive distribution. These metrics are shown across varying validation percentages and holdout strategies. (WRN: Wide ResNet; GCN: Graph Convolutional Network).

4.4 BatchEnsemble Results

Purpose Parameter-efficient methods like BatchEnsemble (Wen et al., b), utilize extensive parameter sharing between ensemble members (see Appendix A.1.3). This raises a potential concern regarding information leakage when combined with non-shared validation sets (disjoint or overlapping), as data used to validate one member might influence others through the shared parameters. Motivated by this potential issue and its possible interaction with initialization, this case study investigates BatchEnsemble performance on WRN-16-4/CIFAR-10. We specifically explore how initializing the member-specific parameters with different strategies impacts overall performance, diversity, calibration, and potential signs of leakage when using shared, disjoint, and overlapping holdout structures.

Method We implemented BatchEnsemble ($M = 4$) on CIFAR-10 using a WRN-16-4 architecture. This method employs shared *slow weights* modulated by member-specific, rank-1 *fast weights* (details in Appendix A.1.3). We investigated three fast weight initialization strategies (Gaussian $\sigma = 0.1$, $\sigma = 0.5$, and random sign ± 1) across shared, disjoint, and overlapping holdout structures (2% validation split). A key difference from standard ensembles is that BatchEnsemble's shared parameters necessitate simultaneous training termination for all members. Here we examine the early stopping procedure based on the holdout strategy: for shared and overlapping holdouts, which permit joint evaluation, we used joint early stopping based on ensemble NLL; for the disjoint strategy, where only individual validation sets exist, we stopped based on the average NLL across these individual sets. Following best practices for BatchEnsemble (Wen et al., a), separate batch normalization layers were used for each member (experimental parameters are detailed in Appendix Table 4).

Results Figure 6 shows a pronounced effect of the fast weight initialization strategy on BatchEnsemble performance. Random sign initialization consistently achieved lower NLL and better calibration (lower ECE) across all holdout strategies compared to both Gaussian initializations, alongside a higher ensemble diversity. While Gaussian initializations sometimes led to low classification error (particularly with overlapping/disjoint holdouts), their NLL and ECE were considerably worse. Figure 7, showing average individual model performance, reveals a potential issue with Gaussian initialization under non-shared holdouts: validation and training metrics closely align but diverge substantially from test metrics. This suggests poor generalization and possible data leakage, where validation data used by one member influences others through shared parameters or statistics. In contrast, random sign initialization exhibits a more typical generalization gap between training, and validation, test performance.

Conclusions The initialization strategy for fast weights is critical for BatchEnsemble performance and behavior. Random sign initialization is strongly preferable to Gaussian initialization for achieving high ensemble diversity, good calibration, and robust generalization. Furthermore, random sign initialization appears to mitigate potential validation data leakage when using non-shared holdouts, aligning BatchEnsemble's behavior more closely with standard deep ensembles. Gaussian initialization, despite potentially low errors in some settings, can lead to poor calibration and questionable generalization.

5 Discussion and Conclusion

This study systematically investigated the practical impact of adopting an ensemble-aware perspective when tuning hyperparameters and applying calibration or regularization techniques, specifically focusing on the potential mismatch between individually optimal settings—the *ensemble optimality gap*. Our experiments across weight decay tuning, temperature scaling, early stopping, different validation strategies, and BatchEnsemble provide several insights into optimizing deep ensemble performance.

Our findings regarding weight decay tuning revealed a nuanced picture of the ensemble optimality gap. While selecting weight decay based on the joint ensemble's NLL on a validation set produced performance comparable to or marginally better than using the single-model optimum in terms of NLL and classification error, the primary benefit manifested as improved calibration (ECE). The optimal weight decay value for ensemble calibration differed significantly from that for individual models, favoring less regularization. This

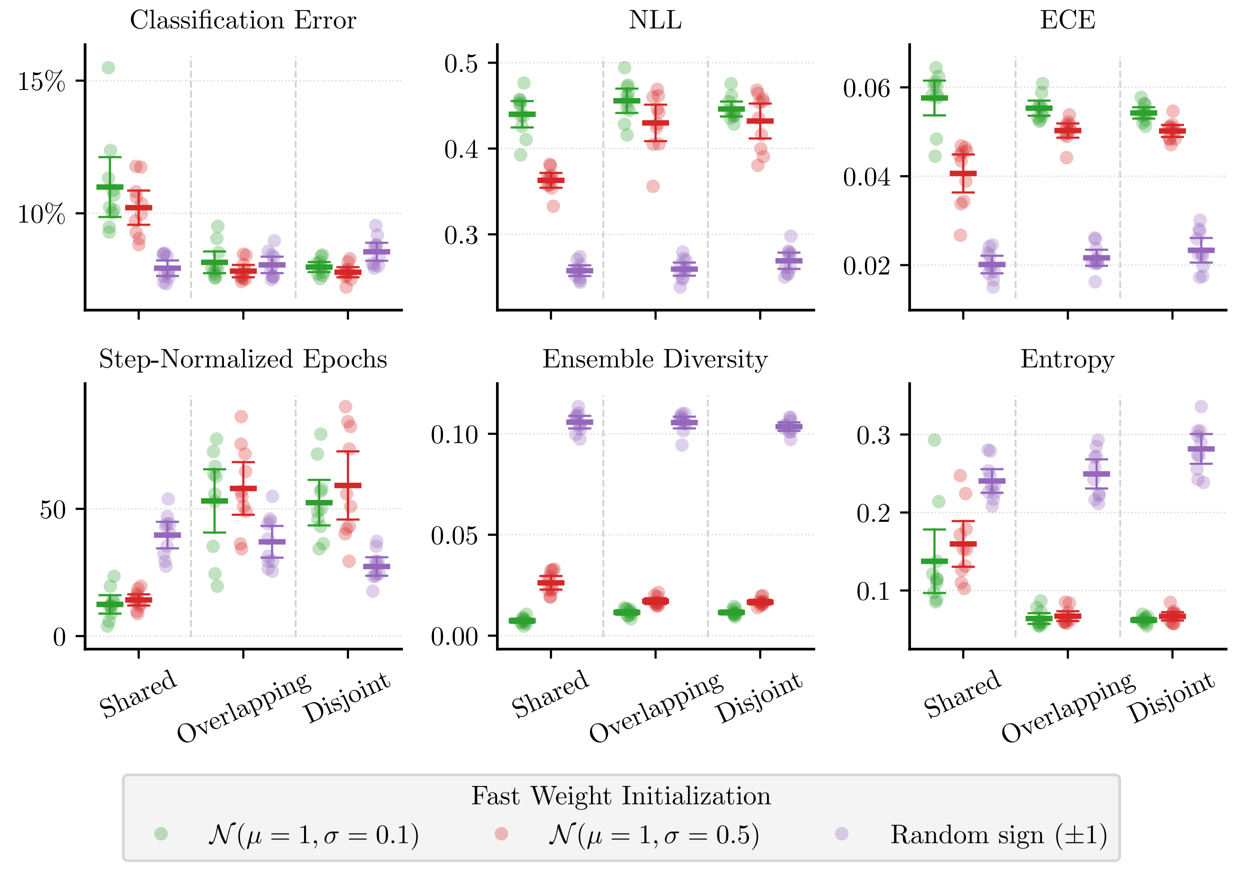

Figure 6: Performance of BatchEnsemble (M=4, WRN-16-4 on CIFAR-10) across different holdout strategies and different strategies for initializing the ensemble member specific *fast weights*. Random sign initialization tends to yield lower NLL and improved calibration (lower ECE) due to greater ensemble diversity. (WRN: Wide ResNet; NLL: negative log-likelihood; ECE: expected calibration error).

suggests that the common practice of tuning weight decay for a single model might lead to over-regularized ensembles from a calibration standpoint.

Turning to post-hoc calibration via temperature scaling, the results strongly advocate for joint optimization. Ensembling individually calibrated models was shown to degrade overall ensemble calibration compared to an uncalibrated baseline, consistent with prior findings (Wu & Gales), regardless of whether optimization is based on NLL (our work) or ECE (prior work). Conversely, optimizing a single temperature based on the joint ensemble's NLL consistently improved ensemble calibration, particularly on the multi-class WRN/CIFAR-10 task, though the benefit was less pronounced on the binary GCN/NCI1 task. Critically, these experiments highlight the need for a sufficiently large validation set for robust temperature estimation; however, using significantly more data than necessary yielded little further improvement in calibration while negatively impacting overall model performance due to the reduced training set size.

Perhaps the clearest advantage for joint optimization was observed with early stopping. Monitoring the performance of the entire ensemble and stopping only when its collective performance ceased to improve consistently led to better ensemble NLL and classification error compared to stopping members individually based on their own optima. This aligns with early ensemble theory (Sollich & Krogh), suggesting that allowing individual members to train longer-potentially slightly past their individual optimal stopping points-benefits the ensemble's generalization capacity. Although such extended training can negatively impact the calibration of individual members considered in isolation, the final ensemble calibration was often improved

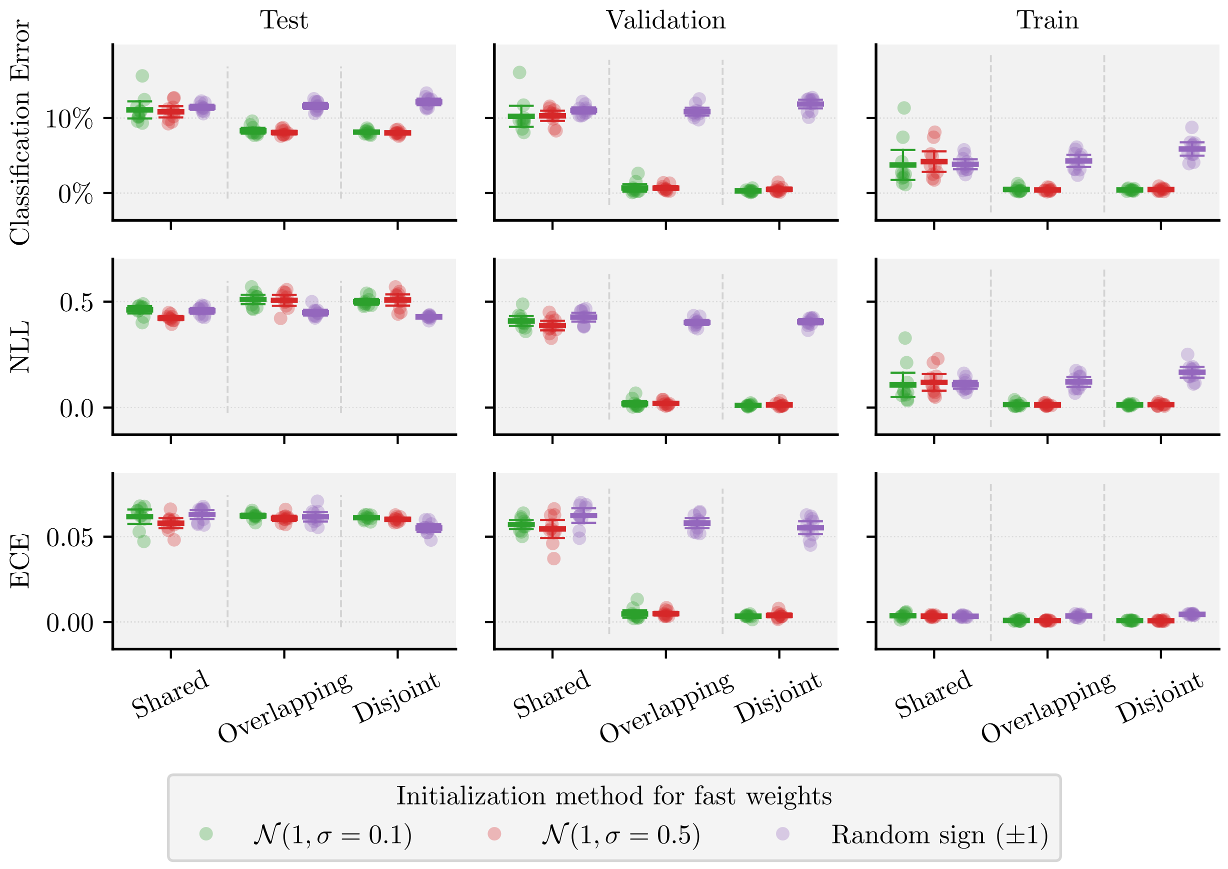

Figure 7: Average individual model performance within BatchEnsemble ($M = 4$, WRN-16-4 on CIFAR-10), comparing different initialization strategies for *fast weights* across shared, overlapping, and disjoint holdout strategies. The results for classification error, NLL, and ECE are shown for the test, validation, and training sets. Notably, for Gaussian initialization with overlapping and disjoint holdouts, the close alignment of validation and training performance (as opposed to test performance) suggests potential data leakage between ensemble members. This observation further highlights the superior generalization achieved with random sign initialization. (WRN: Wide ResNet; NLL: negative log-likelihood; ECE: expected calibration Error).

(WRN-16-4) or maintained (GCN) compared to using individual stopping. Notably, implementing joint early stopping adds minimal computational overhead relative to individual stopping; the primary difference being potentially longer training runs. This makes it a practical and inexpensive way to leverage ensemble effects, unlike potentially costly joint hyperparameter grid searches. Joint early stopping thus provides a compelling and practical avenue to close the ensemble optimality gap, capitalizing on significant performance improvements at little extra computational cost.

The choice of validation data strategy critically determines the feasibility of joint optimization techniques and involves fundamental trade-offs. A shared holdout enables straightforward joint evaluation of the full ensembles but requires reserving a common validation set that is never used for training by any member. Conversely, the disjoint holdout strategy maximizes the utilization of the available non-test data pool for training across the ensemble-ensuring every data point contributes to training all but one member-but completely precludes joint evaluation on validation data unseen by all members being evaluated. Our results for early stopping and temperature scaling suggest that the benefits derived from enabling robust joint evaluation outweigh the potential advantages of maximizing this training data utilization. The proposed overlapping holdout strategy offers a middle ground: similar to disjoint, it ensures all non-test data is used for training somewhere within the ensemble, but by creating specific overlaps within the validation sets assigned to each member, it permits pairwise joint evaluation. This makes it a practical compromise, particularly in low-data regimes where reserving a fully shared holdout might be too costly because it prevents the entire portion of data from being used in training by any model.

The importance of considering ensemble interactions also extends to parameter-efficient methods like BatchEnsemble. Our investigation underscores the paramount importance of the initialization strategy for the member-specific *fast weights* (the trainable rank-1 vectors), a detail potentially overlooked. We found that random sign initialization (assigning ± 1 values) proved crucial for achieving high ensemble diversity and good calibration, behaving much like a standard ensemble. This aligns with some earlier observations regarding its effectiveness (Wenzel et al.). In stark contrast, Gaussian initialization (i.e., sampling from $\mathcal{N}(1, \sigma^2)$, although used in some large-scale applications (Tran et al.; Dehghani et al.), performed poorly in our experiments, showing lower diversity and signs of data leakage with non-shared holdouts. The potential confusion regarding initialization practices is highlighted by reports using ambiguous descriptions like "random sign initialization... of -0.5" (meaning Gaussian initialization) at large scale (Dehghani et al.). Given the stark difference we observed, and the lack of direct comparisons at scale in cited works, further investigation into the impact of initialization on BatchEnsemble's effectiveness seems warranted. Our findings strongly suggest that proper (random sign) initialization leads to robust BatchEnsemble behavior, crucially mitigating data leakage issues with non-shared holdouts and thereby enabling the effective and reliable use of different validation strategies, including meaningful joint evaluation when the structure allows (like overlapping holdouts).

Across these diverse experiments, a unifying principle emerges: the significance of the *ensemble optimality gap* and the practical value of adopting an ensemble-aware perspective. Evaluating and optimizing based on the joint ensemble's behavior during validation-dependent procedures consistently led to ensembles that were more accurate, better calibrated, or both. Our results suggest practitioners should prioritize robust joint evaluation strategies-especially for computationally inexpensive procedures like early stopping and temperature scaling-rather than solely relying on individually tuned components or potentially complex methods aimed at explicitly maximizing diversity during training, which may not always be necessary or beneficial, particularly for large models (Abe et al., b;a). Ultimately, the ensemble is the final predictor, and evaluating it directly during optimization yields better results.

Based on these findings, we offer the following practical recommendations for practitioners training deep ensembles:

- **Weight Decay:** Start by finding the optimal weight decay for a single model. This value provides a reasonable baseline. Then, consider validating the ensemble performance with this weight decay and potentially slightly lower values, especially if ensemble calibration is the primary goal and NLL is the tuning metric. Approximating joint performance by ensembling models trained with nearby weight decay values during a sweep might also be explored.
- **Temperature Scaling:** Always optimize and apply temperature scaling *jointly* based on the ensemble's performance (e.g., minimizing NLL) on a validation set. Avoid calibrating members individually. Crucially, use only a reasonably small validation set; dedicating too much data can unnecessarily harm overall model performance.
- **Early Stopping:** Monitor the validation performance of the *entire ensemble* to determine the stopping point. This allows the ensemble to train longer and achieve better performance compared to stopping based on individual member optima. Ensure the validation set size is appropriate.
- **Validation Strategy:** Use a *shared holdout* set whenever possible to allow for direct joint evaluation. If data is extremely limited, making a shared holdout prohibitively costly, the *overlapping holdout* strategy offers a viable alternative that retains some joint evaluation capability while maximizing data use.
- **BatchEnsemble:** Strongly prefer *random sign initialization* for the fast weights over Gaussian initialization to maximize ensemble diversity, calibration, and robustness, particularly when using disjoint or overlapping holdout strategies.

While our findings across image and graph classification tasks show consistent trends, this study has limitations. We explored only two model/dataset combinations, and did not investigate extremely large-scale models where ensembling dynamics might differ, and joint tuning costs escalate. Whether Gaussian initialization for BatchEnsemble could yield sufficient diversity and stability in such large models remains an open question, given lack of direct comparisons in the literature we surveyed. Furthermore, the implications for other types of ensembles (BNNs, MC-dropout) warrant investigation. Future work could explore these joint optimization dynamics in other domains, develop cost-effective heuristics for joint tuning, investigate variations of the overlapping holdout strategy, such as structures enabling higher-order joint evaluation (e.g., among member triplets), and further study the interplay between model/dataset characteristics and validation strategies.

In conclusion, this work highlights the practical importance of considering ensemble effects during the tuning and calibration process. By adopting an ensemble-aware perspective and leveraging joint evaluation, particularly for computationally efficient techniques like early stopping and temperature scaling, practitioners can build more accurate and reliable deep ensemble models.

References

Taiga Abe, E. Kelly Buchanan, Geoff Pleiss, and John P. Cunningham. Pathologies of Predictive Diversity in Deep Ensembles, a. URL `https://arxiv.org/abs/2302.00704`.

Taiga Abe, E. Kelly Buchanan, Geoff Pleiss, Richard Zemel, and John P. Cunningham. Deep Ensembles Work, But Are They Necessary?, b. URL `http://arxiv.org/abs/2202.06985`.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. 13(10):281–305.

Leo Breiman. Bagging predictors. 24(2):123–140. ISSN 0885-6125, 1573-0565. doi: 10.1007/BF00058655.

Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme Ruiz, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd Van Steenkiste, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Collier, Alexey A. Gritsenko, Vighnesh Birodkar, Cristina Nader Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetic, Dustin Tran, Thomas Kipf, Mario Lucic, Xiaohua Zhai, Daniel Keysers, Jeremiah J. Harmsen, and Neil Houlsby. Scaling vision transformers to 22 billion parameters. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 7480–7512. PMLR. URL `https://proceedings.mlr.press/v202/dehghani23a.html`.

Thomas G. Dietterich. Ensemble Methods in Machine Learning. In Gerhard Goos, Juris Hartmanis, and Jan Van Leeuwen (eds.), *Multiple Classifier Systems*, volume 1857, pp. 1–15. Springer Berlin Heidelberg. ISBN 978-3-540-67704-8 978-3-540-45014-6. doi: 10.1007/3-540-45014-9_1.

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. URL `http://arxiv.org/abs/2005.07186`.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep Ensembles: A Loss Landscape Perspective. URL `http://arxiv.org/abs/1912.02757`.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1050–1059. PMLR. URL `https://proceedings.mlr.press/v48/gal16.html`.

Tilmann Gneiting and Adrian E Raftery. Strictly Proper Scoring Rules, Prediction, and Estimation. 102 (477):359–378. ISSN 0162-1459, 1537-274X. doi: 10.1198/016214506000001437.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. TabM: Advancing Tabular Deep Learning with Parameter-Efficient Ensembling. URL `http://arxiv.org/abs/2410.24210`.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. URL `http://arxiv.org/abs/1706.04599`.

L.K. Hansen and P. Salamon. Neural network ensembles. 12(10):993–1001. ISSN 01628828. doi: 10.1109/34.58871.

L.K. Hansen, J. Larsen, and T. Fog. Early stop criterion from the bootstrap ensemble. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pp. 3205–3208 vol.4. doi: 10.1109/ICASSP.1997.595474. URL `https://ieeexplore.ieee.org/document/595474`.

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. URL `http://arxiv.org/abs/2010.06610`.

Lena Heidemann, Adrian Schwaiger, and Karsten Roscher. Measuring Ensemble Diversity and its Effects on Model Robustness. In *2021 Workshop on Artificial Intelligence Safety, AISafety 2021*. CEUR-WS. doi: 10.24406/PUBLICA-FHG-411897. URL `https://ceur-ws.org/Vol-2916/paper_8.pdf`.

Siddhartha Jain, Ge Liu, Jonas Mueller, and David Gifford. Maximizing Overall Diversity for Improved Uncertainty Estimates in Deep Ensembles. URL `http://arxiv.org/abs/1906.07380`.

Alan Jeffares, Tennison Liu, Jonathan Crabbé, and Mihaela van der Schaar. Joint Training of Deep Ensembles Fails Due to Learner Collusion. URL `http://arxiv.org/abs/2301.11323`.

Hongpeng Jin and Yanzhao Wu. Boosting Deep Ensembles with Learning Rate Tuning. URL `http://arxiv.org/abs/2410.07564`.

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images.

Anders Krogh and Peter Sollich. Statistical mechanics of ensemble learning. 55(1):811–825. ISSN 1063-651X, 1095-3787. doi: 10.1103/PhysRevE.55.811.

Ludmila I. Kuncheva and Christopher J. Whitaker. Measures of Diversity in Classifier Ensembles and Their Relationship with the Ensemble Accuracy. 51(2):181–207. ISSN 08856125. doi: 10.1023/A:1022859003006.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. URL `http://arxiv.org/abs/1612.01474`.

Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks. URL `http://arxiv.org/abs/1511.06314`.

Y. Liu and X. Yao. Ensemble learning via negative correlation. 12(10):1399–1404. ISSN 08936080. doi: 10.1016/S0893-6080(99)00073-8.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining Well Calibrated Probabilities Using Bayesian Binning. 29(1). ISSN 2374-3468. doi: 10.1609/aaai.v29i1.9602.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pp. 625–632. Association for Computing Machinery. ISBN 978-1-59593-180-1. doi: 10.1145/1102351.1102430. URL `https://dl.acm.org/doi/10.1145/1102351.1102430`.

Matteo Pagliardini, Martin Jaggi, François Fleuret, and Sai Praneeth Karimireddy. Agree to Disagree: Diversity through Disagreement for Better Transferability. URL `http://arxiv.org/abs/2202.04414`.

Lutz Prechelt. Automatic early stopping using cross validation: Quantifying the criteria. 11(4):761–767. ISSN 0893-6080. doi: 10.1016/S0893-6080(98)00010-0.

Lorena Qendro, Alexander Campbell, Pietro Lio, and Cecilia Mascolo. Early Exit Ensembles for Uncertainty Quantification.

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. Weisfeiler-lehman graph kernels. 12(77):2539–2561.

Peter Sollich and Anders Krogh. Learning with ensembles: How overfitting can be useful. In D. Touretzky, M.C. Mozer, and M. Hasselmo (eds.), *Advances in Neural Information Processing Systems*, volume 8. MIT Press. URL `https://proceedings.neurips.cc/paper_files/paper/1995/file/1019c8091693ef5c5f55970346633f92-Paper.pdf`.

Yongduo Sui, Xiang Wang, Jiancan Wu, Min Lin, Xiangnan He, and Tat-Seng Chua. Causal Attention for Interpretable and Generalizable Graph Classification. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1696–1705. ACM. ISBN 978-1-4503-9385-0. doi: 10.1145/3534678.3539366. URL `https://dl.acm.org/doi/10.1145/3534678.3539366`.

Cedrique Rovile Njieutcheu Tassi, Jakob Gawlikowski, Auliya Unnisa Fitri, and Rudolph Triebel. The impact of averaging logits over probabilities on ensembles of neural networks. In *CEUR Workshop Proceedings*, volume 3215. CEUR-WS. URL `https://ceur-ws.org/Vol-3215/19.pdf`.

Dustin Tran, Jeremiah Liu, Michael W. Dusenberry, Du Phan, Mark Collier, Jie Ren, Kehang Han, Zi Wang, Zelda Mariet, Huiyi Hu, Neil Band, Tim G. J. Rudner, Karan Singhal, Zachary Nado, Joost van Amersfoort, Andreas Kirsch, Rodolphe Jenatton, Nithum Thain, Honglin Yuan, Kelly Buchanan, Kevin Murphy, D. Sculley, Yarin Gal, Zoubin Ghahramani, Jasper Snoek, and Balaji Lakshminarayanan. Plex: Towards Reliability using Pretrained Large Model Extensions. URL `http://arxiv.org/abs/2207.07411`.

Nikil Wale, Ian A. Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. 14(3):347–375. ISSN 0219-1377, 0219-3116. doi: 10.1007/s10115-007-0103-5.

Yeming Wen, Ghassen Jerfel, Rafael Muller, Michael W. Dusenberry, Jasper Snoek, Balaji Lakshminarayanan, and Dustin Tran. Combining Ensembles and Data Augmentation can Harm your Calibration, a. URL `http://arxiv.org/abs/2010.09875`.

Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: An Alternative Approach to Efficient Ensemble and Lifelong Learning, b. URL `http://arxiv.org/abs/2002.06715`.

Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6514–6527. Curran Associates, Inc. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/481fbfa59da2581098e841b7afc122f1-Paper.pdf`.

Danny Wood, Tingting Mu, Andrew M. Webb, Henry W. J. Reeve, Mikel Luján, and Gavin Brown. A unified theory of diversity in ensemble learning. 24(1). ISSN 1532-4435.

Xixin Wu and Mark Gales. Should Ensemble Members Be Calibrated? URL `http://arxiv.org/abs/2101.05397`.

Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. URL `http://arxiv.org/abs/1605.07146`.

A Model, Dataset, and Experimental Details

A.1 Model Details

A.1.1 Wide ResNet-16-4 (WRN-16-4)

For our experiments on the CIFAR-10 dataset, we utilize the WRN-16-4 architecture (Zagoruyko & Komodakis) as one of our base models for deep ensembles. WRN architectures are characterized by their wider convolutional layers and reduced depth compared to traditional ResNet architectures. Specifically, WRN-16-4 denotes a WRN with 16 convolutional layers and a widening factor of 4. In our implementation, we use the WRN variant where the placement of batch normalization, ReLU activation, and convolution layers follows the order: batch normalization - ReLU - convolution. We do not employ dropout regularization in our models.

A.1.2 Graph Convolutional Network (GCN)

For our experiments on the NCI1 dataset, we utilize a four-layer GCN, based on the architecture presented in Sui et al., but extended to four GCN layers. The network consists of an initial feature transformation layer, followed by four GCN layers, and finally two fully connected layers.

We use ReLU activations after the initial feature transformation and each of the four GCN layers. Batch normalization is applied to the input features before the initial transformation, after each of the four GCN layers, and before each of the two fully-connected layers.

Global sum pooling is applied after the final GCN layer to obtain a graph-level representation. This representation is then passed through a sequence of two fully-connected layers. The first fully connected layer applies batch normalization, followed by a ReLU activation and a linear transformation. The second fully-connected layer is the classification layer and produces the final output logits.

A.1.3 BatchEnsemble Implementation

BatchEnsemble (Wen et al., b) modifies a base network to efficiently train an ensemble using shared weights W (slow weights). For each shared weight W, member-specific weights W_i ($i = 1...M$) are generated using trainable rank-1 vectors $\mathbf{r}_i$ and $\mathbf{s}_i$ (fast weights) as $W_i = W \circ (\mathbf{r}_i \mathbf{s}_i^T)$, where $\circ$ denotes the Hadamard (element-wise) product. Crucially, this rank-1 structure allows the forward pass computations for all M ensemble members to be efficiently vectorized into a single operation, enabling parallel execution on hardware accelerators (Wen et al., b). The initialization of these fast weights is known to be important for performance. As investigated in Section 4.4, our experiments specifically compare three initialization strategies for $\mathbf{r}_i$ and $\mathbf{s}_i$: initializing elements from a Gaussian distribution with mean 1 and standard deviation σ (either $\sigma = 0.1$ or $\sigma = 0.5$), versus initializing elements randomly as ± 1 (random sign / Rademacher distribution). Consistent with recommendations for achieving good performance with BatchEnsemble (Wen et al., a), we use separate batch normalization layers for each ensemble member throughout our experiments.

A.2 Dataset Details

A.2.1 CIFAR-10

We conduct experiments on the CIFAR-10 dataset (Krizhevsky), a widely used benchmark for image classification. CIFAR-10 consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images, and we use the original test set for final performance evaluation in all experiments.

Following common practice for CIFAR-10, we apply the following preprocessing steps: random cropping to size 32 with a padding of 4 pixels, random horizontal flipping with a probability of 0.5, and normalization. The normalization constants (mean and standard deviation for each color channel) are calculated based on the actual training split to prevent contamination from the test and validation data.

A.2.2 NCI1

We conduct experiments on the NCI1 dataset (Shervashidze et al.; Wale et al.), a graph classification dataset with two classes. There are 4110 graphs. We randomly split the dataset into a training set (80%) and a test set (20%) using stratified sampling to ensure class balance, resulting in 3288 training graphs and 822 test graphs. The test set is kept fixed across all experiments. We do not apply any specific preprocessing to the node features or graph structure beyond what is inherent in the dataset.

A.3 Experimental Setup Details

The following tables provide detailed hyperparameters and setup configurations for the experiments presented in Section 4. Each table corresponds to a specific experimental procedure, as referenced in the respective Method paragraphs within Section 4.

Table 1: Hyperparameter tuning (weight decay) parameters.

Parameter	**WRN-16-4 on CIFAR-10**	**GCN on NCI1**
Weight Decay Values	25 log-spaced (3.16e-5 to 1.00e-2)*	17 log-spaced (1e-3 to 1e-2)
Optimizer	SGD	SGD
Momentum	0.9	0.9
Learning Rate Schedule	Cosine annealing (initial 0.1, to 0)	Cosine annealing (initial 0.1, to 0)
Epochs	100	200
Batch Size	128	128
Validation Strategy	Shared holdout (10% stratified)	Shared holdout (10% stratified)
Number of Seeds	5	20
Selection Metric	Average validation NLL	Average validation NLL

* For WRN-16-4, results in Figure 1 are plotted only up to 3.16e-3 as higher values led to severely degraded performance.

Table 2: Temperature scaling parameters

Parameter	**WRN-16-4 on CIFAR-10**	**GCN on NCI1**
Base Model Weight Decay	7.5e-4	3.162e-3
Optimization Algorithm	L-BFGS (lr=0.1, max_iter=100)	L-BFGS (lr=0.1, max_iter=100)
Initial Temperature	1.0	1.0
Validation Strategies	Shared, overlapping	Shared, overlapping
Validation Percentages	0.1%, 0.2%, 0.5%, 1%, 2%, 5%, 10%, 20%	1%, 2%, 5%, 10%, 20%, 50%
Number of Seeds	10	10
Ensemble Size	4	4
Scaling Types	Individual, joint	Individual, joint
Optimization Metric	Validation NLL	Validation NLL

B Diversity Metric

The ensemble diversity metric used in this paper (Sections 2.1, 4.3, 4.4) quantifies the disagreement among ensemble members for a given input data point i. It is defined as the difference between the entropy of the average predicted probability vector $\bar{\boldsymbol{p}}_i = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{p}_i^m$ and the average entropy of the individual member

Table 3: Early stopping parameters

Parameter	**WRN-16-4 on CIFAR-10**	**GCN on NCI1**
Optimizer	Adam	Adam
Learning Rate	0.001	0.001
Weight Decay	0	0
Validation Strategies	Shared, overlapping, disjoint	Shared, overlapping, disjoint
Stopping Criterion	Validation NLL	Validation NLL
Patience	10 epochs	10 epochs
Validation Percentages	0.1%, 0.2%, 0.5%, 1%, 2%, 5%, 10%, 20%	1%, 2%, 5%, 10%, 20%, 50%
Number of Seeds	10	50
Ensemble Size	4	4
Stopping Types	Individual, joint	Individual, joint

Table 4: BatchEnsemble Parameters for WRN-16-4 on CIFAR-10

Parameter	**WRN-16-4 BatchEnsemble on CIFAR10**
Ensemble Size	4
Optimizer	Adam
Learning Rate	0.001
Weight Decay	0
Validation Strategies	Shared, disjoint, overlapping
Stopping Criterion	Validation NLL
Patience	10 epochs
Validation Percentage	2%
Number of Seeds	10
Fast Weight Initialization	$\mathcal{N}(\mu = 1, \sigma = 0.1)$, $\mathcal{N}(\mu = 1, \sigma = 0.5)$, random sign ($\pm 1$)
Batch Normalization	Separate for each ensemble member

predictions $\boldsymbol{p}_i^m$:

$$D_i = H(\bar{\boldsymbol{p}}_i) - \frac{1}{M} \sum_{m=1}^{M} H(\boldsymbol{p}_i^m)$$

where $H(\boldsymbol{p}) = -\sum_c \boldsymbol{p}[c] \log \boldsymbol{p}[c]$ is the Shannon entropy.

This measure is directly related to the Kullback-Leibler (KL) divergence. It can be shown that the diversity defined above is equal to the average KL divergence from the individual member predictions $\boldsymbol{p}_i^m$ to the mean ensemble prediction $\bar{\boldsymbol{p}}_i$:

$$D_i = \frac{1}{M} \sum_{m=1}^{M} D_{KL}(\boldsymbol{p}_i^m || \bar{\boldsymbol{p}}_i)$$

where $D_{\mathrm{KL}}(\boldsymbol{p}||\bar{\boldsymbol{p}}) = \sum_c \boldsymbol{p}[c] \log(\boldsymbol{p}[c]/\bar{\boldsymbol{p}}[c])$. Therefore, this diversity metric intuitively quantifies how much, on average, the probability distribution predicted by an individual member diverges from the consensus prediction of the ensemble. Higher values indicate greater disagreement among ensemble members.

C Early Stopping Performance Under Shared vs. Disjoint Holdout Strategies

While the main text (Figure 4) compared shared and overlapping holdout sets for early stopping, this appendix section specifically addresses the performance of the disjoint holdout strategy. Figure 8 directly contrasts ensembles using disjoint holdouts (evaluated with individual stopping) against those using shared holdouts (evaluated with both individual and joint early stopping). The disjoint strategy is characterized by ensuring that the holdout data for any one model is part of the training data for all other models in the ensemble, thus maximizing data usage across the group but precluding joint evaluation. The results, presented for WRN-16-4/CIFAR-10 and GCN/NCI1 across various validation percentages, illustrate the performance impact of this approach. The comparison suggests that the performance improvement obtained by applying joint early stopping (using shared holdout sets) is generally greater than any advantage gained from the improved non-test coverage of the disjoint holdout sets.

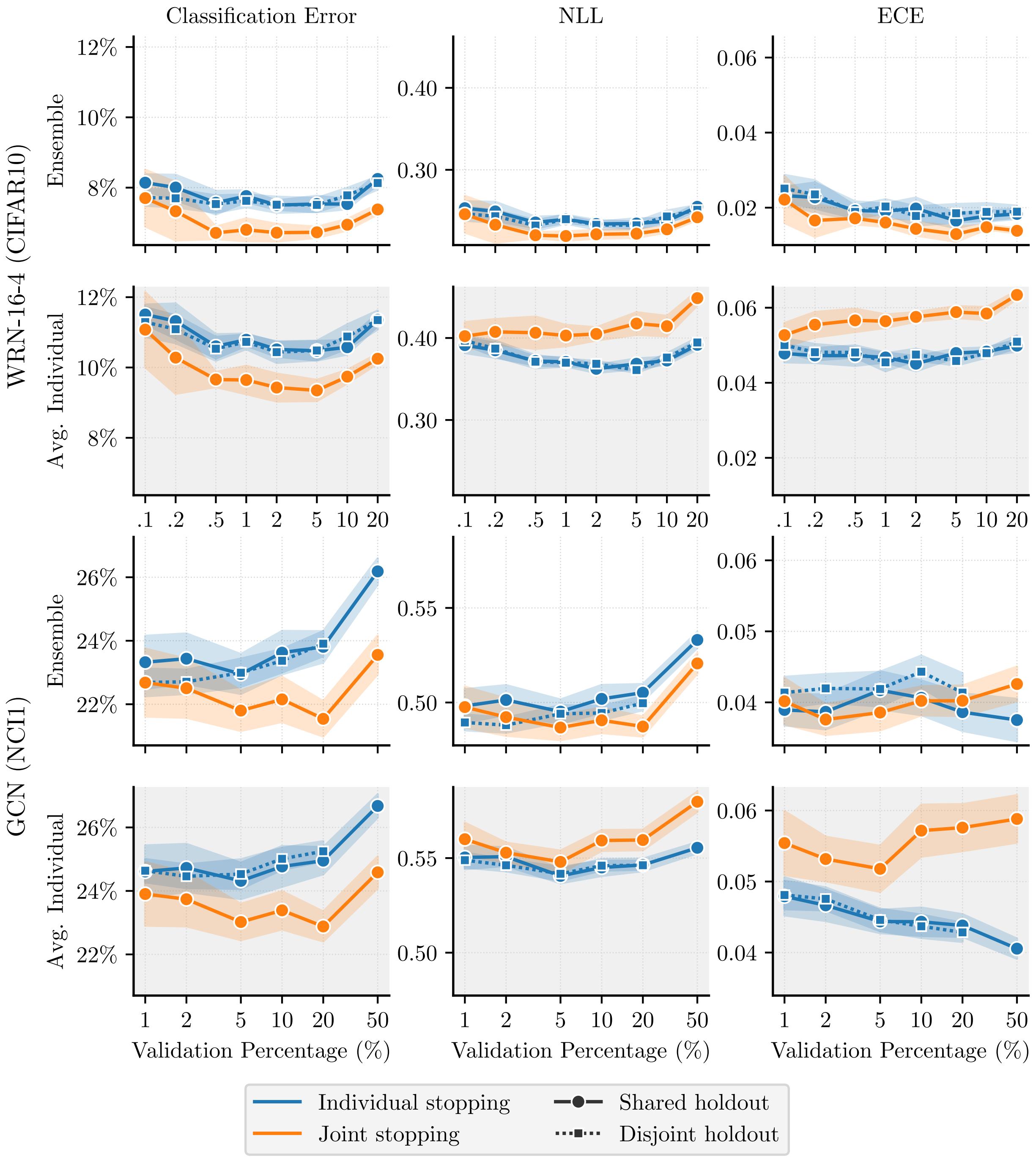

Figure 8: Early stopping test performance comparing disjoint holdout (individual stopping only) and shared holdout (individual and joint stopping) strategies for ensembles ($M = 4$) across varying validation percentages. (WRN: Wide ResNet; GCN: Graph Convolutional Network; NLL: negative log-likelihood; ECE: expected calibration error).