FiLM-Ensemble: Probabilistic Deep Learning via Feature-wise Linear Modulation

Anonymous Author(s)
Affiliation
Address
email

Abstract

The ability to estimate epistemic uncertainty is often crucial when deploying machine learning in the real world, but modern methods often produce overconfident, uncalibrated uncertainty predictions. A common approach to quantify epistemic uncertainty, usable across a wide class of prediction models, is to train a *model ensemble*. In a naïve implementation, the ensemble approach has high computational cost and high memory demand. This challenges in particular modern deep learning, where even a single deep network is already demanding in terms of compute and memory, and has given rise to a number of attempts to emulate the model ensemble without actually instantiating separate ensemble members. We introduce FiLM-Ensemble, a deep, implicit ensemble method based on the concept of Feature-wise Linear Modulation (FiLM). That technique was originally developed for multi-task learning, with the aim of decoupling different tasks. We show that the idea can be extended to uncertainty quantification: by modulating the network activations of a single deep network with FiLM, one obtains a model ensemble with high diversity, and consequently well-calibrated estimates of epistemic uncertainty, with low computational overhead in comparison. Empirically, FiLM-Ensemble outperforms other implicit ensemble methods, and it and comes very close to the upper bound of an explicit ensemble of networks (sometimes even beating it), at a fraction of the memory cost.

1 Introduction

A key component for reliable and trustworthy machine learning are algorithms that output not only accurate predictions of the target variables, but also well-calibrated estimates of their uncertainty Gal and Ghahramani [2016]. The overall uncertainty of a predictor is usually decomposed into two parts [Der Kiureghian and Ditlevsen, 2009]. *Aleatoric uncertainty* is inherent in the data, for instance due to class overlap or sensor noise. On the contrary, *epistemic uncertainty* characterises the uncertainty of the model weights, due to a lack of knowledge about parts of the input space that are insufficiently represented in the training set. Uncertainty caused by distributional shifts between the training and test data is sometimes conceived as a third source of uncertainty [Malinin and Gales, 2018], but in practice often modelled as part of epistemic uncertainty.

Measuring epistemic uncertainty for complex models such as deep neural networks is not trivial: by definition one cannot derive it from the training data, since it concerns the behavior of the model in regions of the input space that are not represented in the training set. Different methods have been explored [Welling and Teh, 2011, Graves, 2011, Blundell et al., 2015, Lakshminarayanan et al., 2017a, Huang and Belongie, 2017, Gal and Ghahramani, 2016], however the de facto standard remain deep ensemble models [Lakshminarayanan et al., 2017a]. In its basic form, such a *deep ensemble* is simply a collection of independently trained networks that can be regarded as Monte Carlo samples from the model space. To obtain ensemble members with reasonably low correlation, one can exploit

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

the stochastic nature of the optimisation procedure, with different (random) weight initialisation and different (random) batches during stochastic gradient descent. The expectation is that, once trained, the ensemble members will agree for inputs near the training samples, since the loss function favours similar outputs at those locations. Whereas they may disagree in unseen regions of the data space. Thus, the spread of their predictions is a measure of epistemic uncertainty. The ensemble idea is conceptually very simple, but nevertheless yields uncertainties that are well calibrated, i.e., they are in line with the actual prediction errors (in expectation).

The drawback of deep ensembles is their large computational cost. Both computation and memory consumption grow directly proportionally with the number of ensemble members, during training as well as during inference. This makes them impractical in hardware-constrained settings, and leads to a widening gap as models continue to grow in size, while at the same time applications like mobile robotics, augmented reality and smart sensor networks increase the need for mobile and embedded computing.

To improve efficiency, several researchers have explored ways to mimic deep ensembles without explicitly duplicating the underlying network. Possible strategies include the reuse and recombination of network modules [e.g., Wen et al., 2020, Havasi et al., 2021], injection of noise at inference time [e.g., Gal and Ghahramani, 2016], as well as hybrid variants [e.g., Durasov et al., 2021]. Empirically, these models do speed up training and/or inference, but still exhibit a significant performance gap compared to the na ive, explicit ensemble, both w.r.t. prediction quality and w.r.t. the calibration of the predicted (epistemic) uncertainties.

The lottery ticket hypothesis [Frankle and Carbin, 2018] and other network pruning studies, e.g., Han et al. [2015], Lee et al. [2019], Mallya and Lazebnik [2018], underline that neural networks are heavily over-parameterized. Their parameters are used inefficiently, and they can be pruned significantly without large performance drops. In lifelong learning and multi-task learning it is essential to use the network efficiently, in order to limit computational overhead when introducing new tasks. There are recent works that achieve good performance in these tasks by introducing modulation (respectively, adaptation) strategies. In particular, Li et al. [2018], Takeda et al. [2021] propose an efficient lifelong learning / domain adaptation method for multi-task learning, utilizing single feature-wise linear modulation (FiLM).

Inspired by that line of work, we propose a new, efficient ensemble method, FiLM-Ensemble. Our method adapts feature-wise linear modulation as an alternative way to construct an ensemble for (epistemic) uncertainty estimation. FiLM-Ensemble greatly reduces the computational overhead compared to the naïve ensemble approach, while performing almost on par with it, sometimes even better. In a nutshell, FiLM-Ensemble can be described as an implicit model ensemble in which each individual member is defined via its own set of linear modulation parameters for the activations, whereas all other network parameters are shared among ensemble members – and therefore only need to be stored and trained once. Thanks to this design, our method requires only a very small number of additional parameters on top of the base network, e.g., converting a single ResNet-18 model to an ensemble of 16 models increases the parameter count by 1.3%, compared to an increase by 1500% when setting up a naïve ensemble, see Table 1. We further show that FiLM-Ensemble results in more diverse ensemble members compared to other efficient ensemble methods. For instance, on the Cifar10 benchmark it achieves diversity scores $> 6.9\%$ and up to 9.2% (depending on ensemble size), against 6.8% for a naive ensemble, see Fig. 1. Our contributions can be summarized as follows.

- We propose FiLM-Ensemble, a novel parameter- and time-efficient deep ensemble method.
- FiLM-Ensemble is designed in such a way that it can be readily combined with many popular deep learning models, by simply replacing batch normalisation (batchnorm) layers with conditional batchnorm layers.
- We show that FiLM-Ensemble provides an excellent trade-off between accuracy and calibration performance, improving over existing ensemble methods.

2 FiLM-Ensemble

In [Perez et al., 2018] the linear feature modulation was achieved by varying the affine parameters of the batch normalization layers $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$. For each batchnorm layer n in the network, the parameters

are predicted according to some conditioning input z (for instance, different prediction tasks):

$$\boldsymbol{\gamma}_n = g_n(z) \qquad \boldsymbol{\beta}_n = h_n(z) \tag{1}$$

The size of $\boldsymbol{\gamma}_n$ and $\boldsymbol{\beta}_n$ is $\mathbb{R}^{D_n}$ where D_n is the feature dimension at layer n. These parameters are then used to linearly modulate the activations at the n-th layer $\mathbf{F}_n$:

$$\text{FiLM}(\mathbf{F}_n|\boldsymbol{\gamma}_n, \boldsymbol{\beta}_n) = \boldsymbol{\gamma}_n(z) \circ \mathbf{F}_n + \boldsymbol{\beta}_n(z) \ , \tag{2}$$

with $\circ$ being the Hadamard (element-wise) product taken w.r.t. the feature dimension.

In our scenario we aim to use the affine parameters to instantiate different ensemble members, i.e., the conditioning variable z is simply an index $m \in \{1, ..., M\}$ that identifies the ensemble member. The functions $g_n(z)$ and $h_n(z)$ degenerate to look-up tables, so we can dispose of them and simply learn M different sets of affine parameters for every batchnorm layer $\mathbf{F}_n$:

$$\text{FiLM}(\mathbf{F}_n|\boldsymbol{\gamma}_n^m, \boldsymbol{\beta}_n^m) = \boldsymbol{\gamma}_n^m \circ \mathbf{F}_n + \boldsymbol{\beta}_n^m \ . \tag{3}$$

To achieve ensemble members with diverse parameters, we resort to Xavier initialization, i.e., all $\boldsymbol{\gamma}_n^m$ and $\boldsymbol{\beta}_n^m$ are sampled from a uniform distribution bounded between:

$$\pm \frac{\sqrt{3}}{\sqrt{D_\text{n}}} \rho \ , \tag{4}$$

where D_n is the number of features (channels) in the n-th layer, and ρ is an initialization gain factor. In our setting ρ is a tunable hyperparameter that allows one to control the trade-off between predictive accuracy and calibration of the model, see Section 3.6. In general, increasing ρ leads to more diverse ensemble members and thus favours calibration. Note that as $\rho \to 0$, all ensemble members start from similar initial values for $\boldsymbol{\gamma}_n^m$ and $\boldsymbol{\beta}_n^m$ and the FiLM-Ensemble gradually collapses to a single model.

During training, we feed each input sample $\mathbf{x}$ to every ensemble member and obtain predictions $y_m = f_{\boldsymbol{\theta}, \boldsymbol{\gamma}^m, \boldsymbol{\beta}^m}(\mathbf{x})$, which depend bot on the member-specific parameters $(\boldsymbol{\beta}^m, \boldsymbol{\gamma}^m)$ and on the shared parameters θ. All those parameters are optimized together to minimise the chosen loss function. Here we focus on classification and use a standard cross-entropy (CE) loss.

At inference time the final prediction $\hat{y}$ is obtained by averaging the M predictions of the ensemble:

$$\hat{y} = \frac{1}{M} \sum_{m=1}^{M} y_m \ . \tag{5}$$

2.1 Implementation Details

We implemented 1-dimensional and 2-dimensional FiLM-Ensemble layers, to be used in combination with 1-dimensional and 2-dimensional convolution operations, respectively. The forward passes through different ensemble members can easily be parallelized by replicating both the input tensor and the FiLM parameters $\boldsymbol{\gamma}, \boldsymbol{\beta}$ along the batch dimension and applying the affine transformation Eq. 3 (the same holds for the BatchEnsemble method). In thid way all ensemble members run simultaneously on a single device, thus optimally utilizing modern tensor computing hardware without having to load several instances of the classification network into memory. We have implemented FiLM-Ensemble in PyTorch and release the source code, including implementations of the baselines used in the experiments, together with trained model checkpoints. The source code is submitted as supplementary material.

For all experiments, we optimize the model parameters with standard stochastic gradient descent, with momentum $\mu = 0.9$ and weight decay $\lambda = 0.0005$ for regularization. We train for 200 epochs with batch size 128. The learning rate is initially set to 0.1 and decays according to a cosine annealing schedule [Loshchilov and Hutter, 2017]. The initialisation gain is set to $\rho = 2$ for all experiments, except for genome sequences (see Table 3), where $\rho \in \{2, 8, 16, 32\}$. Please refer to the supplementary material for additional, dataset-specific details.

3 Experiments

In the following, we first empirically examine the ability of FiLM-Ensemble to learn independent ensemble members, and compare it to deep ensembles. Then we analyze the predictions accuracy and

uncertainty calibration of FiLM-Ensemble, as well as its computational cost and memory consumption. We compare its performance against several baseline methods and state-of-the-art alternatives on a variety of different datasets, and using different backbone architectures. All experimental results are averaged over three runs with different random seeds.

We evaluate our proposed method on a diverse set of classification tasks, including popular image classification benchmarks, image-based medical diagnosis, and genome sequence analysis. **CIFAR-10** and **CIFAR-100** [Krizhevsky, 2009] are widely used testbeds for image classification, and deep learning in general. They consist of clean images of objects from 10, respectively 100, different semantic classes. Each of the two datasets contains 60,000 images, of which 10,000 are reserved for testing. The semantic classes are uniformly distributed in the datasets and stratified across the train/test spit. **Retina Glaucoma Detection** [Diaz-Pinto et al., 2019] is a real-world clinical dataset that includes microscopic retina images from 956 patients with the neuropathic disease Glaucoma, and from 1401 subjects with normal (healthy) retinas. Each input sample is a single RGB image; we resize all images to 128×128. **REFUGE 2020** [Orlando et al., 2020] was a challenge at the MICCAI-2020 conference. The task consists again in detecting retinal Glaucoma, in a dataset of 800 microscopic retina images with size 1411×1411 pixels, collected from different clinics. We use this dataset as a test set in conjunction with the model trained on the previous dataset [Diaz-Pinto et al., 2019], to evaluate performance under distributional shift (Table 4). **6mA Identification** [Li et al., 2021], is a 1-dimensional sequential genome dataset. It consists of DNA sequences of rice plants, along with binary labels that indicate whether the sequence is a N6-methyladenine (6mA) site. 6mA is an important DNA modification associated with several biological processes, such as regulating gene transcription, DNA replication and DNA repair [Campbell and Kleckner, 1990, Cheng et al., 2016, Pukkila et al., 1983]. Each sequence consists of 41 nucleotides. As there are 4 different types of nucleotides, each one-hot encoded sequence is represented by a 41×4 matrix. In total there are 269,500 training samples and 38,500 test samples.

We compare FiLM-Ensemble against *(i)* a single model without any ensembling, as the elementary baseline, *(ii)* a naïve deep ensemble [Lakshminarayanan et al., 2017b], and *(iii)* MC-Dropout [Gal and Ghahramani, 2016]. Furthermore, we compare with other state-of-the-art methods, namely: *(iv)* Masksemble [Durasov et al., 2021] which can be seen as an extension of MC-Dropout, *(v)* MIMO (" multi-input multi-output") [Havasi et al., 2021] which defines a multi-head architecture where each head acts as one ensemble member, and *(vi)* BatchEnsemble [Wen et al., 2020] which creates an efficient ensemble by expanding the layer weights using low rank matrices. Arguably, this layer-wise modification of an underlying, common representation is the approach that comes closest to our work.

3.1 Diversity Analysis

Diversity is the key to constructing powerful ensembles: nothing is to be gained from highly correlated ensemble members that return similar outputs for (almost) any input [Zhang and Ma, 2012]. In order to analyze the diversity (respectively, the degree of independence) among members, we compute two distance metrics between the members' predictive distributions. Let f_i and f_j denote two different ensemble members for a classification task. We first measure the *disagreement score* $\mathcal{D}$, defined as the fraction of all s test samples for which the two members return different answers, averaged over all possible pairs of members:

$$\mathcal{D} = \frac{2}{M(M-1)} \sum_{i=1}^{M} \sum_{j=i+1}^{M} \sum_{k=1}^{s} \frac{1}{s} \left[f_i(\mathbf{x}_k) \neq f_j(\mathbf{x}_k) \right] , \tag{6}$$

with $[\cdot]$ being the Iverson bracket. Second, we also measure the Kullback–Leibler (KL) divergence between the two predictive distributions $p(f_i)$ and $p(f_j)$, again averaged over all pairs:

$$KL = \frac{2}{M(M-1)} \sum_{i=1}^{M} \sum_{j=i+1}^{M} \sum_{k=1}^{s} p(f_i(\mathbf{x}_k)) \log \frac{p(f_i(\mathbf{x}_k))}{p(f_j(\mathbf{x}_k))} . \tag{7}$$

For both metrics, higher numbers correspond to higher diversity. See Fig. 1. We observe that the average diversity increases with the number of ensemble members. In both metrics, FiLM-Ensemble achieves higher scores than the naïve, explicit deep ensemble. Meaning that the predictions of FiLM-Ensemble are less correlated than those of an equivalent number of networks trained independently with different random seeds.

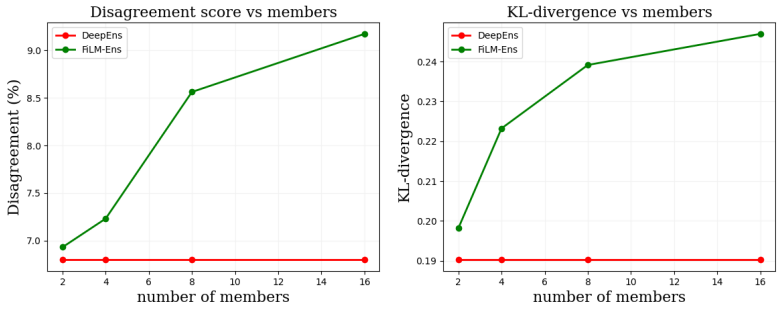

Figure 1: Diversity analysis: CIFAR10/VGG11 experiment. With increasing number of members, FiLM-Ensemble achieves more diverse representations. See Section 3.1.

Table 1: Memory and inference time comparison (CIFAR-10/100 datasets): Number of additional trainable parameters to have 16 ensemble members for different backbones. The run-time shown corresponds to the mean GPU time required to run a forward pass of a batch size of 1 with 16 ensemble members. The bottom table section comprises methods whose forward and backward pass is implemented in parallel over ensemble members. Measurements are done on an NVIDIA GeForce GTX 1080 Ti.

Method	Parameters (↓)		Inference time (ms) (↓)	
Backbone	VGG-11	ResNet-18	VGG-11	ResNet-18
MC-Dropout	0.0%	0.0%	16× 2.5	16×2.4
Deep Ensemble	1500%	1500%	16× 2.3	16× 1.8
Masksemble	0.0%	0.0%	20.8	6.6
MIMO	1.1%	0.9%	2.7	1.9
BatchEnsemble	1.4%	5.2%	2.8	5.2
FiLM-Ensemble	0.9%	1.3%	2.8	5.7

3.2 Computational Cost

We go on to measure the efficiency of our proposed method, both in terms of parameter count and in terms of time needed for a forward pass. Table 1 (left section) reports the numbers of additional numbers of parameters compared to a single network (i.e., without uncertainty calibration). Due to their design, MC-Dropout and Masksemble do not require any additional parameters. Among the remaining methods, FiLM-Ensemble has the lowest (VGG-11) or second-lowest (Resnet18) overhead in terms of parameter count, while performing significantly better, as we will see below. In terms of inference time (Tab. 1, right section), FiLM-Ensemble turns out to be very competitive. Only MIMO with Resnet18 backbone is significantly faster, however this comes at a considerable price in terms of accuracy and calibration, see below.

3.3 CIFAR10 / CIFAR100

We start with widely used standard benchmarks in computer vision, CIFAR10 and CIFAR100. As performance metrics, we plot the test set accuracy and the expected calibration error (ECE) against the ensemble size M in Fig. 2, for all compared methods. In terms of accuracy (left subfigure), FiLM-Ensemble is outperformed only by the explicit deep ensemble, across all tested values of $M \in \{2, 4, 8, 16\}$. These two surpass all other methods by a clear margin. Surprisingly, we observe that BatchEnsemble generally exhibits a negative correlation between ensemble size and test set accuracy, with $M = 4$ performing best among all settings. MIMO shows very poor performance in this experimental setting in terms of accuracy (and also in ECE). We speculate that its shared backbone probably has a tendency to fragment into largely independent ensemble members of low channel depth, which lack the necessary capacity when using comparatively small networks like VGG11 or Resnet18. With regard to ECE (right subfigure), we see all methods improving with growing ensemble size M. FiLM-Ensemble achieves better calibration then the widely used deep

Table 2: Classification performance on CIFAR100, using ResNet-18/34 ($M = 4$) performance comparison. The best score for each metric is printed in **bold**, the second best is underlined.

Backbone	Resnet-18		Resnet-34	
Method	Acc (↑)	ECE (↓)	Acc (↑)	ECE (↓)
Single Network	78.0 ±0.4	0.046 ±0.001	79.3 ±0.2	0.089 ±0.006
Deep Ensemble	**81.6** ±0.3	0.041 ±0.002	**82.0** ±0.1	**0.044** ±0.002
MC-Dropout	75.5 ±0.6	0.064 ±0.003	72.2 ±0.2	0.079 ±0.004
MIMO	48.0 ±2.6	0.083 ±0.017	56.2 ±4.8	0.132 ±0.055
Masksemble	72.5 ±0.5	0.075 ±0.004	70.1 ±1.2	0.067 ±0.004
BatchEnsemble	77.7 ± 0.1	0.052 ± 0.002	78.3 ± 0.1	0.056 ± 0.002
FiLM-Ensemble	79.4 ±0.2	**0.038** ±0.000	80.2 ±0.1	0.045 ±0.001

ensemble and MC-dropout methods, with significant margins at large ensemble sizes. Masksemble and BatchEnsemble achieve very good calibration for sizes $M \in [4 \dots 16]$, but at the cost of significant drops in classification performance. An attractive feature of FiLM-Ensemble is that it offers a simple mechanism for trading off accuracy against calibration, by tuning the gain factor ρ. See Section 3.6.

In Tab. 2 we quantitatively compare all tested methods on the CIFAR-100 dataset, both with a standard Resnet-18 backbone and with a larger Resnet-34. We observe that with both architectures, and in terms of both predictive accuracy and calibration, FiLM-Ensemble always either ranks first, or it ranks second behind the inefficient, explicit deep ensemble. Overall, the experiments with both variants of CIFAR and all three model architectures confirm that our method presents an excellent trade-off between predictive accuracy, uncertainty calibration and computational efficiency.

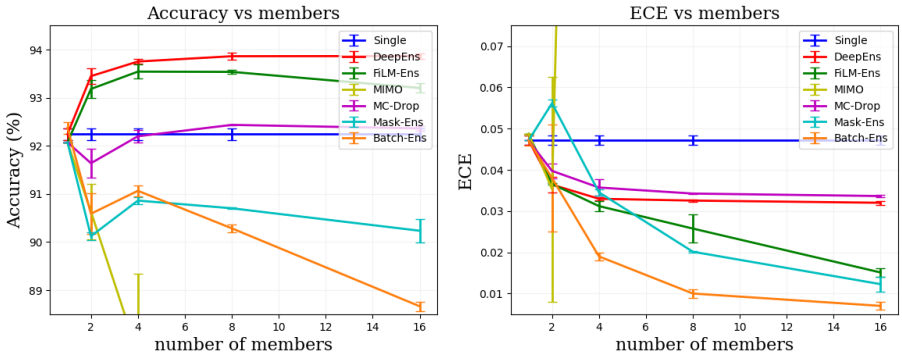

Figure 2: Accuracy and ECE for CIFAR-10, with varying ensemble sizes, using VGG11 as backbone.

3.4 Retinal Glaucoma Detection

Glaucoma is currently the leading reason of irreversible blindness in the world. Detection of glaucomatous structural damages and changes is one of the most challenging tasks in the field of ophthalmology. We evaluate and test, our proposed FiLM-Ensemble and other state-of-the-art algorithms for estimating uncertainty for diagnosis of Glaucoma disease. We show a comparison of the test performance by different uncertainty estimation algorithms in Table 3 and report the performance in terms of accuracy and ECE. Based on Table 3, the best performance achieved by FiLM-Ensemble when our network was composed of 16 members.

3.5 Out-of-Distribution Detection

Domain shift often occurs in medical datasets; thus, detecting out-of-distribution (OOD) samples is an important task in clinical diagnosis. Model uncertainty can be used for OOD detection. We perform OOD detection on REFUGE dataset [Diaz-Pinto et al., 2019] using models trained on Retinal

Table 3: Performance comparison for Retina Glaucoma image dataset. The best score for each metric is printed in **bold**, the second best is underlined.

Method	Accuracy (%) (↑)				ECE (↓)			
# member (M)	2	4	8	16	2	4	8	16
Single	84.4				0.084			
MC-Dropout	67.0	78.4	80.0	82.7	**0.002**	0.046	0.053	0.051
Deep Ensemble	85.6	85.7	86.0	86.8	0.041	0.078	0.091	0.066
MIMO	72.4	69.8	68.9	68.3	0.049	0.061	0.082	0.041
Masksemble	82.7	83.0	80.2	81.7	0.064	0.049	0.021	0.062
BatchEnsemble	84.5	86.5	86.8	87.1	0.035	0.063	0.071	0.066
FiLM-Ensemble	86.3	86.8	86.9	**87.8**	0.062	0.074	0.068	0.055

Table 4: Performance comparison for OOD detection. See Section 3.5. The best score is printed in **bold**, the second best is underlined.

Method	AUROC (%) (↑)			
# member	2	4	8	16
Single	68.42			
MC-Dropout	68.03	69.79	77.94	72.22
Deep Ensemble	76.89	77.91	78.22	78.06
MIMO	57.33	59.49	61.74	60.52
Masksemble	71.22	70.83	72.04	70.95
BatchEnsemble	74.38	72.61	75.44	75.04
FiLM-Ensemble	77.02	77.92	79.43	**79.85**

Glaucoma dataset [Orlando et al., 2020] (in-distribution) and compare AUROC (Area Under the Receiver Operating Characteristic) scores. As shown in Table 4, FiLM-Ensemble can detect OOD test samples much more accurately compared to other methods that are based on prediction uncertainty which suggests that FiLM-Ensemble has better calibration than competitors.

3.6 6mA Identification

Using the 6mA dataset, we showed that FiLM-Ensemble can be readily combined with the existing models that were proposed for 1-dimensional sequential genome data. We used the 1D-CNN architecture whose hyperparameters are already optimized by Li et al. [2021] for the 6mA dataset. FiLM-Ensemble improves the accuracy and calibration of the proposed model. As shown in Fig.3, our method performs on par with deep ensemble and achieves better results than the optimized single model. Besides that, at the expense of a minor accuracy drop, the calibration (ECE) can be improved significantly by increasing the initialization gain factor.

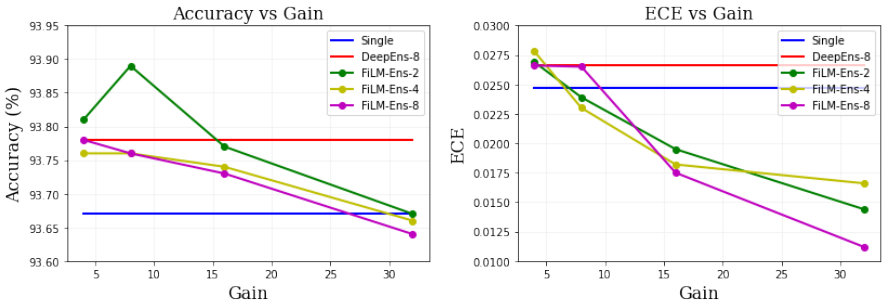

Figure 3: Performance of FiLM-Ensemble for different gain, ρ. See Section 3.6.

4 Related Work

4.1 Epistemic Uncertainty Quantification

A large corpus of related work addresses the estimation of epistemic (model) uncertainty in neural networks. At the heart of such modeling is often the concept of *marginalization instead of optimization*, i.e., integrating out a (possibly uncountably infinite) set of models weighted by their posterior probability, instead of committing to a point estimate of that distribution. A multitude of methods have been proposed to implement approximate Bayesian inference w.r.t. the model weights, given the training data and an appropriate prior, a process that is not analytically tractable in general [Kendall and Gal, 2017].

For instance, methods based on *variational inference* [Graves, 2011, Ranganath et al., 2014, Blundell et al., 2015] seek to learn an approximate posterior distribution which is a member of a simpler family of variational distributions. This variational distribution can often be learned using backpropagation [Blundell et al., 2015] and can then be sampled from or sometimes even used for exact inference. Markov chain Monte Carlo sampling (MCMC) approaches [Neal, 1996, Welling and Teh, 2011, Chen et al., 2014] construct a Markov chain which has the *exact* posterior distribution as its stationary distribution, that can then be employed for sampling. However, in practise, these approaches often fail to sufficiently explore high-dimensional, multi-modal loss landscapes as they are common in deep learning [Gustafsson et al., 2020].

4.2 Ensembles and Sub-networks

In a method referred to as *deep ensembles* [Lakshminarayanan et al., 2017a], a set of M neural network models are randomly and independently initialized, and are subjected to stochastic mini-batch sampling during SGD training. The models generally converge to different minima in the parameter space, and can be considered samples from an approximate posterior [Wilson and Izmailov, 2020, Gustafsson et al., 2020, Izmailov et al., 2021]. Deep ensembles often achieve the best calibration and predictive accuracy [Ovadia et al., 2019, Gustafsson et al., 2020, Ashukha et al., 2020], but suffer from high computational complexity as it requires training, storing, and running inference on several network models.

Many alternative methods have recently been proposed in an attempt of reducing either the computational cost or the storage cost of deep ensembles. Monte Carlo (MC) Dropout [Gal and Ghahramani, 2016] runs multiple forward passes on the dropout layers in order to obtain multiple predictions and obtain an uncertainty estimate. Although, the method requires less computations compared to deep ensembles, it also leads to less accurate uncertainty estimates. Snapshot ensembles [Huang et al., 2017] uses a cyclic learning rate scheduling in order to find multiple local minima, they then store multiple copies of the network to create the ensemble. While this approach reduces the computation cost during training it does not alleviate the storage cost.

BatchEnsemble [Wen et al., 2020] propose to use multiple low rank matrices, which can be stored efficiently, in order to modulate the parameters of a neural network and thus mimic an ensemble of network models. Masksembles [Durasov et al., 2021] aims to improve the performance of MC Dropout by carefully selecting which masks, used to drop certain features, leading to a better uncertainty quantification. Another approach which uses multiple sub-networks is proposed in Havasi et al. [2021]. In this case additional, independent, layers are added at the beginning and at the end of the network in order to obtain multiple prediction with an individual network model. Although such method seems to reduce the computational resource required at training and inference time, the gain is limited to larger backbones, such as use Wide-ResNet (e.g., ResNet28-10), whereas for standard widely used architectures like VGG (e.g., -16), or ResNet (e.g., -34), the approach is not very effective as shown in our experiments.

Although many methods that aim at reducing the computational cost of deep ensembles have been proposed, none of the methods appear to unquestionably outperform the others. As a result the question of how to effectively model uncertainty is still open.

4.3 Feature-wise Linear Modulation

The idea of controlling the batch normalization parameters to modulate a network's function has been explored by many different authors to accomplish different tasks. Conditional batch normalization (CBN) was proposed by de Vries *et al.*, who achieved good performance in VQA experiments by using an MLP to estimate residuals for normalization parameters used in a pre-trained ResNet [De Vries et al., 2017b]. Perez et al. [2018] proposed FiLM also for solving VQA tasks. Strub et al. [2018] modify FiLM to produce normalization parameters in several stages instead of all at once. This formulation is better able to handle longer conditioning information, such as dialogues instead of questions, and achieved excellent results on the GuessWhat?! visual dialogue task [De Vries et al., 2017a].

Such modulation operations have also been used for image style transfer. It has been observed that the statistics of a feature map, which are associated with and can be controlled by the parameters produced by FiLM, are directly related to the style of an image [Huang and Belongie, 2017]. Dumoulin et al. [2017] succeeded in capturing the styles of artistic paintings, as well as combining extracted styles to create new ones, by using conditional instance normalization, which can be seen as a variation of FiLM. Ghiasi et al. [2017] expand on this work by jointly training a style prediction network and a style transfer network, which also operate based on conditional instance normalization. Finally, Brock et al. [2018] used FiLM for natural image synthesis using generative adversarial networks (GANs). They report that this allowed for a reduction in computation and memory costs, as well as a 37% increase in training speed.

Yang et al. [2018] have used FiLM to modulate the layers of a segmentation network to perform video object segmentation. This made it possible to avoid the fine-tuning process that was used by competing methods, which resulted in a speed up of 70 times while achieving similar accuracy. Feature modulation has also been applied for various other tasks. Oreshkin et al. [2018] used FiLM for task-dependent metric scaling, which allowed them to achieve excellent results in few-shot classification. Vuorio et al. [2019] use FiLM for meta-learning via task-aware modulation. The authors note that FiLM outperforms and is more stable than attention-based modulation in this context. Finally, Vinyals et al. [2019] used FiLM in their AlphaStar neural architecture for multi-agent reinforcement learning. To our knowledge, FiLM has so far not been used for (implicit) model ensembling or uncertainty quantification.

5 Conclusion & Future Work

In this paper we present FiLM-Ensemble, a novel implicit deep ensembling method. We achieve high efficiency using a simple yet effective idea – feature-wise linear modulation – which has been shown to be effective in different domains, such as image style transfer, model-agnostic meta-learning, or multi-task learning. Our extensive evaluation shows that FiLM-Ensemble outperforms, or is on par with, state-of-the-art ensemble methods in many different experimental settings. A number of measures designed to enhance implicit ensembles are orthogonal to our approach and could potentially be combined with FiLM-Ensemble to further improve its performance and uncertainty calibration, while minimally increasing the computational costs. It appears straight-forward to add the selection of independent examples for each member during training [as in MIMO, Havasi et al., 2021], temperature scaling [Guo et al., 2017] or data augmentation strategies [such as, e.g., Ramé et al., 2021].

6 Broader Impact

Nowadays, Machine learning has resulted in rapidly growing computational cost and thus energy costs. For instance, training state-of-the-art language models like GPT-3 would amount to at least 1400 MWh or 4.6 million $. Therefore, recently the term *Green AI* has been introduced, referring to AI research that yields novel results while taking into account the computational cost, encouraging a reduction in resources spent. To this extent, we believe that the presented method can foster more efficient ensemble methods and be helpful towards a greener AI.

References

A. Ashukha, A. Lyzhov, D. Molchanov, and D. Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *International Conference on Learning Representations*, 2020.

C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural network. In *International Conference on Machine Learning*, volume 37, 2015.

A. Brock, J. Donahue, and K. Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018.

J. L. Campbell and N. Kleckner. E. coli oric and the dnaa gene promoter are sequestered from dam methyltransferase following the passage of the chromosomal replication fork. *Cell*, 62(5):967–979, 1990.

T. Chen, E. Fox, and C. Guestrin. Stochastic gradient Hamiltonian Monte Carlo. In *International Conference on Machine Learning*, volume 32, 2014.

L. Cheng, J. Sun, W. Xu, L. Dong, Y. Hu, and M. Zhou. Oahg: an integrated resource for annotating human genes with multi-level ontologies. *Scientific reports*, 6(1):1–9, 2016.

H. De Vries, F. Strub, S. Chandar, O. Pietquin, H. Larochelle, and A. Courville. Guesswhat?! visual object discovery through multi-modal dialogue. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017a.

H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, volume 30, 2017b.

A. Der Kiureghian and O. Ditlevsen. Aleatory or epistemic? does it matter? *Structural Safety*, 31(2):105–112, 2009.

A. Diaz-Pinto, A. Colomer, V. Naranjo, S. Morales, Y. Xu, and A. F. Frangi. Retinal image synthesis and semi-supervised learning for glaucoma assessment. *IEEE Transactions on Medical Imaging*, 38(9):2211–2218, 2019.

V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. *ICLR*, 2017.

N. Durasov, T. Bagautdinov, P. Baque, and P. Fua. Masksembles for uncertainty estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13539–13548, 2021.

J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.

G. Ghiasi, H. Lee, M. Kudlur, V. Dumoulin, and J. Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *arXiv preprint arXiv:1705.06830*, 2017.

A. Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, volume 24, 2011.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017.

F. K. Gustafsson, M. Danelljan, and T. B. Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020.

S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

M. Havasi, R. Jenatton, S. Fort, J. Z. Liu, J. Snoek, B. Lakshminarayanan, A. M. Dai, and D. Tran. Training independent subnetworks for robust prediction. In *ICLR*, 2021.

G. Huang, Y. Li, G. Pleiss, Z. Liu, J. E. Hopcroft, and K. Q. Weinberger. Snapshot ensembles: Train 1, get m for free. In *International Conference on Learning Representations*, 2017.

X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision*, 2017.

P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. G. Wilson. What are bayesian neural network posteriors really like? In *International Conference on Machine Learning*, volume 139, 2021.

A. Kendall and Y. Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems*, 2017.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, 2017a.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017b.

N. Lee, T. Ajanthan, and P. H. Torr. Snip: Single-shot network pruning based on connection sensitivity. *ICLR*, 2019.

Y. Li, N. Wang, J. Shi, X. Hou, and J. Liu. Adaptive batch normalization for practical domain adaptation. *Pattern Recognition*, 80:109–117, 2018.

Z. Li, H. Jiang, L. Kong, Y. Chen, K. Lang, X. Fan, L. Zhang, and C. Pian. Deep6ma: A deep learning framework for exploring similar patterns in dna n6-methyladenine sites across different species. *PLoS computational biology*, 17(2):e1008767, 2021.

I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

A. Malinin and M. Gales. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

A. Mallya and S. Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag, Berlin, Heidelberg, 1996.

B. Oreshkin, P. Rodríguez López, and A. Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in Neural Information Processing Systems*, 31, 2018.

J. I. Orlando, H. Fu, J. B. Breda, K. van Keer, D. R. Bathula, A. Diaz-Pinto, R. Fang, P.-A. Heng, J. Kim, J. Lee, et al. Refuge challenge: A unified framework for evaluating automated methods for glaucoma assessment from fundus photographs. *Medical Image Analysis*, 59:101570, 2020.

Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018.

P. J. Pukkila, J. Peterson, G. Herman, P. Modrich, and M. Meselson. Effects of high levels of dna adenine methylation on methyl-directed mismatch repair in escherichia coli. *Genetics*, 104(4):571–582, 1983.

A. Ramé, R. Sun, and M. Cord. Mixmo: Mixing multiple inputs for multiple outputs via deep subnetworks. In *IEEE/CVF International Conference on Computer Vision*, 2021.

R. Ranganath, S. Gerrish, and D. Blei. Black box variational inference. In *International Conference on Artificial Intelligence and Statistics*, 2014.

F. Strub, M. Seurin, E. Perez, H. De Vries, J. Mary, P. Preux, and A. C. Pietquin. Visual reasoning with multi-hop feature modulation. In *European Conference on Computer Vision*, 2018.

M. Takeda, G. Benitez, and K. Yanai. Training of multiple and mixed tasks with a single network using feature modulation. In *International Conference on Pattern Recognition*, pages 719–735. Springer, 2021.

O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

R. Vuorio, S.-H. Sun, H. Hu, and J. J. Lim. Multimodal model-agnostic meta-learning via task-aware modulation. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *International Conference on Machine Learning*, 2011.

Y. Wen, D. Tran, and J. Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020.

A. G. Wilson and P. Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos. Efficient video object segmentation via network modulation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

C. Zhang and Y. Ma. *Ensemble machine learning: methods and applications*. Springer, 2012.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [No]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]