MIMMO: Multi-Input Massive Multi-Output Neural Network - Supplementary Material

Martin Ferianc, Miguel Rodrigues
University College London
{martin.ferianc.19, m.rodrigues}@ucl.ac.uk

A. Derivation of the ELBO Objective

Assume the data $\mathcal{D} = \{x_n, y_n\}_{n=1}^N$ under the model $p(y|x, \mathbf{w})$ where $\mathbf{w}$ are the parameters of the network, x is the input and y is the target. Assume that $\mathcal{D}$ is split into $X = \{x_n\}_{n=1}^N$ and the associated targets for each input $Y = \{y_n\}_{n=1}^N$. Given M members in the MIMMO NN which are independent in the inputs and outputs we can write the sum of the log-likelihood as $\sum_{m=1}^M \log p(Y|X, \mathbf{w}) = \sum_{m=1}^M \sum_{n=1}^N \log p(y_m^n|x_m^n, \mathbf{w})$. This allows us to compute the likelihood of the data for each member separately and then sum them up [11].

Assuming now that we want to induce a prior and a variational distribution over the depth of the members as $p(d_m|\{\gamma_m^i\}_{i=1}^D) = \text{Cat}(d_m|\{\gamma_m^i\}_{i=1}^D)$ independently for each ensemble member m and a variational posterior $q(d_m|\{\theta_m^i\}_{i=1}^D) = \text{Cat}(d_m|\{\theta_m^i\}_{i=1}^D)$, where θ is a vector of learnable parameters for each ensemble member m we bound the log-likelihood of the data by the ELBO objective as below:

$$\begin{aligned}
\mathcal{L}(\mathcal{D}|\mathbf{w}, \theta) &= \log \sum_{i=1}^D p(Y|X, \mathbf{w}, d^i)p(d^i|\gamma^i) \\
&= \log \sum_{i=1}^D p(Y|X, \mathbf{w}, d^i)q(d^i|\theta^i)\frac{p(d^i|\gamma^i)}{q(d^i|\theta^i)} \\
&\geq \mathbb{E}_{q(d^i|\theta^i)}\left[\log \frac{p(Y|X, \mathbf{w}, d^i)p(d^i|\gamma^i)}{q(d^i|\theta^i)}\right] \\
&\geq \mathbb{E}_{q(d^i|\theta^i)}\left[\log p(Y|X, \mathbf{w}, d^i)\right] \\
&- KL(q(d^i|\theta^i)||p(d^i|\gamma^i)) + \text{Constant} \\
&\geq \sum_{m=1}^M \sum_{n=1}^N \sum_{i=1}^D \log p(y_{n,m}^i|x_{n,m}, \mathbf{w}, d_m^i)\theta_m^i \\
&- \alpha \sum_{m=1}^M \sum_{i=1}^D \theta_m^i \log \frac{\theta_m^i}{\gamma_m^i}
\end{aligned}$$

The derivation starts by writing the joint log-likelihood of the data under the model as a sum of individual log-likelihoods for each member of the MIMMO NN. This assumes that the members are independent given the inputs and outputs. Given the categorical prior and a variational posterior over the depth of each member, the variational posterior enters the equation as multiplication with respect to 1. The derivation then applies Jensen's inequality to lower bound the log-likelihood giving the ELBO objective. It consists of two terms: an expected conditional log-likelihood and a KL divergence between the variational posterior and the prior and a constant with respect to the prior distribution. The expected conditional log-likelihood is computed by taking an expectation over the variational posterior for each member each data point and each depth. The KL divergence is computed analytically assuming categorical distributions. By default it is a reverse KL diver-

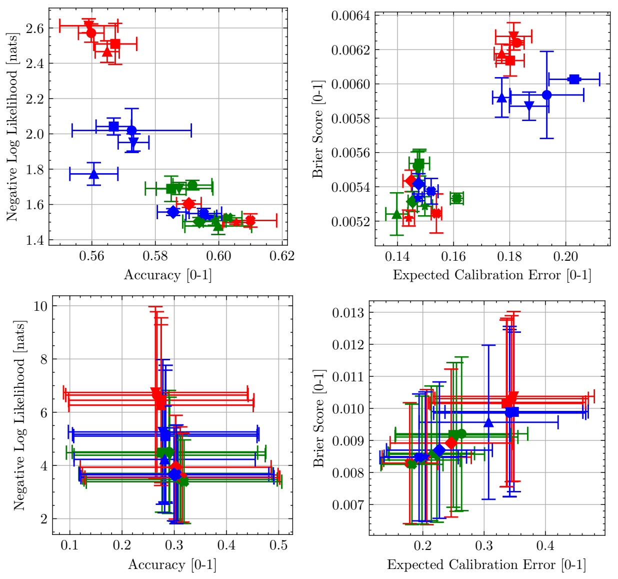

Figure 5. Ablation study on test (upper row) and augmented (lower row) set with varying α. The red points represent $M = 2$, the green points represent $M = 4$ and the blue points represent $M = 8$. The circles represent $\alpha = 0.0$, the squares represent $\alpha = 1.0$, the down triangles represent $\alpha = 100.0$, the up triangles represent $\alpha = 10000.0$, the diamonds represent $\alpha = 45000.0$, the stars represent $\alpha = 100000.0$ and the crosses represent $\alpha = 1000000.0$.

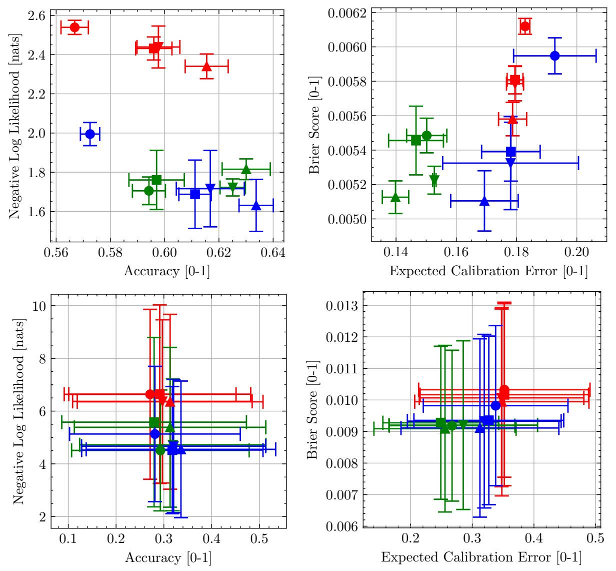

Figure 6. Ablation study on test (upper row) and augmented (lower row) set with a varying number of layers and width. The red points represent $M = 2$, the green points represent $M = 4$ and the blue points represent $M = 8$. The circles represent (1, 2, 3, 1) layers per residual block and 1 width multiplier, the squares represent (2, 4, 6, 2) layers and 1 width multiplier, the down triangles represent (1, 2, 3, 1) layers and 2 width multiplier, and the up triangles represent (2, 4, 6, 2) layers and 2 width multiplier.

gence, but given that KL divergence is not symmetric, we consider experimenting with the forward KL divergence as future work. The α controls the strength of regularization.

B. Experimental Settings

We conducted experiments on the CIFAR-100 dataset with a WideResNet architecture [32] with 4 blocks with (1,2,3,1) layers, (1,2,2,2) strides and (16, 32, 64, 128) channels, expansion 4, groups 1, base width 32 and multiplier 2. The three early-exits for DUN, EE or MIMMO where added exactly after the series of 1, 2 and 3 blocks respectively. No data augmentation was used, or weight regularization and a regular net was able to quickly overfit the data for any experiment. No data augmentation was used such that there is no linkage to the data augmentations during the out of training distribution data tests. Batch repetition was always set to 2, batch size to 64 and we used the Adam optimizer, with a cosine learning rate schedule until the end of training for all experiments. The initial learning rate was set to 0.005. We run for 100 epochs. Input repetition probability was set to 0.25. We warm-started the networks with 5 epochs where the depth parameters were fixed to the prior. All the methods used the same main hyperparameters and we note the differences below. We hand-tuned the main defining parameters to the best of our ability on the 10% of the training data as validation set. **Monte Carlo Dropout**: For Monte Carlo Dropout we used 10 samples, dropout inserted before every convolution or linear layer with a dropout probability of 0.15. **DataMUX**: We used a 3×3 convolution with 1 padding and 1 stride, followed by ReLU activation. **DUN**: We set $\alpha = 1$ in objective (1) and of course $M = 1$. We also used 5 warm-starting epochs during which the depth parameter was fixed to the prior for fairness. **EE**: We set the alpha parameter to weight the early-exit losses to 1.0, gamma to 0.3 and beta to 0.0. **MixMo**: We set initial probability to 0.5, $r = 3.0$ and $\alpha = 2.0$ for the Dirichlet distribution. The input repetition probability was set to 0.0.

All tasks were classifications with a cross-entropy loss and softmax activation on the output layer. We tested out-of-distribution calibration performance on a set of 5 different augmentations with 5 varying levels of severity: random horizontal or vertical translation [0.1, 0.2, 0.3, 0.4, 0.5], rotation [0, 45, 90, 135, 180], brightness [0.9, 0.7, 0.5, 0.3, 0.1], contrast [0.9, 0.7, 0.5, 0.3, 0.1] and amble [0.9, 0.8, 0.7, 0.6, 0.5]. To summarize the performance we compute the mean and the standard deviation across all augmentations and levels giving two numbers for each method. To benchmark the algorithmic performance, we looked at the negative log-likelihood, expected calibration error [10] with 15 bins, Brier score and accuracy. From the hardware perspective, we looked at the number of FLOPs and parameters.

B.1. Feature Reuse Between Members

We adopt the method of [26] to investigate feature reuse between different members depending on M. Namely we probe the weights in the first convolution and the output classifier and we compute the $L1$ norm of the features that are associated with a given member and normalize it by the maximum value of the $L1$ norm for a given member. The overlap between the F features of the M members is computed as the average overlap $= \frac{1}{M}\sum_{m=1}^{M}\frac{F^f_{m,min}}{F^f_{m,max}}$, where F^f_{min} is the minimum $L1$ norm of the features of the f^{th} feature map and F^f_{max} is the maximum $L1$ norm of the features of the f^{th} feature map across all M members.

C. Measuring the Effect of α, M, Depth, Width

C.1. Measuring the Effect of α, M

We observe the algorithmic performance of changing α and M in detail in Fig. 5. Increasing α forces the network to put more weight on all the exits. As illustrated in the Figure, the performance significantly improves by changing α logarithmically for every M. In particular, all the algorithmic metrics improve as α increases, namely the negative log-likelihood, expected calibration error, Brier score and accuracy. In terms of the augmentations, the performance is also improved through changing α. Note that, the strength

Method	Accuracy [0-1]	NLL [nats]	ECE [0-1]	Brier Score [0-1, $\times 10^{-3}$]	FLOPs [M]	Params [M]
Base	0.52 ± 0.01	3.94 ± 0.08	0.36 ± 0.01	7.93 ± 0.11	**109.85**	**1.76**
Ensemble	0.56 ± 0.00	2.87 ± 0.03	0.20 ± 0.00	6.33 ± 0.04	219.7	3.51
MC Dropout	0.54 ± 0.00	1.92 ± 0.01	0.15 ± 0.00	6.02 ± 0.02	1104.6	1.76
DUN	0.55 ± 0.00	3.74 ± 0.04	0.35 ± 0.00	7.55 ± 0.02	172.17	1.99
EE	0.56 ± 0.01	1.74 ± 0.01	0.15 ± 0.00	5.77 ± 0.06	110.46	2.24
MIMO	0.47 ± 0.00	3.53 ± 0.16	0.22 ± 0.00	7.51 ± 0.08	110.35	1.81
DataMux	0.44 ± 0.01	3.64 ± 0.10	0.23 ± 0.00	7.77 ± 0.04	110.53	1.81
MixMo	0.53 ± 0.01	2.06 ± 0.10	0.17 ± 0.00	6.25 ± 0.15	110.35	1.81
MIMMO γ	$\mathbf{0.62 \pm 0.01}$	$\mathbf{1.50 \pm 0.03}$	0.16 ± 0.00	$\mathbf{5.20 \pm 0.08}$	172.82	2.04
MIMMO	0.61 ± 0.01	$\mathbf{1.50 \pm 0.01}$	$\mathbf{0.14 \pm 0.00}$	5.22 ± 0.05	172.82	2.04
Base	0.24 ± 0.16	8.50 ± 3.34	0.58 ± 0.14	12.70 ± 2.92	**109.85**	**1.76**
Ensemble	0.27 ± 0.18	7.05 ± 3.28	0.37 ± 0.14	10.53 ± 2.76	219.7	3.51
MC Dropout	0.23 ± 0.17	4.75 ± 2.06	0.24 ± 0.09	9.47 ± 1.96	1104.6	1.76
DUN	0.28 ± 0.17	7.85 ± 3.20	0.55 ± 0.14	12.10 ± 2.95	172.17	1.99
EE	0.29 ± 0.17	4.60 ± 2.60	0.24 ± 0.10	9.07 ± 2.19	110.46	2.24
MIMO	0.21 ± 0.15	7.85 ± 3.20	0.39 ± 0.13	11.26 ± 2.38	110.35	1.81
DataMux	0.20 ± 0.14	7.67 ± 3.00	0.38 ± 0.11	11.20 ± 2.10	110.53	1.81
MixMo	0.27 ± 0.17	4.67 ± 2.24	0.28 ± 0.11	9.51 ± 2.26	110.35	1.81
MIMMO γ	$\mathbf{0.32 \pm 0.19}$	$\mathbf{3.57 \pm 1.75}$	$\mathbf{0.18 \pm 0.05}$	$\mathbf{8.28 \pm 1.94}$	172.82	2.04
MIMMO	0.31 ± 0.19	3.64 ± 1.81	0.21 ± 0.07	8.48 ± 2.11	172.82	2.04

Table 1. Results on CIFAR-100 for $M = 2$. The results separated by the horizontal bar signify the results with respect to the augmented test dataset with a range of 5 data augmentations across 5 different levels of intensity. The base is a pointwise single network, Ensemble [16], MC Dropout [8], DUN [1], EE [22], MIMO [11], DataMux [20], MIMMO γ does not optimize θ and MIMMO.

of the regularisation is also affected by the overall size of the training set N, meaning that for a smaller training set, the regularisation will be stronger by default without the need to increase α. It is important to add that as seen in the Figures, different α values are suited for different M. In case of $\alpha = 1000000$, which was the maximum value in our experiments, the depth probabilities stayed around 0.25 and this was not optimal for any M. In summary, α is a hyperparameter that is worth tuning for the best performance.

C.2. Measuring the Effect of Depth, Width and M

Next, we examine the impact on the performance when we alter the depth and width of the network, while keeping $\alpha = 1$. We changed the depth by having (1, 2, 3, 1) or (2, 4, 6, 2) layers per residual block and we changed the width by having (16, 32, 64, 128) channels or (32, 64, 128, 256) channels per residual block and report the results in Fig. 6. We observe that the performance changes rather significantly as the network gets deeper and wider, especially in the case of $M = 2, 8$. The results indicate that the width multiplier has a rather same effect as the number of layers. If the number of layers is increased and width is doubled we observe the best algorithmic performance. As seen from the augmentation results, the changes in the performance when changing the depth and width are significant especially if the network is both widened and deepened.

D. Detailed Results

In this Section, we present the numerical results of the experiments shown in Fig. 4 for $M = 2$ in Tab. 1, for $M = 4$ in Tab. 2 and for $M = 8$ in Tab. 3. The results separated by the horizontal bar in the Tables signify the results with respect to the augmented test dataset. The results in bold represent the best results for each metric.

MIMMO, optimized or unoptimized, is the best performing method in terms of algorithmic metrics with a negligible hardware cost in comparison to the ensemble. This is observed not only on the test dataset but also the augmented test dataset across all M. Only when $M = 8$ and the ensemble is used, the ensemble outperforms MIMMO in terms of the algorithmic metrics, but MIMMO is able to remain relatively close, especially in terms of calibration or even better than the ensemble when considering negative log-likelihood.

Nevertheless, as it can be seen when comparing MIMMO and MIMMO γ, changing the α is not enough in terms of hyperparameter optimization to achieve the best results. We hypothesize that this effect can be for two reasons: (i) overfitting to the training set and (ii) the priors for the depth probabilities should be specified per-member for different M.

First, we have observed that even with high α values,

Method	Accuracy [0-1]	NLL [nats]	ECE [0-1]	Brier Score [0-1, $\times 10^{-3}$]	FLOPs [M]	Params [M]
Base	0.52 ± 0.01	3.94 ± 0.08	0.36 ± 0.01	7.93 ± 0.11	**109.85**	**1.76**
Ensemble	0.60 ± 0.00	2.22 ± 0.02	0.15 ± 0.00	5.53 ± 0.03	439.4	7.02
MC Dropout	0.54 ± 0.00	1.92 ± 0.01	0.15 ± 0.00	6.02 ± 0.02	1104.6	1.76
DUN	0.55 ± 0.00	3.74 ± 0.04	0.35 ± 0.00	7.55 ± 0.02	172.17	1.99
EE	0.56 ± 0.01	1.74 ± 0.01	0.15 ± 0.00	5.77 ± 0.06	110.46	2.24
MIMO	0.48 ± 0.02	2.90 ± 0.15	0.27 ± 0.00	7.49 ± 0.23	111.33	1.91
DataMux	0.47 ± 0.00	3.06 ± 0.07	0.28 ± 0.01	7.72 ± 0.07	111.70	1.91
MixMo	0.45 ± 0.00	2.25 ± 0.10	0.17 ± 0.02	7.16 ± 0.19	111.33	1.91
MIMMO γ	$\mathbf{0.60 \pm 0.00}$	1.54 ± 0.02	0.16 ± 0.00	6.02 ± 0.02	174.11	2.15
MIMMO	$\mathbf{0.60 \pm 0.01}$	$\mathbf{1.48 \pm 0.05}$	$\mathbf{0.14 \pm 0.00}$	$\mathbf{5.24 \pm 0.12}$	174.11	2.15
Base	0.24 ± 0.16	8.50 ± 3.34	0.58 ± 0.14	12.70 ± 2.92	**109.85**	**1.76**
Ensemble	0.29 ± 0.19	6.03 ± 3.09	0.28 ± 0.13	9.47 ± 2.67	439.4	7.02
MC Dropout	0.23 ± 0.17	4.75 ± 2.06	0.24 ± 0.09	9.47 ± 1.96	1104.6	1.76
DUN	0.28 ± 0.17	7.85 ± 3.20	0.55 ± 0.14	12.10 ± 2.95	172.17	1.99
EE	0.29 ± 0.17	4.60 ± 2.60	0.24 ± 0.10	9.07 ± 2.19	110.46	2.24
MIMO	0.23 ± 0.15	6.27 ± 2.62	0.42 ± 0.11	11.10 ± 2.27	111.33	1.91
DataMux	0.22 ± 0.14	6.44 ± 2.61	0.44 ± 0.12	11.38 ± 2.34	111.70	1.91
MixMo	0.22 ± 0.14	3.74 ± 1.11	$\mathbf{0.17 \pm 0.05}$	9.11 ± 1.22	111.33	1.91
MIMMO γ	$\mathbf{0.31 \pm 0.19}$	$\mathbf{3.45 \pm 1.60}$	0.18 ± 0.05	$\mathbf{8.31 \pm 1.87}$	174.11	2.15
MIMMO	0.30 ± 0.18	3.70 ± 1.80	0.21 ± 0.07	8.60 ± 2.10	174.11	2.15

Table 2. Results on CIFAR-100 for $M = 4$. The results separated by the horizontal bar signify the results with respect to the augmented test dataset with a range of 5 data augmentations across 5 different levels of intensity. The base is a pointwise single network, Ensemble [16], MC Dropout [8], DUN [1], EE [22], MIMO [11], DataMux [20], MIMMO γ does not optimize θ and MIMMO.

the depth probabilities appear to overfit to the training set and the performance on the test set is not optimal. As M increases, the overfitting seems less pronounced and optimizing the θ parameters leads to better performance on the test set and augmentations across all metrics.

Second, as observed in Fig. 2, different members could choose completely different depths across different levels and this effect was more pronounced for larger M. For $M = 2$ we observed that the learnt depth distributions were very similar for both members. This leads us to believe that the priors could be set in such a way that each member should choose a different depth, but focus on the different levels of the network, for example, member 1 will have the highest weight at depth 3 and member 2 will have the highest weight at depth 4.

In summary for a practical consideration, due to increasing M the members in the network have smaller per-member capacity and it is more difficult to overfit to the training set and also important to tune the depth probabilities per-member. However, if M is smaller, the practitioner could regularize the output layer via the early exits without the need to optimize the θ parameters to achieve good performance. In addition to different tasks and NN architectures we aim to investigate the depth learning process depending on M and the priors in future work.

Method	Accuracy [0-1]	NLL [nats]	ECE [0-1]	Brier Score [0-1, $\times 10^{-3}$]	FLOPs [M]	Params [M]
Base	0.52 ± 0.01	3.94 ± 0.08	0.36 ± 0.01	7.93 ± 0.11	**109.85**	**1.76**
Ensemble	$\mathbf{0.62 \pm 0.00}$	1.82 ± 0.02	$\mathbf{0.14 \pm 0.00}$	$\mathbf{5.06 \pm 0.02}$	878.8	14.05
MC Dropout	0.54 ± 0.00	1.92 ± 0.01	0.15 ± 0.00	6.02 ± 0.02	1104.6	1.76
DUN	0.55 ± 0.00	3.74 ± 0.04	0.35 ± 0.00	7.55 ± 0.02	172.17	1.99
EE	0.56 ± 0.01	1.74 ± 0.01	0.15 ± 0.00	5.77 ± 0.06	110.46	2.24
MIMO	0.47 ± 0.02	3.38 ± 0.16	0.33 ± 0.01	8.03 ± 0.26	113.31	2.12
DataMux	0.42 ± 0.00	4.12 ± 0.07	0.38 ± 0.00	8.98 ± 0.06	114.05	2.12
MixMo	0.29 ± 0.11	3.21 ± 0.41	0.19 ± 0.07	8.88 ± 0.47	113.31	2.12
MIMMO γ	0.60 ± 0.00	1.56 ± 0.01	0.15 ± 0.00	5.39 ± 0.03	176.70	2.35
MIMMO	0.60 ± 0.00	$\mathbf{1.53 \pm 0.01}$	0.15 ± 0.00	5.33 ± 0.01	176.70	2.35
Base	0.24 ± 0.16	8.50 ± 3.34	0.58 ± 0.14	12.70 ± 2.92	**109.85**	**1.76**
Ensemble	$\mathbf{0.31 \pm 0.20}$	5.24 ± 2.89	0.24 ± 0.13	8.90 ± 2.67	878.8	14.05
MC Dropout	0.23 ± 0.17	4.75 ± 2.06	0.24 ± 0.09	9.47 ± 1.96	1104.6	1.76
DUN	0.28 ± 0.17	7.85 ± 3.20	0.55 ± 0.14	12.10 ± 2.95	172.17	1.99
EE	0.29 ± 0.17	4.60 ± 2.60	0.24 ± 0.10	9.07 ± 2.19	110.46	2.24
MIMO	0.22 ± 0.15	7.08 ± 2.76	0.49 ± 0.11	11.87 ± 2.35	113.31	2.12
DataMux	0.19 ± 0.13	7.95 ± 2.84	0.54 ± 0.11	12.57 ± 2.19	114.05	2.12
MixMo	0.14 ± 0.09	3.96 ± 0.47	$\mathbf{0.10 \pm 0.05}$	9.52 ± 0.34	113.31	2.12
MIMMO γ	0.30 ± 0.19	3.62 ± 1.74	0.19 ± 0.06	$\mathbf{8.47 \pm 1.95}$	176.70	2.35
MIMMO	$\mathbf{0.31 \pm 0.18}$	$\mathbf{3.56 \pm 1.69}$	0.20 ± 0.07	8.49 ± 2.02	176.70	2.35

Table 3. Results on CIFAR-100 for $M = 8$. The results separated by the horizontal bar signify the results with respect to the augmented test dataset with a range of 5 data augmentations across 5 different levels of intensity. The base is a pointwise single network, Ensemble [16], MC Dropout [8], DUN [1], EE [22], MIMO [11], DataMux [20], MIMMO γ does not optimize θ and MIMMO.