# MIMMO: Multi-Input Massive Multi-Output Neural Network

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

## Abstract

*Neural networks (NNs) have achieved superhuman accuracy in multiple tasks, but NNs predictions' certainty is often debatable, especially if confronted with out of training distribution data. Averaging predictions of an ensemble of NNs can recalibrate the certainty of the predictions, but an ensemble is computationally expensive to deploy in practice. Recently, a new hardware-efficient multi-input multi-output (MIMO) NN was proposed to fit an ensemble of independent NNs into a single NN. In this work, we propose the addition of early-exits to the MIMO architecture with inferred depth-wise weightings to produce multiple predictions for the same input, giving a more diverse ensemble. We denote this combination as MIMMO: a multi-input, massive multi-output NN and we show that it can achieve better accuracy and calibration compared to the MIMO NN, simultaneously fit more NNs and be similarly hardware efficient as MIMO or the early-exit ensemble.*

## 1. Introduction

Neural networks (NNs) are powerful models that can outperform other heuristic-based methods in many domains, such as computer vision [4] or language processing [28]. However, NNs must also provide reliable estimates of their confidence in their predictions [10, 21]. This is essential, especially in safety-critical applications such as autonomous driving [2]. One way to obtain a calibrated predictor is to use an ensemble of NNs that are trained independently and average their outputs [16]. Ensembling $M$ NNs produces a calibrated model that closely reflects its confidence in predictions in its accuracy [21]. However, ensembling is not feasible for tasks requiring efficient computation, such as autonomous driving. This is because ensembling needs to run and train $M$ NNs.

Recently, multi-input multi-output (MIMO) NN [11] has been proposed to compress an ensemble of $M$ independent NNs into a single net. This method relies on processing $M$ inputs and outputs that are admitted and extracted by the network simultaneously, but its accuracy or calibration qualities decline when $M > 1$. Nonetheless, a MIMO architecture is superior in hardware efficiency in comparison to an ensemble of $M$ structurally-independent NNs [16].

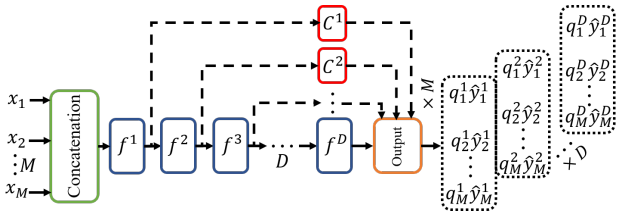


Figure 1. The *network* processes $M$ inputs $\mathbf{x} = \{x_m\}_{m=1}^{M}$ in parallel through their *concatenation* e.g., in the channel dimension for images. Given a network that can be split into $D$ *Blocks* $\{f^i(\cdot)\}_{i=1}^{D}$, we add $D-1$ branches at the output of every hidden layer, except the last. The selected hidden block's features are first processed through a *Connector* $\{C^i(\cdot)\}_{i=1}^{D-1}$ and then fed right into the *Output* to obtain predictions $\{\hat{y}_m^i\}_{i=1,m=1}^{D,M}$. The predictions are weighed through learnt variational distribution $\{\hat{q}_m^i\}_{i=1,m=1}^{D,M}$.

In this paper, we aim to tackle MIMO's algorithmic performance decline when $M > 1$, by investigating a combination of MIMO [11] with an early-exit ensemble, created by adding additional outputs at different depths [1, 22]. The resultant *MIMMO*: a multi-input, massive multi-output NN is visualized in Fig. 1. The model processes $M$ inputs in parallel and produces $M \times D$ outputs. The early hidden representations are processed through resource-conservative *Connectors* and then fed right into the *Output*. The weighting of output for each depth and member is learnt through Bayesian inference [1] via a variational distribution $q(\theta) \in R^{M \times D}$ where $\theta$ are the learnable parameters, independent for each member in $M$. By learning the weighting of the output per-member and per-depth, we wish to maximise the MIMO's ensemble capacity to fit more NNs in parallel.

In the early stage of this work, we conduct ablations. We also benchmark the algorithmic performance and hardware efficiency on residual convolutional NNs (CNNs) in classification on CIFAR-100. Our results indicate that the proposed method achieves improved accuracy and negative log-likelihood with reasonable hardware overhead.

## 2. Related Work

NNs can achieve high accuracy on various tasks, but they often suffer from overconfidence. An overconfident NN makes false predictions with high certainty, which is dangerous in safety-critical applications such as autonomous driving [9]. Despite different sophisticated attempts [3, 8, 9, 19, 21], ensembles maintain the state-of-the-art in confidence calibration, without requiring any particular assumptions about the data or the task [16]. Nevertheless, ensembles pose restrictions both during training and deployment due to requiring to train and run $M$ NNs in parallel.

Recently, a novel method [11] has been proposed to compress multiple independent NNs into a single one. This method is based on a simple yet elegant idea: train a network with multiple inputs and outputs (MIMO) instead of training multiple NNs as in [16]. MIMO can be seen as a collection of single networks that leverage the overparameterisation of the NN [18] to fit multiple networks in the same architecture [11]. The MIMO NN was able to achieve supreme or on-pair confidence calibration and accuracy over other related efficient ensemble approaches such as *Rank-1 Bayesian NNs*, *BatchEnsemble*, and *Hyper Batch Ensembles* [6, 30, 31]. Nevertheless, MIMO suffers from a algorithmic performance decay as $M > 1$.

The work in [24] suggests a related MIMO architecture, that trains a main network that takes multiple inputs and produces a feature representation through regularising the net with the information bottleneck [27]. However, the applicability of [24] to $M > 2$ is unclear along with its confidence calibration. The authors of [20] have demonstrated the advantages of per-member non-linear encoding and decoding the inputs and outputs of a MIMO network. Nevertheless, their method was aimed primarily towards vision transformers [5] and they were not able to provide substantial gains on CNNs without significant hardware overhead. In [23, 25, 26], the authors have proposed a data augmentation method, however, its performance diminished when $M > 2$ and the method is strictly limited to vision inputs or transformers. [17] uses group convolutions to separate individual ensemble members during training and evaluation.

Additionally, [1,22] introduce a single input multi-output method which adds early-exits at different depths of the network. In practice, this means that additional output heads are added and connected to the hidden representations of the network at different depths. The method of [1] is formulated as a whole as a Bayesian NN in which the distribution over depth and branches' outputs is inferred.

To the best of our knowledge, [29] is the closest work to ours that adds auxiliary heads at different depths of the MIMO network in continual learning to avoid forgetting [7]. Our work departs from [29] in three ways: (i) each head is supervised by a different order of labels for the same task, whereas in our work the labels are not reshuffled and there is only a single task; (ii) the authors use different output layers for different stages of the net, which results in hidden early-exit representations that are not used in the final output and in parameter increase due to additional classifier heads; (iii) in our work, we learn the weighting of the early-exit for each depth and member.

In contrast to the aforementioned methods, our approach adds auxiliary exits at different depths $D$ of the MIMO, without additional parameter-demanding output layers. The hidden representations are processed through a single output layer and each depth-wise output is weighted through a variational distribution learnt for each member in $M$ giving in total $M \times D$ outputs instead of $M$ for $M$ inputs.

## 3. Method

### 3.1. Multi Input Multi Output Ensemble

Let $\mathcal{D} = \{x^n, y^n\}_{n=1}^N$ be the training set of $N$ input-output pairs. MIMO [11] is a NN configuration that is trained by taking $M$ different inputs $\mathbf{x} = \{x_m\}_{m=1}^M$ and targets $\mathbf{y} = \{y_m\}_{m=1}^M$, uniformly and iid sampled from $\mathcal{D}$, and returns $M$ outputs $\hat{\mathbf{y}} = \{\hat{y}_m\}_{m=1}^M$. Bold notation is used for multi-input and output symbols. For example, in a computer vision classification task, the input is formed by concatenating the $M$ inputs along the images channels' where if the channel size is 3 and $M = 3$ the input convolution has 9 input channels instead of 3. The output of the network is then $M \times \mathcal{Y}$ e.g. 30 for MNIST classification with $\mathcal{Y} = 10, M = 3$. MIMO can be extracted from Fig. 1 by omitting the dashed arrows and their associated blocks and setting $D = 1$. During training, the network is trained using different reshuffled sets of $M$ examples formed by the training set. During the evaluation, the model is fed with $M$ repeated inputs e.g. $x_i \sim \mathcal{D}$, $\mathbf{x} = \{x_i\}_{m=1}^M$ and the outputs are averaged across the $M$ repetitions as $\hat{y} = \frac{1}{M}\sum_{m=1}^M \hat{y}_m$.

### 3.2. Single Input Early Exit Ensemble

Given a network $f(\cdot)$, it can be split between $D$ different hidden layers or blocks $\{f^i(\cdot)\}_{i=1}^D$; $f(\cdot) = f^1(\cdot) \circ f^2(\cdot) \circ \cdots \circ f^D(\cdot) \circ h(\cdot)$, where $h(\cdot)$ is the output layer. An input $x$ is fed to the network and it is sequentially processed by each block $f^i(\cdot)$, where the output of the $i^{\text{th}}$ layer is fed to the next layer $f^{i+1}(\cdot)$ until the $D^{\text{th}}$ layer which is fed to the output layer $h(\cdot)$ to produce a single output $\hat{y}$. A prime example of this notion is the ResNet [12] architecture, where the network consists of sets of residual blocks $f^i(\cdot)$. The work in [1] proposed an early exit ensemble by adding trainable connector layers $\{C^i(\cdot)\}_{i=1}^{D-1}$ for $\{f^i(\cdot)\}_{i=1}^{D-1}$ but using the same output layer $h(\cdot)$ for all $C^i(\cdot)$. In particular, if we assume a learnable categorical distribution over the contribution of different depths $q(d|\theta)$, [1] have shown that the depth distribution is learnable through stochastic variational inference [13] and naturally regularised, given a categorical

prior $p(d|\theta)$. [1] can be recovered from Fig. 1 by setting $M = 1$. During the evaluation, the model is fed with some input $x$ and the outputs are weighted with the learnt variational posterior over the network depth across the $D$ repetitions as $\hat{y} = \sum_{i=1}^{D} \hat{y}^i \theta^i$. Implementation-wise, the $\theta$ are obtained as a softmax over its logits.

### 3.3. Multi Input Massive Multi Output Ensemble

In this work, we argue that, akin to [1], the depth of the resultant pathway for an ensemble member should be learnable. However, and in contrast, we scale it to $M$ simultaneously learnable ensemble members producing in the end $D$ outputs for each member in $M$ instead of 1. Our goal is to obtain a more diverse ensemble of NNs as well as provide an additional supervision and a regularisation signal during training [11]. Our *MIMMO* is visualized in Fig. 1.

We assume a network $f(\cdot)$ that is split between $D$ different hidden layers or blocks $\{f^i(\cdot)\}_{i=1}^{D}$; connectors $\{C^i(\cdot)\}_{i=1}^{D-1}$ and an output layer $h(\cdot)$; $f(\cdot) = f^1(\cdot) \circ f^2(\cdot) \circ \cdots \circ f^D(\cdot) \circ h(\cdot)$. Instead of processing a single input, we assume that the network is fed with $M$ different inputs $\mathbf{x} = \{x_m\}_{m=1}^{M}$. Hence the output $h(f_D(\cdot))$ by default produces $M$ outputs $\hat{\mathbf{y}}^D = \{\hat{y}_m^D\}_{m=1}^{M}$, since the last hidden layer is connected to the output by default. Additionally, we assume that the hidden features $\{f^i(\cdot)\}_{i=1}^{D-1}$ are also fed to the output layer $h(\cdot)$, through the connectors $\{C^i(\cdot)\}_{i=1}^{D-1}$, producing $M \times D$ outputs $\hat{\mathbf{y}} = \{\hat{y}_m^i\}_{i=1,m=1}^{D,M} = \text{concat}(\{(h(C^i(f^i(\cdot))))\}_{i=1}^{D-1}, \hat{\mathbf{y}}^D)$.

To learn the depth weighting of the outputs we assume a categorical prior $p(d_m|\{\gamma_m^i\}_{i=1}^{D}) = \text{Cat}(d_m|\{\gamma_m^i\}_{i=1}^{D})$ independently for each ensemble member $m$. Similarly, we assume a variational posterior $q(d_m|\{\theta_m^i\}_{i=1}^{D}) = \text{Cat}(d_m|\{\theta_m^i\}_{i=1}^{D})$, where $\theta$ is a vector of learnable weights for each ensemble member $m$. Assuming independence between the ensemble members and sampling of the input, the evidence lower bound (ELBO) can be written as:

$$\mathcal{L}(\mathcal{D}|\mathbf{w}, \theta) \geq \sum_{m=1}^{M} \frac{N}{B} \sum_{b=1}^{B} \sum_{i=1}^{D} \log p(y_{b,m}^i | x_{b,m}, \mathbf{w}, d_m^i) \theta_m^i - \alpha \sum_{m=1}^{M} \sum_{i=1}^{D} \theta_m^i \log \frac{\theta_m^i}{\gamma_m^i} \quad (1)$$

where $\mathbf{w}$ are the network parameters, $N$ is the size of the training set, $B$ is the batch size, $y_{b,m}^i$ is the target for the $m^{\text{th}}$ input in the $b^{\text{th}}$ batch at depth $i$, $x_{b,m}$ is the corresponding input, $\alpha$ is a hyperparameter that controls the strength of the regularising KL divergence term, and $\gamma_m^i$ are fixed hyperparameters that define the prior distribution. As shown in Fig. 2, we noticed that it is necessary to add the $\alpha$ hyperparameter to the ELBO, otherwise, the distribution often quickly collapses to a single mode, i.e. $q(d_m^D|\theta_m) = 1$ for all $m$. Additionally, we introduce $E = 5$ warm-starting

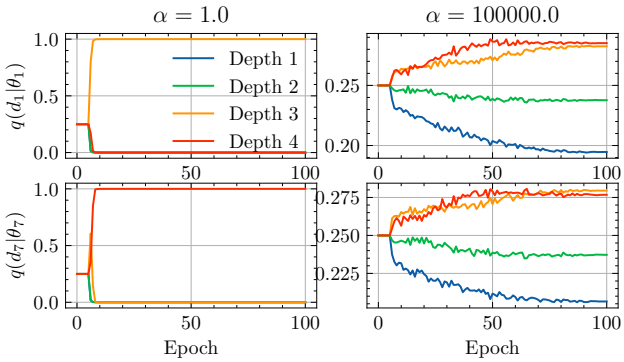


Figure 2. Evolution of probability distributions $q(d_m|\theta_m)$ during training for top left $\alpha = 1.0, m = 1$, top right $\alpha = 100000.0, m = 1$, bottom left $\alpha = 1.0, m = 7$, bottom right $\alpha = 100000.0, m = 7$ for $M = 8$ and $D = 4$.

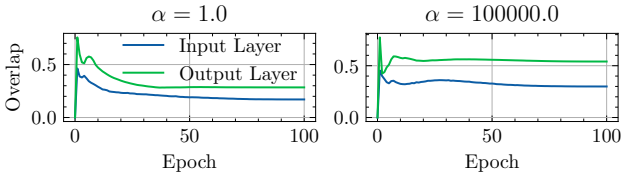


Figure 3. Input and output layer feature overlap during training for left $\alpha = 1.0$, and right $\alpha = 100000.0$ for $M = 8$ and $D = 4$.

epochs where the $\theta$ are not optimised which enables the network to learn some useful features before committing to a specific depth. We elaborate on Eq. (1) in the Appendix A.

The objective in Eq. (1) can be interpreted as maximizing the expected log-likelihood of each ensemble member under its own variational posterior over depth while minimizing the KL divergence between the variational posterior and the prior. This encourages each ensemble member to learn a different depth distribution that best fits the net's limited capacity. During evaluation, we use a deterministic approximation of Eq. (1) by computing $\hat{y} = \frac{1}{M} \sum_{m=1}^{M} \sum_{i=1}^{D} \hat{y}_m^i \theta_m^i$ while feeding the network with the same input $x$ for all $M$ ensemble members. In the case of a CNN, we adopt the reshaping connector layer $C(\cdot)$ with stride 1 from [1], nevertheless, in the case of a fully linear network, we could use a simple linear layer.

## 4. Experiments

We conducted experiments on CIFAR100 with respect to a WideResNet [32] architecture. We used an Adam optimizer, no weight regularization, cosine learning rate schedule, no data augmentation except normalization and a batch size of 64, with the number of batch repetitions 2, and input repetition probability of 0.25 [11]. For more details on the experimental setup, see Appendix B. We set the prior distribution to be uniform, i.e. $\gamma = 1/D$ for all $d$ and $m$. All experiments were repeated 3 times with different random seeds. For detailed results, look at Appendix C & D.

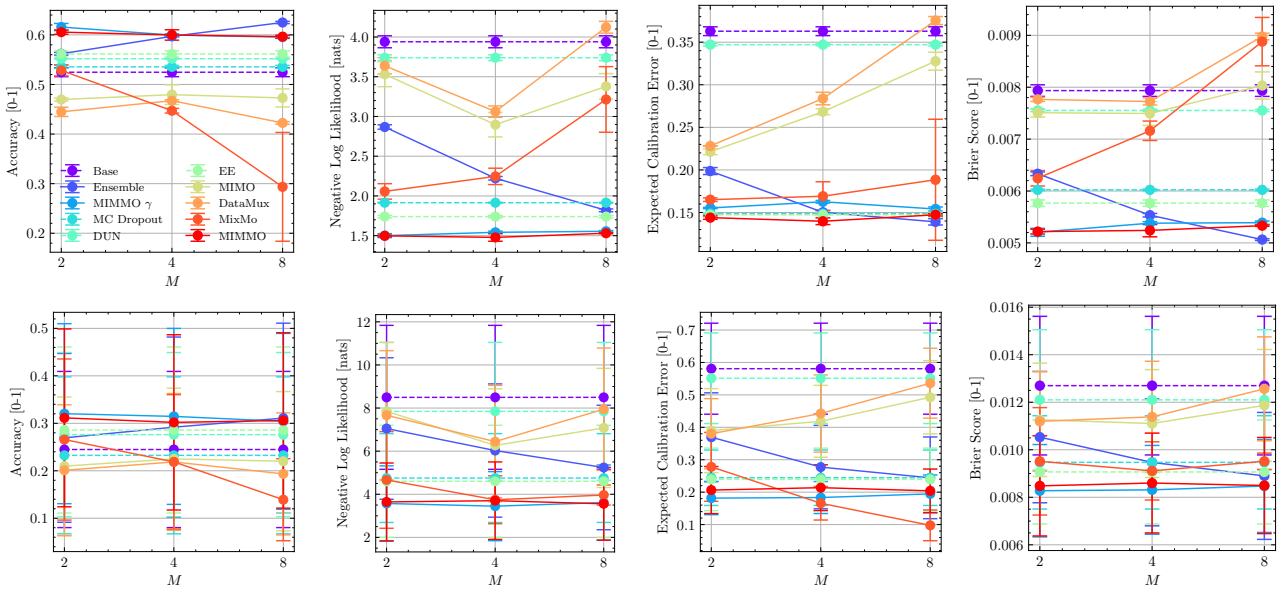


Figure 4. Performance comparison on CIFAR-100 with varying $M$. The horizontal axis shows the number of ensemble members $M$ and the vertical axis shows the accuracy, negative log-likelihood, Brier score, and expected calibration error [10]. The top row shows the results on the test dataset, the bottom row shows the results on the test dataset with augmentations. The base is a pointwise single network, Ensemble [16], MC Dropout [8], DUN [1], EE [22], MIMO [11], DataMux [20], MIMMO $\gamma$ does not optimize $\theta$ and MIMMO.

### 4.1. Measuring the Effect of $\alpha$

In Fig. 2, we investigate the influence of $\alpha$ on the probability distributions $q$. Specifically, we vary alpha in $\alpha = \{1, 100000\}$ and $m = \{1, 7\}$, while fixing $M = 8$ and $D = 4$. Moreover, we plot the input and output layer feature overlap during training for $\alpha = \{1, 100000\}$ in Fig. 3 for $M = 8$ and $D = 4$. We chose the $\alpha$ values relative to the size of the training dataset which was set to $N = 45000$ samples. Higher $\alpha$ values mean stronger regularization, targeting a distribution more similar to the prior.

As motivated earlier, given different members, it can be seen that their favored depth probabilities in a single network are different as seen in Fig. 2. Unexpectedly, the members prefer the deeper layers more than the shallower ones, most notably if the regularization strength is small, $\alpha = 1$. Moreover, as seen in Fig. 3, the input and output layer feature overlap does change with $\alpha$, meaning that it does have an impact on sharing features across the $M$ members as $\alpha$ increases. Empirically, $\alpha$ mainly impacts the depth distributions, while the width and the depth of the net have a marginal effect with $\alpha = 1$. For the main comparison, we set $\alpha_{M=2} = 100000, \alpha_{M=4} = 10000, \alpha_{M=8} = 100000$.

### 4.2. Comparison

In Fig. 4 we compared our framework with the state-of-the-art methods and varied $M$ on CIFAR-100. In the top row we show the results on the test dataset, while in the bottom row, we show the mean results on the test dataset with a range of 5 data augmentations across 5 levels of intensity. We used data augmentations to simulate out of distribution data to test calibration and robustness. We compared the algorithmic performance, plotted on the y-axis, with respect to accuracy, negative log-likelihood, Brier score, and expected calibration error [10]. For each metric, lower is better, except accuracy. On the x-axis, we varied the number of ensemble members $M$, the dashed lines represent approaches where $M$ is naturally fixed. MIMMO is able to achieve improved algorithmic performance, namely higher accuracy or lower negative log-likelihood and Brier score across $M$ on the test dataset. Interestingly, in the case of $M = 2, 4$ the method is able to achieve better accuracy and lower negative log-likelihood than the naive ensemble, as seen in Fig. 4, rows 1 and 2, columns 1 and 2. We also show results where $\theta$ parameters are not optimized as MIMMO $\gamma$ and it can be seen that even without optimization, it is beneficial to consider early-exits in the network as there are certain gains in the performance. In case $M = 2$, the optimized version is slightly worse than the unoptimized version, which demands further investigation in the direction of regularization which prevents overfitting on the training data.

In terms of the hardware cost if $M = 2, 4, 8$ MIMMO has $172.82, 174.11, 176.70$ million FLOPs and $2.04, 2.11, 2.35$ million parameters, respectively. If compared to a naive ensemble with $M = 2, 4, 8$, MIMMO is $1.27, 2.52, 4.9$ and $1.67, 3.27, 5.9$ more hardware efficient in terms of FLOPs and parameters, respectively. In comparison to the original MIMO, the FLOP and parameter counts

of MIMMO are in general $\sim 1.55$ and $\sim 1.11$ times higher, respectively. We leave the benchmarking in terms of real-world hardware cost such as latency and memory consumption for future work. For detailed results and exact FLOPs and parameter counts refer to Appendix D.

An important aspect of supporting MIMMO on real-world hardware is the capability of the designated platform to route the intermediate hidden representations from the early-exits to the final output, rather than processing each layer sequentially. Note that, the hidden representations from the early-exits do not need to be cached but rather need to be routed to the final layer and only the predictions from the final layer per depth and member need to be cached, adding a small overhead to the memory consumption. Optimising the hardware execution of MIMMO with respect to the number of early-exits and the number of members is an interesting direction for future work. Alternatively, the hardware cost of MIMMO could be reduced by considering a limited set of early-exits which would be smaller than $D$, which is achievable by modifying the prior distribution.

## 5. Conclusion

There are limitations and open questions that need to be addressed in future work. First, we used CNNs as our base architecture for MIMMO. However, given the recent advances in vision transformers [5], it would be worthwhile to investigate how MIMMO can be applied to different types of NNs. Second, we focused on image classification task on a single dataset, but there are other tasks that could benefit from MIMMO such as segmentation [14] and object detection [15]. Third, we benchmarked MIMMO with respect to FLOPs and parameters, but it would be interesting to investigate the real-world hardware cost such as latency and intermediate memory consumption.

In summary, we conceptualised, developed and investigated a method for learning adaptive depth ensembles of MIMO NNs using variational inference. We showed its potential to improve algorithmic performance and hardware efficiency compared to the MIMO and a naive ensemble. In the future, we plan to investigate the applicability of MIMMO to other tasks and NNs, as well as benchmarking and optimizing real-world hardware cost.

**Acknowledgement.** Martin Ferianc was sponsored through a scholarship from the Institute of Communications and Connected Systems at UCL. Additionaly, we would like to thank the ECV'23 reviewers, Chao Zhou, Afroditi Papadaki, Zhuo Zhi, Kristina Ulicna and Jaromir Latal for feedback on the draft.

## References

[1] Javier Antorán, James Allingham, and José Miguel Hernández-Lobato. Depth uncertainty in neural networks. *Advances in neural information processing systems*, 33:10620–10634, 2020. 1, 2, 3, 4, 9, 10, 11

[2] Fabio Arnez, Huascar Espinoza, Ansgar Radermacher, and François Terrier. A comparison of uncertainty estimation approaches in deep learning components for autonomous vehicle applications. *arXiv preprint arXiv:2006.15172*, 2020. 1

[3] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015. 2

[4] Junyi Chai, Hao Zeng, Anming Li, and Eric WT Ngai. Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Machine Learning with Applications*, 6:100134, 2021. 1

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2, 5

[6] Michael W Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable Bayesian neural nets with rank-1 factors. In *ICML*, 2020. 2

[7] Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999. 2

[8] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016. 2, 4, 9, 10, 11

[9] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021. 2

[10] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017. 1, 4, 8

[11] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M Dai, and Dustin Tran. Training independent subnetworks for robust prediction. *arXiv preprint arXiv:2010.06610*, 2020. 1, 2, 3, 4, 7, 9, 10, 11

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[13] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013. 2

[14] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017. 5

[15] Florian Kraus and Klaus Dietmayer. Uncertainty estimation in one-stage object detection. In *2019 ieee intelligent*

*transportation systems conference (itsc)*, pages 53–60. IEEE, 2019. 5

[16] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017. 1, 2, 4, 9, 10, 11

[17] Olivier Laurent, Adrien Lafage, Enzo Tartaglione, Geoffrey Daniel, Jean-Marc Martinez, Andrei Bursuc, and Gianni Franchi. Packed-ensembles for efficient uncertainty estimation. *arXiv preprint arXiv:2210.09184*, 2022. 2

[18] Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. *Advances in neural information processing systems*, 31, 2018. 2

[19] Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip HS Torr, and Yarin Gal. Deterministic neural networks with inductive biases capture epistemic and aleatoric uncertainty. *arXiv preprint arXiv:2102.11582*, 2021. 2

[20] Vishvak Murahari, Carlos E Jimenez, Runzhe Yang, and Karthik Narasimhan. Datamux: Data multiplexing for neural networks. *arXiv preprint arXiv:2202.09318*, 2022. 2, 4, 9, 10, 11

[21] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in neural information processing systems*, 32, 2019. 1, 2

[22] Lorena Qendro, Alexander Campbell, Pietro Lio, and Cecilia Mascolo. Early exit ensembles for uncertainty quantification. In *Machine Learning for Health*, pages 181–195. PMLR, 2021. 1, 2, 4, 9, 10, 11

[23] Alexandre Ramé, Rémy Sun, and Matthieu Cord. Mixmo: Mixing multiple inputs for multiple outputs via deep subnetworks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 823–833, 2021. 2

[24] Masoumeh Soflaei, Hongyu Guo, Ali Al-Bashabsheh, Yongyi Mao, and Richong Zhang. Aggregated learning: A vector-quantization approach to learning neural network classifiers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5810–5817, 2020. 2

[25] Rémy Sun, Clément Masson, Nicolas Thome, and Matthieu Cord. Adapting multi-input multi-output schemes to vision transformers. In *CVPR 2022 workshop on Efficient Deep Learning for Computer Vision (ECV)*, 2022. 2

[26] Rémy Sun, Alexandre Ramé, Clément Masson, Nicolas Thome, and Matthieu Cord. Towards efficient feature sharing in mimo architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2697–2701, 2022. 2, 8

[27] Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015. 2

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

[29] Lu Wang and KL Eddie Law. Using multiple heads to subsize meta-memorization problem. In *Artificial Neural Networks and Machine Learning–ICANN 2022: 31st International Conference on Artificial Neural Networks, Bristol, UK, September 6–9, 2022, Proceedings; Part IV*, pages 496–507. Springer, 2022. 2

[30] Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2020. 2

[31] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. In *Neural Information Processing Systems*, 2020. 2

[32] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 3, 8