A. Supplementary material

We provide further results on CIFAR100 in order to show the importance of all components of our proposed CNMMs. Moreover, we provide additional qualitative results of semantic segmentation on the CityScapes dataset.

A.1. Ablative study of CNMM

Using sampling during training. During learning, CNMMs generate a set of samples $\tilde{\pi}^{s_t}_{t-1}$ using Eq. (7). In contrast, during inference we use the expectations $\pi^{s_t}_{t-1}$ instead. In order to evaluate the importance of sampling during learning, we have optimized a CNMM by using the aforementioned expectations instead of samples. Figure 9 shows the results obtained by the model using this approach, denoted as "Training with expectations". We observe that, compared to the CNMM using sampling, the accuracy decreases faster when different pruning ratios are applied. We attribute this to the fact that our sampling procedure can be regarded as a continuous-relaxation of dropout, where a subset of functions $f^{s_t}_{t-1}(\mathbf{h}^{t-1}_{t-1})$ are randomly removed when computing the output tensor $\mathbf{h}^{s_t}_{t-1}$. As a consequence, the learned model is more robust to the pruning process where some of the convolutional blocks are removed during inference. This is not the case when deterministic expectations are used in Eq. (7) rather than samples.

Comparison with a deterministic model. We compare the performance of our CNMM with a deterministic variant using the same architecture. Concretely, in Eq. (7) we ignore samples $\tilde{\pi}^{s_t}_{t-1}$ and simply sum the feature maps $\tilde{\mathbf{h}}^{s_t}_{t-1}$ and $\tilde{\mathbf{h}}^{t-1}_{t-1}$. Note that the resulting model is analogous to a MSDNet [17] using early-exit classifiers. We report the results in Figure 9, denoted as "Deterministic with early-exits". We observe that our CNMM model obtains better performance than its deterministic counterpart. Moreover, same as MSDNets, accelerating the deterministic model is only possible by using the early-exits. In contrast, the complementary pruning algorithm available in CNMM allows for a finer granularity to control the computational cost.

Expectation approximation during inference. In order to validate our approximation of $p(y|\mathbf{X};\theta) = \mathbb{E}_{p(\mathbf{H}_T|\mathbf{X})}[p(y|\mathbf{H}_T;\theta)]$ during inference, we evaluate the performance obtained by using a Monte-Carlo procedure for the same purpose. In particular, we generate N samples from the output distribution $p(\mathbf{H}_T|\mathbf{H}_0)$. Then, we compute the class probabilities $p(y|\mathbf{H}_T;\theta)$ for each sample and average them. Table 1 shows the results obtained by varying the number of samples. We observe that our approach offers a similar performance as the Monte-Carlo approximation using $N = 5$. For a higher number of samples, we observe slight improvements in the results. However, note that a Monte-Carlo approximation is very inefficient since it requires N independent evaluations of the model.

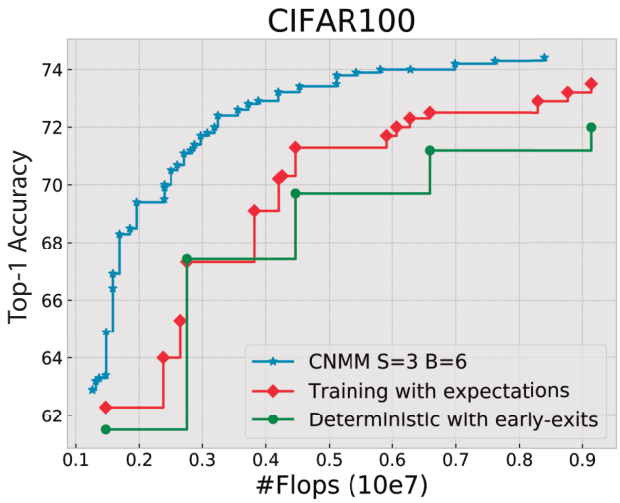

Figure 9. FLOPs *vs.* performance curves for our model (CNMM), and the variants described in Section A.1. As in Figure 7 of the main paper, the curves are obtained by using the optimal combination of early-exits and pruning when possible. In this manner, results for CNMM represent upper-envelope of all the different curves depicted in Fig. 6 of the main paper.

In particular, the last row in Table 1 is 30 times more costly to obtain than the two first rows. The minimal gain obtained with more samples could probably be more efficiently obtained by using a larger model.

Approximation	FLOPs	Top-1 Accuracy
Expectation (used)	93M	74.4
Sampling N=1	93M	71.2
Sampling N=5	463M	74.4
Sampling N=15	1390M	74.5
Sampling N=30	2780M	74.6

Table 1. Comparison of the results obtained in CIFAR100 by approximating the CNMM output using our approach or a Monte-Carlo procedure with different number of samples N.

A.2. Additional Qualitative Results

In Figure 10 we provide additional qualitative results for semantic segmentation obtained by a single trained CNMM model, using various opertating points with different number of FLOPs during inference.

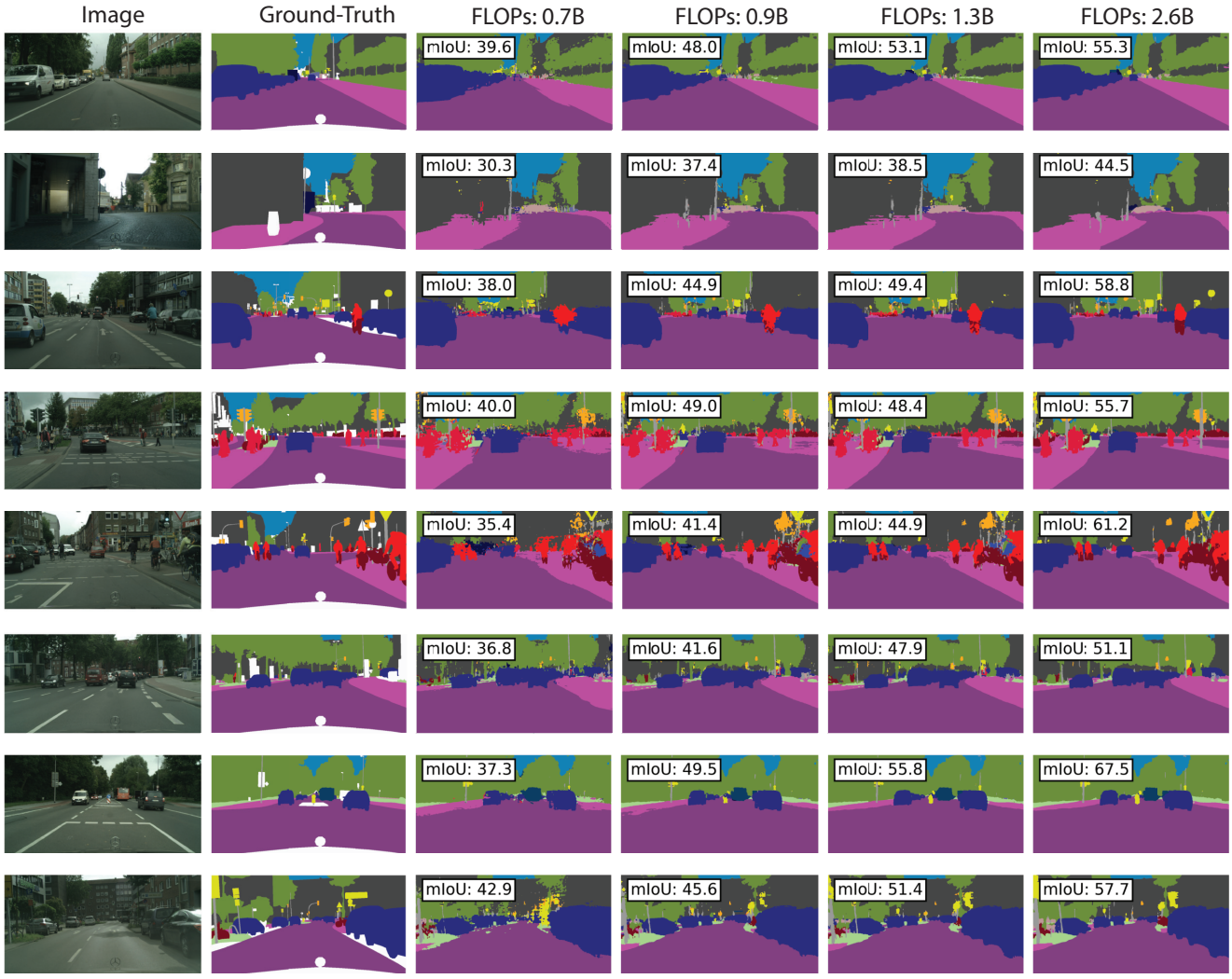

Figure 10. Pixel-level predictions for a single CNMM adapting the number of FLOPs required during inference.