9 EXTENDED APPENDIX

9.1 REDS performance on DNN and CNN architectures

In addition to the results on DS-CNN reported in the main paper, we show in Table 4 and Table 5 REDS performance on DNN and CNN architectures (with full fine-tuning) and compare to training model of each capacity from scratch and training REDS from scratch. Despite full fine-tuning, the results for S architecture show superior performance of the BU heuristic over TD.

MACs	Small (S) - Accuracy 83.82				Large (L) - Accuracy 86.87			
	Scratch	Knapsack BU	Knapsack TD	REDS training	Scratch	Knapsack BU	Knapsack TD	REDS training
100%	84.30 ±0.11	83.52 ±0.07	82.80 ±0.16	82.13 ±0.20	86.54 ±0.24	86.46 ±0.34	86.25±0.19	85.06 ±0.19
75%	83.77 ±0.23	82.29 ±0.35	81.88 ±0.30	81.23 ±0.21	85.96 ±0.13	86.38 ±0.65	86.09±0.03	84.93 ±0.20
50%	80.91 ±0.11	78.36 ±1.40	78.59 ±0.24	77.05 ±0.34	85.24 ±0.35	85.62 ±0.24	85.58±0.35	84.08 ±0.22
25%	69.77 ±0.67	64.42 ±1.99	61.43 ±3.35	63.69 ±3.26	84.22 ±0.13	82.61 ±0.61	83.00 ±0.62	82.03 ±0.59

Table 4: Test set accuracy [%] of training S and L fully-connected (DNN) architectures taken from [51]: training a network of each size from scratch ("Scratch"), conversion from a pre-trained network using two knapsack versions ("Knapsack BU" and "Knapsack TD"), and training REDS structure from scratch ("REDS training"). Reported results from three independent runs. The accuracy of each 100 % network reported in [51] is listed in the header row.

MACs	Small (S) - Accuracy 92.24				Large (L) - Accuracy 93.24			
	Scratch	Knapsack BU	Knapsack TD	REDS training	Scratch	Knapsack BU	Knapsack TD	REDS training
100%	91.10±0.23	91.60±0.39	91.20 ±0.35	88.89 ±0.26	92.94±0.20	92.83±0.26	92.97±0.15	90.97 ±0.21
75%	90.40±0.27	90.63±0.19	90.20 ±0.13	87.64 ±0.46	92.74±0.12	92.74±0.23	92.54 ±0.07	90.67 ±0.09
50%	89.07±0.24	88.39±0.31	88.39 ±0.52	85.99 ±0.19	92.44±0.30	91.95±0.22	91.93 ±0.28	90.36 ±0.30
25%	82.57±0.41	79.52±0.67	79.28 ±0.54	79.25 ±0.40	90.98±0.60	90.24±0.35	90.31 ±0.03	88.25 ±0.66

Table 5: The same as in Table 4 for S and L convolutional architectures (CNN) from [51].

Fig. 10 shows the impact of the architecture on REDS structure found by the knapsack BU solver. We present the results for DNN S, L and CNN S, L from left to right, respectively.

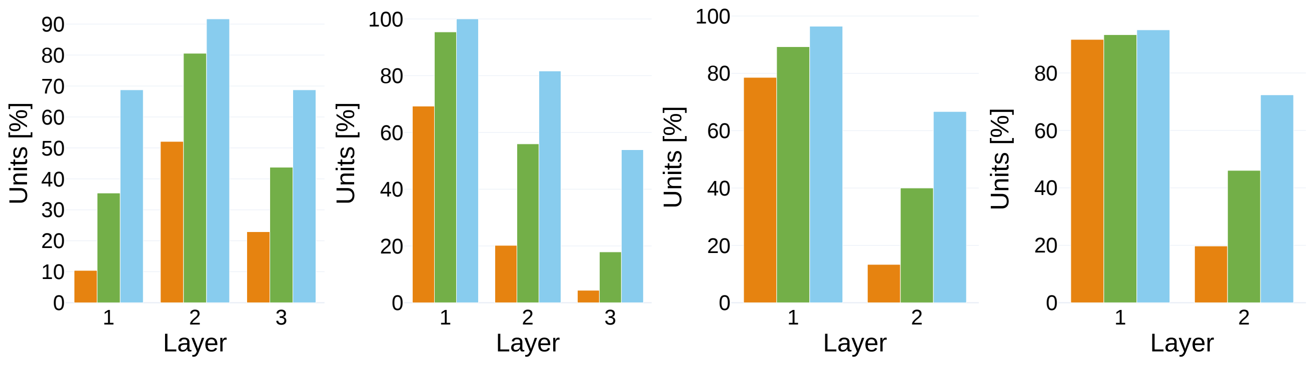

Figure 10: Analysis of the subnetwork architecture obtained by the knapsack BU heuristics. From left to right: DNN S, DNN L, CNN S and CNN L on Google Speech Commands. The patterns as to which computational units constitute a child subnetwork are architecture-specific.

9.2 Iterative knapsack problem proofs

Theorem 1. *A_c yields a worst case ratio of $\frac{2}{3}$*, i.e., *$P(I(A_c)) \geq \frac{2}{3} \cdot P(I(Opt_c))$, if all items of the knapsack have a weight $\leq c/2$. This bound is tight.*

Proof. Let us arrange the items of $I(Opt_c)$ in the following order O: we first take all the items from $I(Opt_c) \cap I(Opt_{c/2})$ in an arbitrary order. Note that these are the items that are in both optimal solutions, *i.e.*, for both capacity c and $c/2$. Then we take all the items that are not included in $I(A_{c/2})$ followed by the items of $I(Opt_c) \cap I(A_{c/2})$ (again in arbitrary order). Now we have two cases:

Case 1: There does not exist a split item in $I(Opt_c)$ with respect to O and capacity $c/2$. Hence $W(I(Opt_{c/2})) = c/2$. It is easy to see that in this case $A_c = Opt_c$.

Case 2: Let I_s be the split item in $I(Opt_c)$ with respect to O and capacity $c/2$. In this case we get that the weight of all the items I_b^O before I_s as well as the weight of all the items I_f^O that follow I_s is smaller than $c/2$. It follows that $P(I(Opt_{c/2})) \geq P(I_b^O)$ and that $P(I(A_{c/2})) \geq P(I_f^O)$. Since all items have a weight $\leq c/2$ and by the fact that I_s is not contained in $I(Opt_{c/2})$ we know that its profit is less or equal than the minimum of $P(I(A_{c/2}))$ and $P(I(Opt_{c/2}))$. Therefore, it holds that $P(I_s) \leq \frac{1}{2}P(I(A_c))$. Hence we get:

$$\begin{aligned} P(I(Opt_c)) &= P(I_b^O) + P(I_s) + P(I_f^O) \leq P(I(A_c)) + \frac{1}{2}P(I(A_c)) \\ &= \frac{3}{2}P(I(A_c)) \end{aligned}$$

It remains to show the bound is tight. We introduce the following knapsack instance with four items and a large positive constant P.

item:	1	2	3	4
weight:	$c/3 + \epsilon$	$c/3$	$c/3$	$c/3$
profit:	$P + \epsilon$	P	P	P

Here $I(Opt_{c/2}) = \{1\}$ which only leaves space for one additional item for the larger capacity. Hence we get that $P(I(A_c)) = 2P + \epsilon$, whereas $P(I(Opt_c)) = 3P$. □

Theorem 2. *$D_{c/2}$ yields a worst case ratio of $\frac{1}{2}$*, i.e., *$P(I(D_{c/2})) \geq \frac{1}{2} \cdot P(I(Opt_{c/2}))$ if all items of the knapsack have a weight $\leq c/2$. This bound is tight.*

Proof. Consider a knapsack of size c with optimal solution set $I(Opt_c)$ and the knapsack problem with capacity $c/2$ defined on the restricted item set $I(Opt_c)$ with solution set $I(D_{c/2})$. We will show that: $P(I(D_{c/2})) \geq \frac{1}{2} \cdot P(I(Opt_{c/2}))$.

We first arrange the items of $I(Opt_c)$ in an ordering O' such that they start with those items contained also in $I(Opt_{c/2})$. Then we identify the split item I_s according to O' for capacity $c/2$ and partition $I(Opt_c)$ into three parts. $D_1 = I_b^{O'}$, $D_2 = I_s$ and D_3 contains all the remaining items. If no split item exists, we simply set $I_s = \emptyset$. We now show that:

$$\max(P(D_1), P(D_2), P(D_3)) \geq \frac{P(I(Opt_{c/2}))}{2} \tag{3}$$

Assuming that this is not the case, we would get that:

$$\max(P(D_1), P(D_2), P(D_3)) < \frac{P(I(Opt_{c/2}))}{2}$$

This would imply

$$P(I(Opt_c)) = P(D_1) + P(D_2) + P(D_3) < P(I(Opt_{c/2})) + P(D_3).$$

However, since $I(Opt_{c/2}) \cap D_3 = \emptyset$ and $W(I(Opt_{c/2})) \leq c/2$, $I(Opt_{c/2}) \cup D_3$ would constitute a feasible solution better than $I(Opt_c)$, which is a contradiction. Thus, we have shown (3).

For $i = 1, \ldots, 3$, there is $W(D_i) \leq c/2$ and all items in D_i are available for $D_{c/2}$. Therefore, (3) implies

$$P(I(D_{c/2})) \geq \max(P(D_1), P(D_2), P(D_3)) \geq \frac{1}{2}P(I(Opt_{c/2})).$$

It remains to show the bound is tight. We introduce the following knapsack instance with four items and a large positive constant P.

Item:	1	2	3	4
Weight:	$c/3$	$c/3$	$c/3$	$c/2$
Profit:	$P+\epsilon$	$P+\epsilon$	$P+\epsilon$	2P

Here $I(Opt_c) = \{1,2,3\}$. $D_{c/2}$ then selects one of these items and no more items fit into the knapsack. $Opt_{c/2}$ selects item 4, which shows that the ratio of $\frac{1}{2}$ is tight. □

Note that in case that we have instances, where the weight of certain items is greater that $c/2$, it is easy to construct instances with arbitrary bad ratios for both cases.

9.3 REDS performance with 10 nested subnetworks

Table 6 and Fig. 11 show the performance of DNN, CNN and DS-CNN on Google Speech Commands, when REDS structure comprises 10 subnetworks, compared to 4 subnetworks in the main paper. A larger number of subnetworks does not degrade model accuracy.

MACs	**DNN**		**CNN**		**DS-CNN**	
	Small	Large	Small	Large	Small	Large
100%	83.07 ±0.35	86.19 ±0.26	91.16 ±0.45	93.1 ±0.1	93.5 ±0.15	94.34 ±0.07
90%	82.93 ±0.4	86.17 ±0.36	90.4 ±0.47	92.84 ±0.27	93.33 ±0.11	94.32 ±0.1
80%	82.67 ±0.53	86.1 ±0.08	90.1 ±0.07	92.77 ±0.06	92.84 ±0.15	94.31 ±0.1
70%	81.67 ±0.53	85.78 ±0.11	89.43 ±0.41	92.25 ±0.47	92.64 ±0.24	94.21 ±0.14
60%	80.37 ±0.57	85.66 ±0.25	88.84 ±0.25	92.28 ±0.11	92.27 ±0.31	94.08 ±0.03
50%	78.26 ±0.53	85.33 ±0.09	88.4 ±0.2	92.03 ±0.23	91.04 ±0.51	93.97 ±0.04
40%	75.37 ±1.26	84.8 ±0.45	85.76 ±0.18	91.78 ±0.04	89.42 ±0.66	93.83 ±0.22
30%	67.76 ±1.59	82.66 ±0.18	81.92 ±0.49	90.52 ±0.54	87.63 ±1.03	93.59 ±0.14
20%	52.36 ±6.99	80.19 ±0.39	73.74 ±0.04	88.61 ±0.51	84.14 ±2.57	93.35 ±0.15
10%	23.93 ±5.88	50.7 ±7.5	58.87 ±5.27	81.15 ±1.39	58.46 ±3.35	90.38 ±0.17

Table 6: Test set accuracy [%] from Small (S) and Large (L) pretrained fully-connected (DNN), convolutional (CNN), and depth-wise separable convolutional (DS-CNN) networks taken from [51]. For each pre-trained architecture, REDS can support ten subnetworks obtained from the Knapsack BU formulation. The accuracies of the DS-CNN and CNN subnetworks do not degrade drastically until the lowest percentage of MACs considered. In contrast, the accuracies in the DNN subnetworks show a more pronounced drop from 30% MACs.

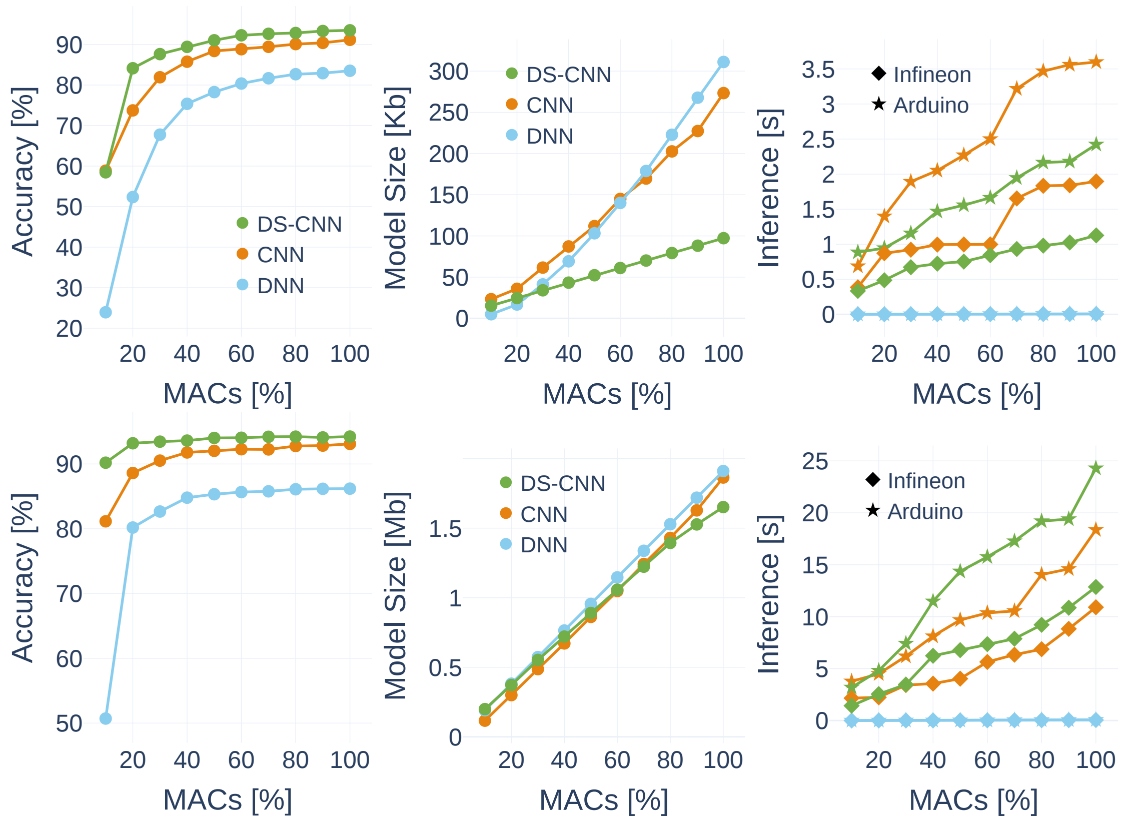

Figure 11: REDS size S (top row) and L (bottom row) architectures analysis finetuned on Google Speech Commands [49] with ten subnetworks. The plots from left to right show the subnetworks size, the subnetworks accuracy and the subnetworks inference time as a function of MAC percentage.

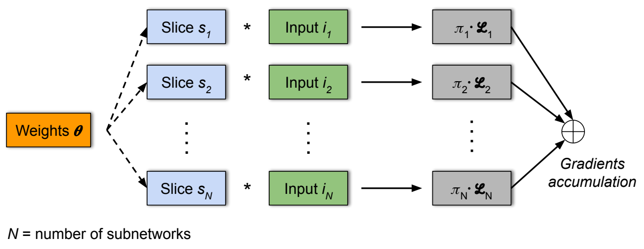
Weights θ
Slice s_1
Slice s_2
Slice s_N
Input i_1
Input i_2
Input i_N
$\pi_1 \cdot \mathcal{L}_1$
$\pi_2 \cdot \mathcal{L}_2$
$\pi_N \cdot \mathcal{L}_N$
Gradients accumulation
N = number of subnetworks

Figure 12: REDS fine-tuning. The parameters $\pi_i{}_{i=1}^{N}$ ensure the contribution of individual models to the loss aligns with the fraction of the shared weights.

9.4 REDS on Fashion-MNIST and CIFAR10

The results in Table 7 and Table 8 show REDS performance using DS-CNN architecture of size S on Fashion-MNIST and CIFAR10. BU heuristic was used to obtain the results. REDS supports a different data domain without degrading the accuracy of the pre-trained model, reported in the header row. Compared to the state-of-the-art such as μNAS [32], REDS demonstrates a faster architecture search time for both Fashion-MNIST and CIFAR10. In the former, REDS takes 19 minutes as opposed to 3 days; in the latter, REDS takes 90 minutes as opposed to 39 days while requiring less memory for model storage for both datasets. After finding and freezing the 25% MACs subnetwork architecture, the BU heuristic takes only a few seconds to find the other 50% and 75% MACs subnetworks architectures.

MACs	Acc (%) - Pre-trained 90.59	Model Size (Kb)	Time Taken (m)
100%	91.6 ±0.2	128.54	–
75%	91.51 ±0.28	107.73	1.58 [s]
50%	90.75 ±0.32	87.4	5.83 [s]
25%	89.22 ±0.45	66.63	19 [m]

Table 7: Analysis of the BU knapsack subnetworks obtained from a depth-wise separable convolutional (DS-CNN) S network, pre-trained on Fashion-MNIST [50]. REDS supports a different data domain without degrading the accuracy of the pre-trained model, reported in the header row.

MACs	Acc (%) - Pre-trained 79.36	Model Size (Kb)	Time Taken
100%	81.07 ±0.71	128.54	–
75%	80.17 ±0.69	109.41	2.89 [s]
50%	76.72 ±1.37	88.01	10.59 [s]
25%	68.66 ±1.65	69.63	90 [m]

Table 8: The same evaluation as in Table 7 for CIFAR10 [29].

9.5 REDS energy efficiency

Table 9 reports the energy consumption of inference time by different network architectures measured by Power Profiler Kit (PPK2) on Nordic nRF52840 (Arduino Nano 33 BLE Sense). In comparison, the model adaptation time takes less than 0.01 mJ.

MACs	DNN [mJ]	CNN [mJ]	DS-CNN [mJ]
100%	0.18	90.61	61
75%	0.15	86.01	44.03
50%	0.07	50.3	33.81
25%	0.05	44.81	20.44

Table 9: Energy consumption of REDS size S architectures measured by the Power Profiler Kit (PPK2) on Nordic nRF52840. The results are obtained by performing an inference pass for each subnetwork model and recording the inference current.