Appendix

Appendix A. An Overview of the Method and Computational Costs

Figure 9 presents and overview scheme of the method:

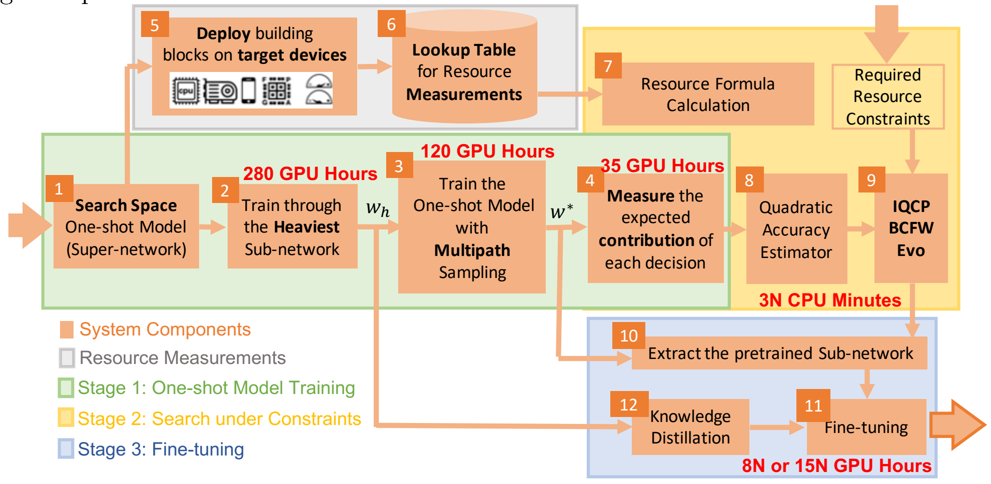

5 Deploy building blocks on target devices
6 Lookup Table for Resource Measurements
7 Resource Formula Calculation
Required Resource Constraints
1 Search Space One-shot Model (Super-network)
280 GPU Hours
2 Train through the Heaviest Sub-network
w_h
120 GPU Hours
3 Train the One-shot Model with Multipath Sampling
w^*
35 GPU Hours
4 Measure the expected contribution of each decision
8 Quadratic Accuracy Estimator
9 IQCP BCFW Evo
3N CPU Minutes
10 Extract the pretrained Sub-network
12 Knowledge Distillation
11 Fine-tuning
8N or 15N GPU Hours
System Components
Resource Measurements
Stage 1: One-shot Model Training
Stage 2: Search under Constraints
Stage 3: Fine-tuning

Figure 9: An Overview scheme of the IQNAS method with computational costs

The search space, latency measurements and formula, supernetwork training and fine-tuning blocks (1,2,3,5,6,7,10,11,12) are identical to those introduced in HardCoRe-NAS:

We first train for 250 epochs a one-shot model w_h using the heaviest possible configuration, i.e., a depth of 4 for all stages, with $er = 6, k = 5 \times 5, se = on$ for all the blocks. Next, to obtain w^*, for additional 100 epochs of fine-tuning w_h over 80% of a 80-20 random split of the ImageNet train set Deng et al. (2009). The training settings are specified in appendix C. The first 250 epochs took 280 GPU hours and the additional 100 fine-tuning epochs took 120 GPU hours, both Running with a batch size of 200 on 8×NVIDIA V100, summing to a total of 400 hours on NVIDIA V100 GPU to obtain both w_h and w^*.

A significant benefit of this training scheme is that it also shortens the generation of trained models. The common approach of most NAS methods is to re-train the extracted sub-networks from scratch. Instead, we follow HardCoRe-NAS and leverage having two sets of weights: w_h and w^*. Instead of retraining the generated sub-networks from a random initialization we opt for fine-tuning w^* guided by knowledge distillation Hinton et al. (2015) from the heaviest model w_h. Empirically, we observe that this surpasses the accuracy obtained when training from scratch at a fraction of the time: 8 GPU hours for each generated network.

The key differences from HardCoRe-NAS reside in the latency constrained search blocks (4,8,9 of figure 9) that have to do with constructing the quadratic accuracy estimator of section 3.3 and solving the IQCQP problem. The later requires only several minutes on CPU for each generated netowork (compared to 7 GPU hours of HardCoRe-NAS), while the former requires to measure the individual accuracy contribution of each decision. Running a validation epoch to estimate $\mathbb{E}[Acc]$ and also for each decision out of all 255 entries in the vector ζ to obtain $\Delta^s_{b,c}$ and Δ^s_b requires 256 validation epochs in total that last for 3.5 GPU hours. Figure 7 (Left) shows that reducing the variance by taking 10 validation epochs per measurement is beneficial. Thus a total of 2560 validation epochs requires 35 GPU hours only once.

Overall, we are able to generate a trained model within a small marginal cost of 8 GPU hours. The total cost for generating N trained models is $435 + 8N$, much lower than the $1200+25N$ reported by OFA Cai et al. (2019) and more scalable compared to the $400+15N$ reported by HardCoRe-NAS. See Table 1. The reduced search cost frees more compute that can be utilized for a longer fine-tuning of 15 GPU hours to surpass HardCoRe-NAS with the same marginal cost of $15N$. This makes our method scalable for many devices and latency requirements.

Appendix B. More Specifications of the Search Space

Inspired by EfficientNet Tan and Le (2019) and TF-NAS Hu et al. (2020), HardCoRe-NAS Nayman et al. (2021) builds a layer-wise search space that we utilize, as explained in Section 3.1 and detailed in Table 3. The input shapes and the channel numbers are the same as EfficientNetB0. Similarly to TF-NAS and differently from EfficientNet-B0, we use ReLU in the first three stages. As specified in Section 3.1, the ElasticMBInvRes block is the *elastic* version of the MBInvRes block as in HardCoRe-NAS, introduced in Sandler et al. (2018). Those blocks of stages 3 to 8 are to be searched for, while the rest are fixed.

Stage	Input	Operation	C_{out}	Act	b
1	$224^2 \times 3$	3×3 Conv	32	ReLU	1
2	$112^3 \times 32$	MBInvRes	16	ReLU	1
3	$112^2 \times 16$	ElasticMBInvRes	24	ReLU	$[2, 4]$
4	$56^2 \times 24$	ElasticMBInvRes	40	Swish	$[2, 4]$
5	$28^2 \times 40$	ElasticMBInvRes	80	Swish	$[2, 4]$
6	$14^2 \times 80$	ElasticMBInvRes	112	Swish	$[2, 4]$
7	$14^2 \times 112$	ElasticMBInvRes	192	Swish	$[2, 4]$
8	$7^2 \times 192$	ElasticMBInvRes	960	Swish	1
9	$7^2 \times 960$	1×1 Conv	1280	Swish	1
10	$7^2 \times 1280$	AvgPool	1280	-	1
11	1280	Fc	1000	-	1

c	er	k	se
1	2	3×3	off
2	2	3×3	on
3	2	5×5	off
4	2	5×5	on
5	3	3×3	off
6	3	3×3	on
7	3	5×5	off
8	3	5×5	on
9	6	3×3	off
10	6	3×3	on
11	6	5×5	off
12	6	5×5	on

Table 3: Search space specifications and indexing. "MBInvRes" is the basic block in Sandler et al. (2018). "ElasticMBInvRes" denotes the *elastic* blocks (Section 3.1) to be searched for. "C_{out}" stands for the output channels. Act denotes the activation function used in a stage. "b" is the number of blocks in a stage, where $[\underline{b}, \bar{b}]$ is a discrete interval. If necessary, the down-sampling occurs at the first block of a stage. "er" stands for the expansion ratio of the point-wise convolutions, "k" stands for the kernel size of the depth-wise separable convolutions and "se" stands for Squeeze-and-Excitation (SE) with *on* and off denoting with and without SE respectively. The configurations are indexed according to their expected latency.

Appendix C. Reproducibility and Experimental Setting

In all our experiments we train the networks using SGD with a learning rate of 0.1, cosine annealing, Nesterov momentum of 0.9, weight decay of 10^{-4}, applying label smoothing

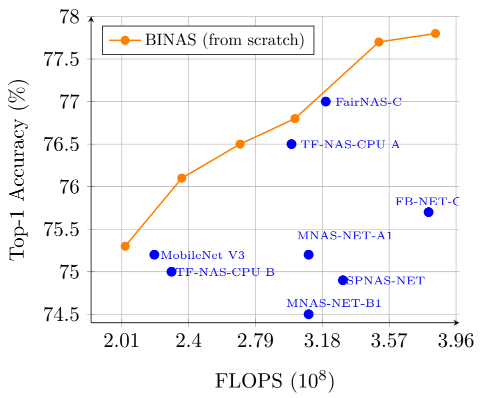
Szegedy et al. (2016) of 0.1, cutout, Autoaugment Cubuk et al. (2018), mixed precision and EMA-smoothing.

The supernetwork is trained following Nayman et al. (2021) over 80% of a random 80-20 split of the ImageNet train set. We utilize the remaining 20% as a validation set for collecting data to obtain the accuracy predictors and for architecture search with latencies of $40, 45, 50, \ldots, 60$ and $25, 30, 40$ milliseconds running with a batch size of 1 and 64 on an Intel Xeon CPU and and NVIDIA P100 GPU, respectively.

The evolutionary search implementation is adapted from Cai et al. (2019) with a population size of 100, mutation probability of 0.1, parent ratio of 0.25 and mutation ratio of 0.5. It runs for 500 iterations, while the the BCFW runs for 2000 iterations and its projection step for 1000 iterations. The MIQCP solver runs up to 100 seconds with CPLEX default settings.

Appendix D. Comparing other Resources

Figure 10 shows a comparison of Imagenet top-1 accuracy and FLOPS between generated models by our method and other leading NAS methods, some of which optimized for FLOPS. Our models outperform the rest in terms of the tradeoff between accuracy and FLOPS.



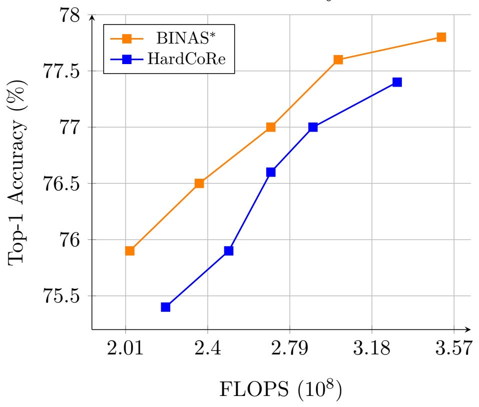


Figure 10: Imagenet Top-1 accuracy vs FLOPS. (Right) All models are trained from scratch. BINAS generates better models than other methods with a reduced scalable marginal cost of 8 GPU hours for each additional model. (Right) Surpassing State-of-the-Art. BINAS generates superior models than HardCoReNAS for the same search space, supernetwork weights and marginal cost of 15 GPU hours.

Table 4 presents comparisons in terms of model size, e.g. number of parameters, FLOPS together with latency.

Appendix E. Proof of Theorem 3.1

Theorem E.1 *Consider n independent random variables $\{X_i\}_{i\in[1,2\ldots,n]}$ conditionally independent with another random variable A. Suppose in addition that there exists a positive real number $0 < \epsilon \ll 1$ such that for any given X_i, the following term is bounded by above:*

$$\left|\frac{\mathbb{P}\left[A=a|X_i=x_i\right]}{\mathbb{P}\left[A=a\right]}-1\right|<\epsilon.$$

Model	Size (10^6)	FLOPS (10^8)	Latency (ms)	Top-1 (%)
MnasNetB1	4.4	3.1	39	74.5
TFNAS-B	4.9	2.3	40	75.0
SPNASNet	4.4	3.3	41	74.9
OFA CPU	4.9	3.6	42	75.7
HardCoRe A	5.3	2.5	40	75.8
BINAS*(40)	5.6	2.3	40	76.5
MobileNetV3	5.4	2.2	45	75.2
FBNet	5.5	3.8	47	75.7
MnasNetA1	3.9	3.1	55	75.2
HardCoRe B	5.2	2.7	44	76.4
BINAS*(45)	7.1	2.7	45	77.0
MobileNetV2	6.1	5.8	70	76.5
TFNAS-A	7.1	3.0	60	76.5
HardCoRe C	7.5	2.9	50	77.1
BINAS*(50)	7.0	3.0	50	77.6
EfficientNetB0	5.3	3.9	85	77.3
HardCoRe D	8.1	3.3	55	77.6
BINAS*(55)	7.7	3.5	55	77.8
FairNAS-C	4.4	3.2	60	77.0
HardCoRe E	8.2	3.4	61	78.0
BINAS*(60)	8.2	3.8	59	78.0

Table 4: ImageNet top-1 accuracy, model size (number of parameters), FLOPS and latency comparison with other methods. The latency is reported for Intel Xeon CPU with a batch size of 1.

Then we have that:

$$\mathbb{E}\left[A|X_1=x_1,\ldots,X_n=x_n\right]=\mathbb{E}\left[A\right]+\sum_i\left(\mathbb{E}\left[A|X_i=x_i\right]-\mathbb{E}\left[A\right]\right)\left(1+O(n\epsilon)\right)$$

Proof We consider two independent random variables X, Y that are also conditionally independent with another random variable A. Our purpose is to approximate the following conditional expectation:

$$\mathbb{E}\left[A|X=x,Y=y\right].$$

We start by writing :

$$\mathbb{P}\left[A=a|X=x,Y=y\right]=\frac{\mathbb{P}\left[X=x,Y=y|A=a\right]\mathbb{P}\left[A=a\right]}{\mathbb{P}\left[X=x,Y=y\right]}$$

Assuming the conditional independence, that is

$$\mathbb{P}\left[X=x,Y=y|A=a\right]=\mathbb{P}\left[X=x|A=a\right]\mathbb{P}\left[Y=y|A=a\right],$$

we have

$$\begin{aligned}\mathbb{P}\left[A=a|X=x,Y=y\right] &= \frac{\mathbb{P}\left[X=x|A=a\right]\mathbb{P}\left[Y=y|A=a\right]\mathbb{P}\left[A=a\right]}{\mathbb{P}\left[X=x\right]\mathbb{P}\left[Y=y\right]} \\ &= \frac{\mathbb{P}\left[A=a|X=x\right]\mathbb{P}\left[A=a|Y=y\right]}{\mathbb{P}\left[A=a\right]}\end{aligned} \tag{13}$$

Next, we assume that the impact on A of the knowledge of X is bounded, meaning that there is a positive real number $0<\epsilon\ll 1$ such that:

$$\left|\frac{\mathbb{P}\left[A=a|X=x\right]}{\mathbb{P}\left[A=a\right]}-1\right|<\epsilon.$$

We have:

$$\begin{aligned}\mathbb{P}\left[A=a|X=x\right] &= \mathbb{P}\left[A=a\right]+\mathbb{P}\left[A=a|X=x\right]-\mathbb{P}\left[A=a\right] \\ &= \mathbb{P}\left[A=a\right]\left(1+\underbrace{\left(\frac{\mathbb{P}\left[A=a|X=x\right]}{\mathbb{P}\left[A=a\right]}-1\right)}_{\epsilon_x}\right)\end{aligned}$$

Using a similar development for $\mathbb{P}\left[A=a|Y=y\right]$ we also have:

$$\mathbb{P}\left[A=a|Y=y\right] = \mathbb{P}\left[A=a\right]\left(1+\underbrace{\left(\frac{\mathbb{P}\left[A=a|Y=y\right]}{\mathbb{P}\left[A=a\right]}-1\right)}_{\epsilon_y}\right)$$

Plugging the two above equations in (13), we have:

$$\begin{aligned}&\mathbb{P}\left[A=a|X=x,Y=y\right]=\frac{\mathbb{P}\left[A=a\right](1+\epsilon_x)\mathbb{P}\left[A=a\right](1+\epsilon_y)}{\mathbb{P}\left[A=a\right]} \\ &=\mathbb{P}\left[A=a\right](1+\epsilon_x)(1+\epsilon_y) \\ &=\mathbb{P}\left[A=a\right](1+\epsilon_x+\epsilon_y)+O(\epsilon^2) \\ &=\mathbb{P}\left[A=a\right]\left(1+\left(\frac{\mathbb{P}\left[A=a|X=x\right]}{\mathbb{P}\left[A=a\right]}-1\right)+\left(\frac{\mathbb{P}\left[A=a|Y=y\right]}{\mathbb{P}\left[A=a\right]}-1\right)\right)+O(\epsilon^2) \\ &=\mathbb{P}\left[A=a\right]\left(\frac{\mathbb{P}\left[A=a|X=x\right]}{\mathbb{P}\left[A=a\right]}+\frac{\mathbb{P}\left[A=a|Y=y\right]}{\mathbb{P}\left[A=a\right]}-1\right)+O(\epsilon^2) \\ &=\mathbb{P}\left[A=a|X=x\right]+\mathbb{P}\left[A=a|Y=y\right]-\mathbb{P}\left[A=a\right]+O(\epsilon^2)\end{aligned}$$

Then, integrating over a to get the expectation leads to:

$$\begin{aligned}\mathbb{E}\left[A|X=x,Y=y\right]=&\int_a a\mathbb{P}\left[A=a|X=x,Y=y\right]da \\ =&\mathbb{E}\left[A|X=x\right]+\mathbb{E}\left[A|Y=y\right]-\mathbb{E}\left[A\right]+O(\epsilon^2) \\ =&\mathbb{E}\left[A\right]+(\mathbb{E}\left[A|X=x\right]-\mathbb{E}\left[A\right]) \\ &+(\mathbb{E}\left[A|Y=y\right]-\mathbb{E}\left[A\right])+O(\epsilon^2).\end{aligned}$$

In the case of more than two random variables, $X_1,X_2,\ldots,X_n$, denoting by $u_n=\mathbb{E}\left[A|X_1=x_1,\ldots,X_n=x_n\right]-\mathbb{E}\left[A\right]$, and by $v_n=\mathbb{E}\left[A|X_n=x_n\right]-\mathbb{E}\left[A\right]$, we have $|u_n-u_{n-1}-v_n|<$

ϵu_{n-1} A simple induction shows that:

$$\begin{aligned} v_n + (1-\epsilon)v_{n-1} + &(1-\epsilon)^2 v_{n-2} + \cdots + (1-\epsilon)^n v_1 \\ &< u_n < v_n + (1+\epsilon)v_{n-1} + (1+\epsilon)^2 v_{n-2} + \cdots + (1+\epsilon)^n v_1 \end{aligned}$$

Hence:

$$(1-\epsilon)\sum v_i < u_n < (1+\epsilon)^n \sum v_i$$

that shows that

$$u_n = \left(\sum v_i\right)(1 + O(n\epsilon))$$

■

Now we utilize Theorem E.1 for proving Theorem 3.1. Consider a one-shot model whose subnetworks' accuracy one wants to estimate:
$\mathbb{E}[Acc| \cap_{s=1}^{S} \cap_{b=1}^{D} O_b^s, \cap_{s=1}^{S} d^s]$, with $\alpha_{b,c}^s = \mathbb{1}_{O_b^s = O_c}$ the one-hot vector specifying the selection of configuration c for block b of stage s and $\beta_b^s = \mathbb{1}_{d^s = b}$ the one-hot vector specifying the selection of depth b for stage s, such that $(\boldsymbol{\alpha}, \boldsymbol{\beta}) \in \mathcal{S}$ with $\mathcal{S}$ specified in equation 2 as described in section 3.1.

We simplify our problem and assume $\{O_b^s, d^s\}, d^s$ for $s = 1, \ldots, S$ and $b = 1, \ldots, D$ are conditionally independent with the accuracy Acc. In our setting, we have

$$\begin{aligned} \mathbb{E}[Acc| \cap_{s=1}^{S} \cap_{b=1}^{D} O_b^s; \cap_{s=1}^{S} d^s] &= \mathbb{E}[Acc| \cap_{s=1}^{S} \cap_{b=1}^{d^s} \{O_b^s, d^s\}; \cap_{s=1}^{S} d^s] \qquad (14) \\ &\approx \mathbb{E}[Acc] \\ &+ \sum_{s=1}^{S} \sum_{b=1}^{D} \left(\mathbb{E}[Acc|O_b^s, d^s] - \mathbb{E}[Acc]\right) \mathbb{1}_{b \le d_s} (1 + O(N\epsilon)) \qquad (15) \\ &+ \sum_{s=1}^{S} \left(\mathbb{E}[Acc|d^s] - \mathbb{E}[Acc]\right) \mathbb{1}_{b \le d^s} (1 + O(N\epsilon)) \qquad (16) \end{aligned}$$

where equation 14 is since the accuracy is independent of blocks that are not participating in the subnetowrk, i.e. with $b > d^s$, and equations 15 and 16 are by utilizing Theorem E.1.

Denote by b^s and c_b^s to be the single non zero entries of β^s and α_b^s respectively, whose entries are β_b^s for $b = 1, \ldots, D$ and $\alpha_{b,c}^s$ for $c \in \mathcal{C}$ respectively. Hence $\beta_b^s = \mathbb{1}_{b=b^s}$ and $\alpha_{b,c}^s = \mathbb{1}_{c=c_b^s}$. Thus we have,

$$\begin{aligned} \mathbb{E}[Acc|d^s = b^s] - \mathbb{E}[Acc] &= \sum_{b=1}^{D} \mathbb{1}_{b=b^s} (\mathbb{E}[Acc|d^s = b] - \mathbb{E}[Acc]) \\ &= \sum_{b=1}^{D} \beta_b^s (\mathbb{E}[Acc|d^s = b] - \mathbb{E}[Acc]) = \sum_{b=1}^{D} \beta_b^s \Delta_b^s \qquad (17) \end{aligned}$$

Similarly,

$$\begin{aligned}\mathbb{E}\left[Acc|O_b^s = O_{c_b^s}, d_s = b\right] - \mathbb{E}\left[Acc\right] &= \sum_{c\in\mathcal{C}} \mathbb{1}_{c=c_b^s}(\mathbb{E}\left[Acc|O_b^s = O_c, d_s = b\right] - \mathbb{E}\left[Acc\right]) \\ &= \sum_{c\in\mathcal{C}} \alpha_{b,c}^s(\mathbb{E}\left[Acc|O_b^s = O_c, d_s = b\right] - \mathbb{E}\left[Acc\right]) \\ &= \sum_{c\in\mathcal{C}} \alpha_{b,c}^s \Delta_{b,c}^s \qquad (18)\end{aligned}$$

And since effectively $\mathbb{1}_{b\leq d^s} = \sum_{b'=b}^{D} \beta_{b'}^s$ we have,

$$\begin{aligned}\left(\mathbb{E}\left[Acc|O_b^s = O_{c_b^s}, d_s = b\right] - \mathbb{E}\left[Acc\right]\right) \mathbb{1}_{b\leq d_s} &= \sum_{c\in\mathcal{C}} \alpha_{b,c}^s \Delta_{b,c}^s \cdot \mathbb{1}_{b\leq d_s} \\ &= \sum_{b'=b}^{D}\sum_{c\in\mathcal{C}} \alpha_{b,c}^s \cdot \Delta_{b,c}^s \cdot \beta_{b'}^s \qquad (19)\end{aligned}$$

Finally by setting equations 17 and 19 into 15 and 16 respectively, we have,

$$\begin{aligned}\mathbb{E}\left[Acc\middle|\cap_{s=1}^{S}\cap_{b=1}^{D} O_b^s, \cap_{s=1}^{S} d^s\right] = \mathbb{E}[Acc] &+ (1+\mathcal{O}(N\epsilon)) \cdot \sum_{s=1}^{S}\sum_{b=1}^{D} \beta_b^s \cdot \Delta_b^s \\ &+ (1+\mathcal{O}(N\epsilon)) \cdot \sum_{s=1}^{S}\sum_{b=1}^{D}\sum_{b'=b}^{D}\sum_{c\in\mathcal{C}} \alpha_{b,c}^s \cdot \Delta_{b,c}^s \cdot \beta_{b'}^s\end{aligned}$$

Appendix F. Deriving a Closed Form Solution for a Linear Regression

We are given a set (X, Y) of architecture encoding vectors and their accuracy measured on a validation set.

We seek for a quadratic predictor f defined by parameters $\mathbf{Q} \in \mathbb{R}^{n\times n}, \mathbf{a} \in \mathbb{R}^n, b \in \mathbb{R}$ such as

$$f(\mathbf{x}) = \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{a}^T\mathbf{x} + b$$

Our purpose being to minimise the MSE over a training-set X_{train}, we seek to minimize:

$$\min_{\mathbf{Q},\mathbf{a},b} \sum_{(\mathbf{x},\mathbf{y})\in(X_{\text{train}},Y_{\text{train}})} \|\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{x}^T\mathbf{a} + b - \mathbf{y}\|^2 \qquad (20)$$

We also have that

$$\mathbf{x}^T\mathbf{Q}\mathbf{x} = \text{trace}\left(\mathbf{Q}\mathbf{x}\mathbf{x}^T\right)$$

Denoting by $\mathbf{q}$ the column-stacking of $\mathbf{Q}$, the above expression can be expressed as:

$$\mathbf{x}^T\mathbf{Q}\mathbf{x} = \text{trace}\left(\mathbf{Q}\mathbf{x}\mathbf{x}^T\right) = \mathbf{q}\left(\mathbf{x}\otimes\mathbf{x}\right)$$

where $\otimes$ denotes the Kronecker product. Hence, equation 20 can be expressed as:

$$\min_{\mathbf{q},\mathbf{a},b} \sum_{(\mathbf{x},y)\in(X_{\text{train}},Y_{\text{train}})} \|\left(\mathbf{x},\mathbf{x}\otimes\mathbf{x}\right)^T\left(\mathbf{a},\mathbf{q}\right) + b - y\|^2. \qquad (21)$$

Denoting by $\tilde{\mathbf{x}} = (\mathbf{x}, \mathbf{x}\otimes\mathbf{x})$ and $\mathbf{v} = (\mathbf{a}, \mathbf{q})$, we are led to a simple regression problem:

$$\min_{\mathbf{v},b} \sum_{(\tilde{\mathbf{x}},\mathbf{y})\in(\tilde{X}_{\text{train}},Y_{\text{train}})} \|\tilde{\mathbf{x}}^T\mathbf{v}+b-y\|^2. \tag{22}$$

We rewrite the objective function of equation 22 as:

$$(\tilde{\mathbf{x}}^T\mathbf{v}+b-\mathbf{y})^T(\tilde{\mathbf{x}}^T\mathbf{v}+b-y) = \mathbf{v}^T\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\mathbf{v}+2(b-y)\tilde{\mathbf{x}}^T\mathbf{v}+(b-y)^2$$

Stacking the $\tilde{\mathbf{x}}, \mathbf{y}$ in matrices $\tilde{\mathbf{X}}, \mathbf{Y}$, the above expression can be rewritten as:

$$\mathbf{v}^T\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\mathbf{v}+2(b\mathbb{1}-\mathbf{Y})^T\tilde{\mathbf{X}}^T\mathbf{v}+(b\mathbb{1}-\mathbf{Y})^T(b\mathbb{1}-\mathbf{Y}) \tag{23}$$

Deriving with respect to b leads to: $2\mathbb{1}^T\tilde{\mathbf{X}}^T\mathbf{v}+2nb-2\mathbb{1}^T\mathbf{Y}=0$ Hence:

$$b=\frac{1}{n}\left(\mathbb{1}^T(\mathbf{Y}-\tilde{\mathbf{X}}^T\mathbf{v})\right)=\frac{1}{n}\left((\mathbf{Y}-\tilde{\mathbf{X}}^T\mathbf{v})^T\mathbb{1}\right)$$

We hence have

$$\begin{aligned}(b\mathbb{1}-\mathbf{Y})^T(b\mathbb{1}-\mathbf{Y}) &= nb^2-2b\mathbb{1}^T\mathbf{Y}+\mathbf{Y}^T\mathbf{Y}\\ &= \frac{1}{n}\left(\mathbf{v}^T\tilde{\mathbf{X}}\mathbb{1}\mathbb{1}^T\tilde{\mathbf{X}}^T\mathbf{v}-2\mathbf{Y}^T\mathbb{1}\mathbb{1}^T\tilde{\mathbf{X}}^T\mathbf{v}+2\mathbf{Y}^T\mathbb{1}\mathbb{1}^{T\tilde{T}}\mathbf{v}\right)\\ &= \frac{1}{n}\mathbf{v}^T\tilde{\mathbf{X}}\mathbb{1}\mathbb{1}^T\tilde{\mathbf{X}}^T\mathbf{v}\end{aligned}$$

In addition:

$$(b\mathbb{1}-\mathbf{Y})^T\tilde{\mathbf{X}}^T\mathbf{v}=\frac{1}{n}\left(\mathbf{Y}^T\mathbb{1}\mathbb{1}^T\tilde{\mathbf{X}}^T\mathbf{v}-\mathbf{v}^T\tilde{\mathbf{X}}\mathbb{1}\mathbb{1}^T\tilde{\mathbf{X}}^T\mathbf{v}\right)-\mathbf{Y}^T\tilde{\mathbf{X}}^T\mathbf{v}$$

Hence, equation 23 can be rewritten as:

$$\mathbf{v}^T\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\mathbf{v}-\frac{1}{n}\mathbf{v}^T\tilde{\mathbf{X}}\mathbb{1}\mathbb{1}^T\tilde{\mathbf{X}}^T\mathbf{v}+\mathbf{Y}^T\left(\mathbf{Id}-\frac{1}{n}\mathbb{1}\mathbb{1}^T\right)\tilde{\mathbf{X}}^T\mathbf{v}$$

Denoting by $\mathbf{I}=\left(\mathbf{Id}-\frac{1}{n}\mathbb{1}\mathbb{1}^T\right)$, $\hat{\mathbf{X}}=\mathbf{I}\tilde{\mathbf{X}}^T$ and by $\hat{\mathbf{Y}}=\mathbf{I}\mathbf{Y}$, and noticing that $\mathbf{I}^T\mathbf{I}=\mathbf{I}$, we then have:

$$\hat{\mathbf{X}}^T\hat{\mathbf{X}}v=\hat{\mathbf{X}}^T\hat{\mathbf{Y}}$$

To solve this problem we can find am SVD decomposition of $\hat{\mathbf{X}}=\mathbf{U}\mathbf{D}\mathbf{V}^T$, hence:

$$\mathbf{V}\mathbf{D}^2\mathbf{V}^Tv=\mathbf{V}\mathbf{D}\mathbf{U}^T\hat{\mathbf{Y}}$$

that leads to:

$$v=\mathbf{V}\mathbf{D}^{-1}\mathbf{U}^T\hat{\mathbf{Y}}$$

The general algorithm to find the decomposition is the following:

In order to choose the number k of principal components described in the above algorithm, we can perform a simple hyper parameter search using a test set. In the below figure, we plot the Kendall-Tau coefficient and MSE of a quadratic predictor trained using a closed form regularized solution of the regression problem as a function of the number of principal components k, both on test and validation set. We can see that above 2500 components, we reach a saturation that leads to a higher error due to an over-fitting on the training set. Using 1500 components leads to a better generalization. The above scheme is

Algorithm 2 Closed Form Solution of a Linear Regression for the Quadratic Predictors

```
input  $\{x_i = (\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i) \in \mathbb{R}^n, y_i = Acc(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i)\}_{i=1}^N, k =$ number of principal components
1: Compute $\tilde{x}_i = (x_i, x_i \otimes x_i),\ \forall i \in \{1, \ldots, N\}$
2: Perform a centering on $\tilde{x}_i$ computing $\hat{x}_i = \tilde{x}_i - \text{mean}_{i=1,\ldots,N}(\tilde{x}_i),\ \forall i \in \{1, \ldots, N\}$
3: Perform a centering on $y_i$ computing $\hat{y}_i = y_i - \text{mean}_{i=1,\ldots,N}(y_i),\ \forall i \in \{1, \ldots, N\}$
4: Define $\hat{X} = \text{stack}(\{\hat{x}_i\}_{i=1}^N)$
5: Compute a $k$-low rank SVD decomposition of $\hat{X}$, defined as $U\,\text{diag}(s)V^T$
6: Compute $W = V\,\text{diag}(s^{-1})U^T\hat{y}$
7: Compute $b = \text{mean}\left(y - \tilde{X}W\right)$
8: Define $a = W_{1:n}$
9: Reshape the end of the vector $W$ as an $n \times n$ matrix, $Q = \text{reshape}(W_{n+1:n+1+n^2}, n, n)$
output  $b, a, Q$
```

another way to regularize a regression and, unlike Ridge Regression, can be used to solve problems of very a high dimesionality without the need to find the pseudo inverse of a high dimensional matrix, without using any optimization method, and with a relatively robust discrete unidimensional parameter that is easier to tune.

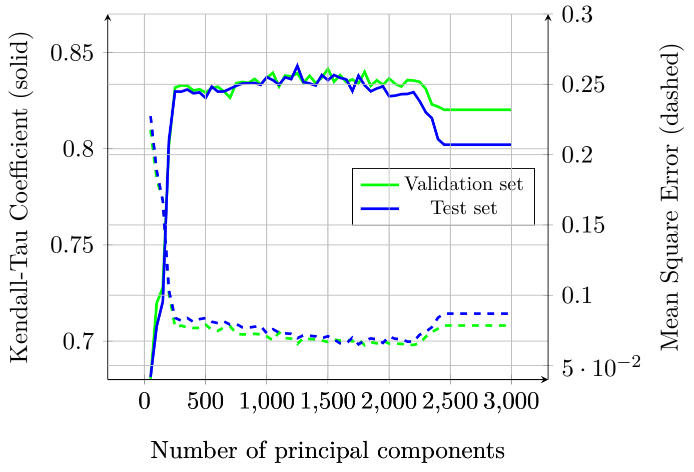


Figure 11: Kendall-Tau correlation coefficients and MSE of different predictors vs number of principal components

Appendix G. Convergence Guarantees for Solving BLCP with BCFW with Line-Search

In this section we proof Theorem 3.2, guaranteeing that after $\mathcal{O}(1/\epsilon)$ many iterations, Algorithm 1 obtains an ϵ-approximate solution to problem 1.

G.1. Convergence Guarantees for a General BCFW over a Product Domain

The proof is heavily based on the convergence guarantees provided by Lacoste-Julien et al. (2013) for solving:

$$\min_{\zeta \in \mathcal{M}^{(1)} \times \cdots \times \mathcal{M}^{(n)}} f(\zeta) \tag{24}$$

with the BCFW algorithm 3, where $\mathcal{M}^{(i)} \subset \mathbb{R}^{m_i}$ is the convex and compact domain of the i-th coordinate block and the product $\mathcal{M}^{(1)} \times \cdots \times \mathcal{M}^{(n)} \subset \mathbb{R}^m$ specifies the whole domain, as $\sum_{i=1}^{n} m_i = m$. $\zeta^{(i)} \in \mathbb{R}^{m_i}$ is the i-th coordinate block of ζ and $\zeta^{\setminus(i)}$ is the rest of the coordinates of ζ. $\nabla^{(i)}$ stands for the partial derivatives vector with respect to the i-th coordinate block.

Algorithm 3 Block Coordinate Frank-Wolfe (BCFW) on Product Domain

```
input $\zeta_0 \in \mathcal{M}^{(1)} \times \cdots \times \mathcal{M}^{(n)} \subset \mathbb{R}^m$
1: for $k = 0, \dots, K$ do
2:     Pick $i$ at random in $\{1, \dots, n\}$
3:     Find $s_k = \text{argmin}_{s \in \mathcal{M}^{(i)}} s^T \cdot \nabla^{(i)} f(\zeta_k)$
4:     Let $\tilde{s}_k =: 0_m \in \mathbb{R}^m$ is the zero padding of $s_k$ such that we then assign $\tilde{s}_k^{(i)} := s_k$
5:     Let $\gamma := \frac{2n}{k+2n}$,
       or perform line-search: $\gamma =: \text{argmin}_{\gamma' \in [0,1]} f\left((1-\gamma') \cdot \zeta_k + \gamma' \cdot \tilde{s}_k\right)$
6:     Update $\zeta_{k+1} = (1-\gamma) \cdot \zeta_k + \gamma \cdot \tilde{s}_k$
7: end for
```

The following theorem shows that after $\mathcal{O}(1/\epsilon)$ many iterations, Algorithm 3 obtains an ϵ-approximate solution to problem 24, and guaranteed ϵ-small duality gap.

Theorem G.1 *For each $k > 0$ the iterate ζ_k Algorithm 3 satisfies:*

$$E[f(\zeta_k)] - f(\zeta^*) \leq \frac{2n}{k+2n}\left(C_f^{\otimes} + (f(\zeta_0) - f(\zeta^*))\right)$$

where ζ^* is the solution of problem 24 and the expectation is over the random choice of the block i in the steps of the algorithm.

Furthermore, there exists an iterate $0 \leq \hat{k} \leq K$ of Algorithm 3 with a duality gap bounded by $E[g(\zeta_{\hat{k}})] \leq \frac{6n}{K+1}\left(C_f^{\otimes} + (f(\zeta_0) - f(\zeta^*))\right)$.

Here the duality gap $g(\zeta) \geq f(\zeta) - f(\zeta^*)$ is defined as following:

$$g(\zeta) = \max_{s \in \mathcal{M}^{(1)} \times \cdots \times \mathcal{M}^{(n)}} (\zeta - s)^T \cdot \nabla f(\zeta) \tag{25}$$

and the *global product curvature* constant $C_f^{\otimes} = \sum_{i=1}^n C_f^{(i)}$ is the sum of the (partial) curvature constants of f with respect to the individual domain $\mathcal{M}^{(i)}$:

$$C_f^{(i)} = \sup_{\substack{x \in \mathcal{M}^{(1)} \times \cdots \times \mathcal{M}^{(n)}, \\ s^{(i)} \in \mathcal{M}^{(i)}, \gamma \in [0,1], \\ y^{(i)} = (1-\gamma)x^{(i)} + \gamma s^{(i)}, \\ y^{\setminus(i)} = x^{\setminus(i)}}} \frac{2}{\gamma^2} \left(f(y) - f(x) - \left(y^{(i)} - x^{(i)} \right)^T \nabla^{(i)} f(x) \right) \tag{26}$$

which quantifies the maximum relative deviation of the objective function f from its linear approximations, over the domain $\mathcal{M}^{(i)}$.

The proof of theorem G.1 is given in Lacoste-Julien et al. (2013).

G.2. Analytic Line-Search for Bilinear Objective Functions

The following theorem provides a trivial analytic solution for the line-search of algorithm 3 (line 5) where the objective function has a bilinear form.

Theorem G.2 *The analytic solution of the line-search step of algorithm 3 (line 5) with a bilinear objective function of the form:*

$$f(\zeta) = \sum_{i=1}^n \left(\zeta^{(i)}\right)^T \cdot p_f^{(i)} + \sum_{i=1}^n \sum_{j=i}^n \left(\zeta^{(i)}\right)^T \cdot Q_f^{(i,j)} \cdot \zeta^{(j)} \tag{27}$$

with $p_f^{(i)} \in \mathbb{R}^{m_i}$ and $Q_f^{(i,j)} \in \mathbb{R}^{m_i \times m_j}$, reads $\gamma \equiv 1$ at all the iterations.

Proof In each step of algorithm 3 at line 3, a linear program is solved:

$$\begin{aligned} s &= \underset{s' \in \mathcal{M}^{(i)}}{\operatorname{argmin}} \ \nabla^{(i)} f(\zeta)^T \cdot s' \\ &= \underset{s' \in \mathcal{M}^{(i)}}{\operatorname{argmin}} \left(\left(p_f^{(i)}\right)^T + \sum_{j=1}^{i-1} \left(\zeta^{(j)}\right)^T \cdot Q_f^{(i,j)} + \sum_{j=i+1}^{n} \left(\zeta^{(j)}\right)^T \cdot \left(Q_f^{(i,j)}\right)^T \right) \cdot s' \end{aligned} \tag{28}$$

and the Line-Search at line 5 reads:

$$\begin{aligned} \gamma &=: \underset{\gamma' \in [0,1]}{\operatorname{argmin}} \ f\left((1-\gamma') \cdot \zeta + \gamma' \cdot \tilde{s}\right) \\ &= \underset{\substack{\gamma' \in [0,1] \\ y = (1-\gamma') \cdot \zeta^{(i)} + \gamma' \cdot s}}{\operatorname{argmin}} \begin{pmatrix} \left(p_f^{(i)}\right)^T + \sum_{j=1}^{i-1} \left(\zeta^{(j)}\right)^T \cdot Q_f^{(i,j)} \\ + \sum_{j=i+1}^{n} \left(\zeta^{(j)}\right)^T \cdot \left(Q_f^{(i,j)}\right)^T \end{pmatrix} \cdot y \\ &= \underset{\substack{\gamma' \in [0,1] \\ y = (1-\gamma') \cdot \zeta^{(i)} + \gamma' \cdot s}}{\operatorname{argmin}} \nabla^{(i)} f(\zeta)^T \cdot y \end{aligned} \tag{29}$$

Since $\zeta^{(i)}, s \in \mathcal{M}^{(i)}$ and $\gamma \in [0,1]$ then the convex combination of those also satisfies $y \in \mathcal{M}^{(i)}$, hence considering that s is the optimizer of 28 the solution to 29 reads $y := s$ and hence $\gamma := 1$. Thus, effectively the analytic solution to line-search for a bilinear objective function is $\gamma \equiv 1$ at all times. ■

G.3. Solving BLCP by BCFW with Line-Search

In addition to a bilinear objective function as in equation 27, consider also a domain that is specified by the following bilinear constraints:

$$\sum_{i=1}^{n}\left(\zeta^{(i)}\right)^{T}\cdot p_{\mathcal{M}}^{(i)}+\sum_{i=1}^{n}\sum_{j=i}^{n}\left(\zeta^{(i)}\right)^{T}\cdot Q_{\mathcal{M}}^{(i,j)}\cdot\zeta^{(j)}\leq T \quad ; \quad A\cdot\zeta\leq b \tag{30}$$

with $p_{\mathcal{M}}^{(i)}\in\mathbb{R}^{m_i}$, $Q_{\mathcal{M}}^{(i,j)}\in\mathbb{R}^{m_i\times m_j}$, $A\in\mathbb{R}^{C\times m}$ and $b\in\mathbb{R}^{C}$ for $C\leq 0$, such that the individual domain of the i-th coordinate block is specified by the following linear constraints:

$$\left(\left(p_{\mathcal{M}}^{(i)}\right)^{T}+\sum_{j=1}^{i-1}\left(\zeta^{(j)}\right)^{T}\cdot Q_{\mathcal{M}}^{(i,j)}+\sum_{j=i+1}^{n}\left(\zeta^{(j)}\right)^{T}\cdot\left(Q_{\mathcal{M}}^{(i,j)}\right)^{T}\right)\cdot\zeta^{(i)}\leq T \tag{31}$$

$$A^{(i)}\cdot\zeta^{(i)}\leq b^{(i)} \tag{32}$$

where $A^{(i)}\in\mathbb{R}^{C\times m_i}$ are the rows $r\in\{1+\sum_{j<i}m_j,\ldots,\sum_{j\leq i}m_j\}$ of A and $b^{(i)}\in\mathbb{R}_i^{m}$ are the corresponding elements of b.

Thus in each step of algorithm 3 at line 3, a linear program is solved:

$$\begin{aligned}
\min_{\zeta^{(i)}} & \left(\left(p_{f}^{(i)}\right)^{T}+\sum_{j=1}^{i-1}\left(\zeta^{(j)}\right)^{T}\cdot Q_{f}^{(i,j)}+\sum_{j=i+1}^{n}\left(\zeta^{(j)}\right)^{T}\cdot\left(Q_{f}^{(i,j)}\right)^{T}\right)\cdot\zeta^{(i)}\\
s.t. & \left(\left(p_{\mathcal{M}}^{(i)}\right)^{T}+\sum_{j=1}^{i-1}\left(\zeta^{(j)}\right)^{T}\cdot Q_{\mathcal{M}}^{(i,j)}+\sum_{j=i+1}^{n}\left(\zeta^{(j)}\right)^{T}\cdot\left(Q_{\mathcal{M}}^{(i,j)}\right)^{T}\right)\cdot\zeta^{(i)}\leq T\\
& A^{(i)}\cdot\zeta^{(i)}\leq b^{(i)}
\end{aligned}$$

And thus equipped with theorem G.2, algorithm 4 provides a more specific version of algorithm 3 for solving BLCP.

In section 3, we deal with $n=2$ blocks where $\zeta=(\boldsymbol{\alpha},\boldsymbol{\beta})$ such that:

$$\begin{array}{lllllll}
\zeta^{(1)}=\boldsymbol{\alpha} & m_1=D\cdot S\cdot|\mathcal{C}| & p_f^{(1)}=p_\alpha & p_{\mathcal{M}}^{(1)}=0 & A^{(1)}=A_{\mathcal{S}}^{\alpha} & b^{(1)}=b_{\mathcal{S}}^{\alpha} & Q_f^{(1,2)}=Q_{\alpha\beta}\\
\zeta^{(2)}=\boldsymbol{\beta} & m_2=D\cdot S & p_f^{(2)}=p_\beta & p_{\mathcal{M}}^{(2)}=0 & A^{(2)}=A_{\mathcal{S}}^{\beta} & b^{(2)}=b_{\mathcal{S}}^{\beta} & Q_{\mathcal{M}}^{(1,2)}=\Theta
\end{array} \tag{33}$$

Thus for this particular case of interest algorithm 4 effectively boils down to algorithm 1.

G.3.1. Proof of Theorem 1

Let us first compute the curvature constants $C_f^{(i)}$ (equation 26) and $C_f^{\otimes}$ for the bilinear objective function as in equation 27.

Algorithm 4 BCFW with Line-Search on QCQP Product Domain

```
input $\zeta_0 \in \{\zeta \mid \sum_{i=1}^{n} \left(\zeta^{(i)}\right)^T \cdot p_{\mathcal{M}}^{(i)} + \sum_{i=1}^{n}\sum_{j=i}^{n} \left(\zeta^{(i)}\right)^T \cdot Q_{\mathcal{M}}^{(i,j)} \cdot \zeta^{(j)} \leq T \quad ; \quad A \cdot \zeta \leq b\}$
1: for $k = 0, \dots, K$ do
2:     Pick $i$ at random in $\{1, \dots, n\}$
3:     Keep the same values for all other coordinate blocks $\zeta_{k+1}^{\setminus(i)} = \zeta_k^{\setminus(i)}$ and update:
       $$\zeta_{k+1}^{(i)} = \underset{s}{\text{argmin}} \left( \left(p_f^{(i)}\right)^T + \sum_{j=1}^{i-1} \left(\zeta^{(j)}\right)^T \cdot Q_f^{(i,j)} + \sum_{j=i+1}^{n} \left(\zeta^{(j)}\right)^T \cdot \left(Q_f^{(i,j)}\right)^T \right) \cdot s$$
       $$s.t. \left( \left(p_{\mathcal{M}}^{(i)}\right)^T + \sum_{j=1}^{i-1} \left(\zeta^{(j)}\right)^T \cdot Q_{\mathcal{M}}^{(i,j)} + \sum_{j=i+1}^{n} \left(\zeta^{(j)}\right)^T \cdot \left(Q_{\mathcal{M}}^{(i,j)}\right)^T \right) \cdot s \leq T$$
       $$A^{(i)} \cdot s \leq b^{(i)}$$
4: end for
```

Lemma G.3 *Let f have a bilinear form, such that:*
$f(x) = \sum_{i=1}^{n} \left(x^{(i)}\right)^T \cdot p_f^{(i)} + \sum_{i=1}^{n}\sum_{j=i}^{n} \left(x^{(i)}\right)^T \cdot Q_f^{(i,j)} \cdot x^{(j)}$ *then* $C_f^{\otimes} = 0$.

Proof Separating the i-th coordinate block:

$$f(x) = \sum_{l=1}^{n} \left(x^{(l)}\right)^T \cdot p_f^{(l)} + \sum_{l=1}^{n}\sum_{j=l}^{n} \left(x^{(l)}\right)^T \cdot Q_f^{(l,j)} \cdot x^{(j)} \tag{34}$$

$$= \left(x^{(i)}\right)^T \cdot p_f^{(i)} + \sum_{j=1}^{i-1} \left(x^{(j)}\right)^T \cdot Q_f^{(i,j)} \cdot x^{(i)} + \sum_{j=i+1}^{n} x^{(i)} \cdot Q_f^{(i,j)} \cdot x^{(j)} \tag{35}$$

$$+ \sum_{l=1}^{n} \mathbb{1}_{l \neq i} \left(x^{(l)}\right)^T \cdot p_f^{(l)} + \sum_{l=1}^{n}\sum_{j=l}^{n} \mathbb{1}_{l \neq i} \cdot \mathbb{1}_{j \neq i} \left(x^{(l)}\right)^T \cdot Q_f^{(l,j)} \cdot x^{(j)} \tag{36}$$

where $\mathbb{1}_A$ is the indicator function that yields 1 if A holds and 0 otherwise.

Thus for y with $y^{(i)} = (1-\gamma)x^{(i)} + \gamma s^{(i)}$ and $y^{\setminus(i)} = x^{\setminus(i)}$, we have:

$$f(y) = \left(y^{(i)}\right)^T \cdot p_f^{(i)} + \sum_{j=1}^{i-1} \left(y^{(j)}\right)^T \cdot Q_f^{(i,j)} \cdot y^{(i)} + \sum_{j=i+1}^{n} y^{(i)} \cdot Q_f^{(i,j)} \cdot y^{(j)} \tag{37}$$

$$+ \sum_{l=1}^{n} \mathbb{1}_{l \neq i} \left(y^{(l)}\right)^T \cdot p_f^{(l)} + \sum_{l=1}^{n}\sum_{j=l}^{n} \mathbb{1}_{l \neq i} \cdot \mathbb{1}_{j \neq i} \left(x^{(l)}\right)^T \cdot Q_f^{(l,j)} \cdot y^{(j)} \tag{38}$$

$$= \left(y^{(i)}\right)^T \cdot p_f^{(i)} + \sum_{j=1}^{i-1} \left(x^{(j)}\right)^T \cdot Q_f^{(i,j)} \cdot y^{(i)} + \sum_{j=i+1}^{n} y^{(i)} \cdot Q_f^{(i,j)} \cdot x^{(j)} \tag{39}$$

$$+ \sum_{l=1}^{n} \mathbb{1}_{l \neq i} \left(x^{(l)}\right)^T \cdot p_f^{(l)} + \sum_{l=1}^{n}\sum_{j=l}^{n} \mathbb{1}_{l \neq i} \cdot \mathbb{1}_{j \neq i} \left(x^{(l)}\right)^T \cdot Q_f^{(l,j)} \cdot x^{(j)} \tag{40}$$

Hence,

$$f(y) - f(x) = \nabla^{(i)} f(x) \cdot \left(y^{(i)} - x^{(i)}\right) \tag{41}$$

since 36 and 40 cancel out and,

$$\nabla^{(i)} f(x) = \left(\left(p_f^{(i)}\right)^T + \sum_{j=1}^{i-1} \left(x^{(j)}\right)^T \cdot Q_f^{(i,j)} + \sum_{j=i+1}^{n} \left(x^{(j)}\right)^T \cdot \left(Q_f^{(i,j)}\right)^T \right) \tag{42}$$

Hence we have,

$$C_f^{(i)} = 0 \quad \forall i \in \{1, \dots, n\} \qquad ; \qquad C_f^{\otimes} = \sum_{i=1}^{n} C_f^{(i)} = 0 \tag{43}$$

■

Thus for a bilinear objective function, theorem G.1 boils down to:

Theorem G.4 *For each $k > 0$ the iterate ζ_k Algorithm 4 satisfies:*

$$E[f(\zeta_k)] - f(\zeta^*) \leq \frac{2n}{k + 2n} \left(f(\zeta_0) - f(\zeta^*)\right)$$

where ζ^* is the solution of problem 24 and the expectation is over the random choice of the block i in the steps of the algorithm. Furthermore, there exists an iterate $0 \leq \hat{k} \leq K$ of Algorithm 4 with a duality gap bounded by $E[g(\zeta_{\hat{k}})] \leq \frac{6n}{K+1} \left(f(\zeta_0) - f(\zeta^*)\right)$.

And by setting $n = 2$ with equations 33 for $f(\zeta) := ACC(\zeta)$, theorem 3.2 follows.

Appendix H. Sparsity Guarantees for Solving BLCP with BCFW with Line-Search

In order to proof 3.3, we start with providing auxiliary lemmas proven at Nayman et al. (2021). To this end we define the *relaxed* Multiple Choice Knapsack Problem (MCKP):

Definition H.1 *Given $n \in \mathbb{N}$, and a collection of k distinct covering subsets of $\{1, 2, \cdots, n\}$ denoted as $N_i, i \in \{1, 2, \cdots, k\}$, such that $\cup_{i=1}^{k} N_i = \{1, 2, \cdots, n\}$ and $\cap_{i=1}^{k} N_i = \varnothing$ with associated values and costs p_{ij}, t_{ij} $\forall i \in \{1, \dots, k\}, j \in N_i$ respectively, the relaxed Multiple Choice Knapsack Problem (MCKP) is formulated as following:*

$$\begin{aligned} \max_{vu} \; & \sum_{i=1}^{k} \sum_{j \in N_i} p_{ij} \boldsymbol{u}_{ij} \\ s.t. \; & \sum_{i=1}^{k} \sum_{j \in N_i} t_{ij} \boldsymbol{u}_{ij} \leq T \\ & \sum_{j \in N_i} \boldsymbol{u}_{ij} = 1 \qquad \forall i \in \{1, \dots, k\} \\ & \boldsymbol{u}_{ij} \geq 0 \qquad \forall i \in \{1, \dots, k\}, j \in N_i \end{aligned} \tag{44}$$

where the binary constraints $\boldsymbol{u}_{ij} \in \{0, 1\}$ of the original MCKP formulation Kellerer et al. (2004) are replaced with $\boldsymbol{u}_{ij} \geq 0$.

Definition H.2 *An one-hot vector $\boldsymbol{u}_i$ satisfies:*

$$||\boldsymbol{u}_i^*||^0 = \sum_{j \in N_i} |\boldsymbol{u}_{ij}^*|^0 = \sum_{j \in N_i} \mathbb{1}_{\boldsymbol{u}_{ij}^* > 0} = 1$$

where $\mathbb{1}_A$ is the indicator function that yields 1 if A holds and 0 otherwise.

Lemma H.1 *The solution $\mathbf{u}^*$ of the relaxed MCKP equation 44 is composed of vectors $\mathbf{u}_i^*$ that are all one-hot but a single one.*

Lemma H.2 *The single non one-hot vector of the solution $\mathbf{u}^*$ of the relaxed MCKP equation 44 has at most two nonzero elements.*

See the proofs for Lemmas H.1 and H.1 in Nayman et al. (2021) (Appendix F).

In order to prove Theorem 3.3, we use Lemmas H.1 and H.1 for each coordinate block $\zeta^{(i)}$ for $i \in \{1, \dots, n\}$ separately, based on the observation that at every iteration $k = 0, \dots, K$ of algorithm 4, each sub-problem (lines 3,5) forms a *relaxed* MCKP equation 44. Thus replacing

- $\boldsymbol{u}$ in equation 44 with $\zeta^{(i)}$.
- p with $\left(p_f^{(i)}\right)^T + \sum_{j=1}^{i-1} \left(\zeta^{(j)}\right)^T \cdot Q_f^{(i,j)} + \sum_{j=i+1}^{n} \left(\zeta^{(j)}\right)^T \cdot \left(Q_f^{(i,j)}\right)^T$
- The elements of t with the elements of

$$\left(p_{\mathcal{M}}^{(i)}\right)^T + \sum_{j=1}^{i-1} \left(\zeta^{(j)}\right)^T \cdot Q_{\mathcal{M}}^{(i,j)} + \sum_{j=i+1}^{n} \left(\zeta^{(j)}\right)^T \cdot \left(Q_{\mathcal{M}}^{(i,j)}\right)^T$$

- The simplex constraints with the linear inequality constraints specified by $A^{(i)}, b^{(i)}$.

Hence for every iteration theorem 3.3 holds and in particular for the last iteration $k = K$ which is the output of solution of algorithm 4.

By setting $n = 2$ with equations 33 for $f(\zeta) := ACC(\zeta)$, algorithm 4 boils down to algorithm 1 and thus theorem 3.3 holds for the later as special case of the former.

Appendix I. On the Transitivity of Ranking Correlations

While the predictors in section 5.2 yields high ranking correlation between the predicted accuracy and the accuracy measured for a subnetwork of a given one-shot model, the ultimate ranking correlation is with respect to the same architecture trained as a standalone from scratch. Hence we are interested also in the transitivity of ranking correlation. Langford et al. (2001) provides such transitivity property of the Pearson correlation between random variables P, O, S standing for the predicted, the one-shot and the standalone accuracy respectively:

$$|Cor(P,S) - Cor(P,O) \cdot Cor(O,S)| \leq \sqrt{(1 - Cor(P,O)^2) \cdot (1 - Cor(O,S)^2)}$$

This is also true for the Spearman correlation as a Pearson correlation over the corresponding ranking. Hence, while the accuracy estimator can be efficiently acquired for any given one-shot model, the quality of this one-shot model contributes its part to the overall ranking correlation. In this paper we use the official one-shot model provided by Nayman et al. (2021) with a reported Spearman correlation of $\rho_{P,O} = 0.99$ to the standalone networks. Thus together with the Spearman correlation of $\rho_{O,S} = 0.97$ of the proposed accuracy estimator, the overall Spearman ranking correlation satisfies $\rho_{P,S} \geq 0.93$.

Appendix J. Averaging Individual Accuracy Contribution for Interpretability

In this section we provide the technical details about the way we generate the results presented in section 5.3, Figure 8 (Middle and Right).

We measure the average contribution of adding a S&E layer at each stage s by:

$$\frac{1}{D}\sum_{b=1}^{D}\left(\frac{1}{|\mathcal{C}_{se}|}\sum_{c\in\mathcal{C}_{se}}\Delta_{b,c}^{s} - \frac{1}{|\bar{\mathcal{C}}_{se}|}\sum_{c\in\bar{\mathcal{C}}_{se}}\Delta_{b,c}^{s}\right)$$

and each block b by:

$$\frac{1}{S}\sum_{s=1}^{S}\left(\frac{1}{|\mathcal{C}_{se}|}\sum_{c\in\mathcal{C}_{se}}\Delta_{b,c}^{s} - \frac{1}{|\bar{\mathcal{C}}_{se}|}\sum_{c\in\bar{\mathcal{C}}_{se}}\Delta_{b,c}^{s}\right)$$

where $\mathcal{C}_{se} = \{2,4,6,8,10,12\}$ ($|\mathcal{C}_{se}| = 6$) are all the configurations that include S&E layers according to Table 3 (Right) and its complementary set $\bar{\mathcal{C}}_{se} = \{1,3,5,7,9,11\}$ ($|\bar{\mathcal{C}}_{se}| = 6$).

Similarly the average contribution of increasing the kernel size from 3×3 to 5×5 at each stage s by:

$$\frac{1}{D}\sum_{b=1}^{D}\left(\frac{1}{|\mathcal{C}_{5\times5}|}\sum_{c\in\mathcal{C}_{5\times5}}\Delta_{b,c}^{s} - \frac{1}{|\mathcal{C}_{3\times3}|}\sum_{c\in\mathcal{C}_{3\times3}}\Delta_{b,c}^{s}\right)$$

and each block b by:

$$\frac{1}{S}\sum_{s=1}^{S}\left(\frac{1}{|\mathcal{C}_{5\times5}|}\sum_{c\in\mathcal{C}_{5\times5}}\Delta_{b,c}^{s} - \frac{1}{|\mathcal{C}_{3\times3}|}\sum_{c\in\mathcal{C}_{3\times3}}\Delta_{b,c}^{s}\right)$$

where $\mathcal{C}_{3\times3} = \{1,2,5,6,9,10\}$ ($|\mathcal{C}_{3\times3}| = 6$) and $\mathcal{C}_{5\times5} = \{3,4,7,8,11,12\}$ ($|\mathcal{C}_{5\times5}| = 6$) are all the configurations with a kernel size of 3×3 and 5×5 respectively according to Table 3 (Right).

The average contribution of increasing the expansion ratio from 2 to 3 at each stage s by:

$$\frac{1}{D}\sum_{b=1}^{D}\left(\frac{1}{|\mathcal{C}_{3}|}\sum_{c\in\mathcal{C}_{3}}\Delta_{b,c}^{s} - \frac{1}{|\mathcal{C}_{2}|}\sum_{c\in\mathcal{C}_{2}}\Delta_{b,c}^{s}\right)$$

and each block b by:

$$\frac{1}{S}\sum_{s=1}^{S}\left(\frac{1}{|\mathcal{C}_{3}|}\sum_{c\in\mathcal{C}_{3}}\Delta_{b,c}^{s} - \frac{1}{|\mathcal{C}_{2}|}\sum_{c\in\mathcal{C}_{2}}\Delta_{b,c}^{s}\right)$$

where $\mathcal{C}_{3} = \{5,6,7,8\}$ ($|\mathcal{C}_{3}| = 4$) and $\mathcal{C}_{2} = \{1,2,3,4\}$ ($|\mathcal{C}_{2}| = 4$) are all the configurations with an expansion ratio of 3 and 2 respectively according to Table 3 (Right).

Finally, the average contribution of increasing the expansion ratio from 3 to 6 at each stage s by:

$$\frac{1}{D}\sum_{b=1}^{D}\left(\frac{1}{|\mathcal{C}_6|}\sum_{c\in\mathcal{C}_6}\Delta_{b,c}^{s}-\frac{1}{|\mathcal{C}_3|}\sum_{c\in\mathcal{C}_3}\Delta_{b,c}^{s}\right)$$

and each block b by:

$$\frac{1}{S}\sum_{s=1}^{S}\left(\frac{1}{|\mathcal{C}_6|}\sum_{c\in\mathcal{C}_6}\Delta_{b,c}^{s}-\frac{1}{|\mathcal{C}_3|}\sum_{c\in\mathcal{C}_3}\Delta_{b,c}^{s}\right)$$

where $\mathcal{C}_6=\{9,10,11,12\}$ ($|\mathcal{C}_6|=4$) are all the configurations with an expansion ratio of 6 according to Table 3 (Right).

Appendix K. Extended Figures

Due to space limit, here we present extended figures to better describe the estimation of individual accuracy contribution terms described in section 3.2.

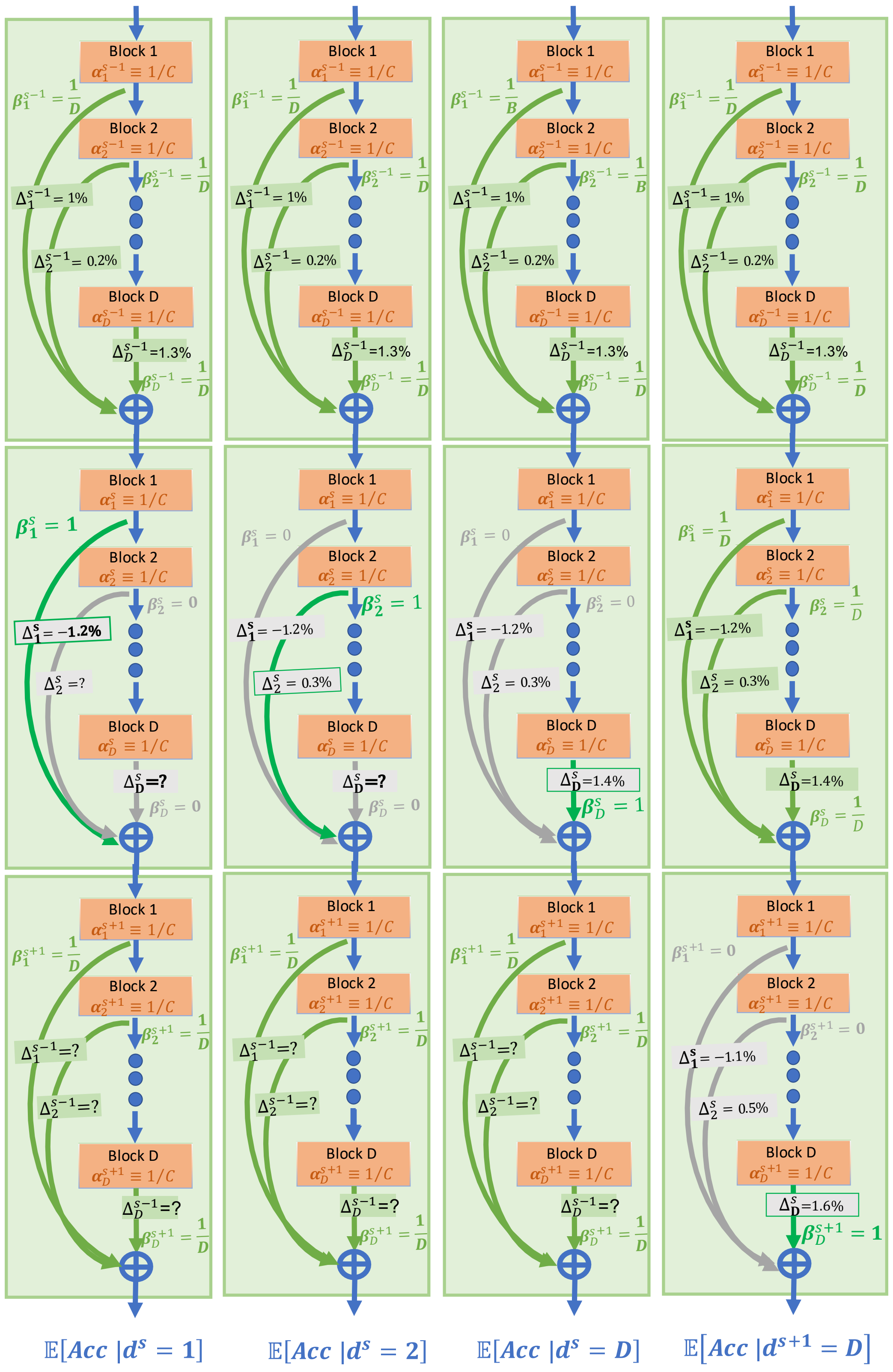


Figure 12: Extension of Figure 4: Estimating the expected accuracy gap caused by macroscopic design choices of the depth of the stages.

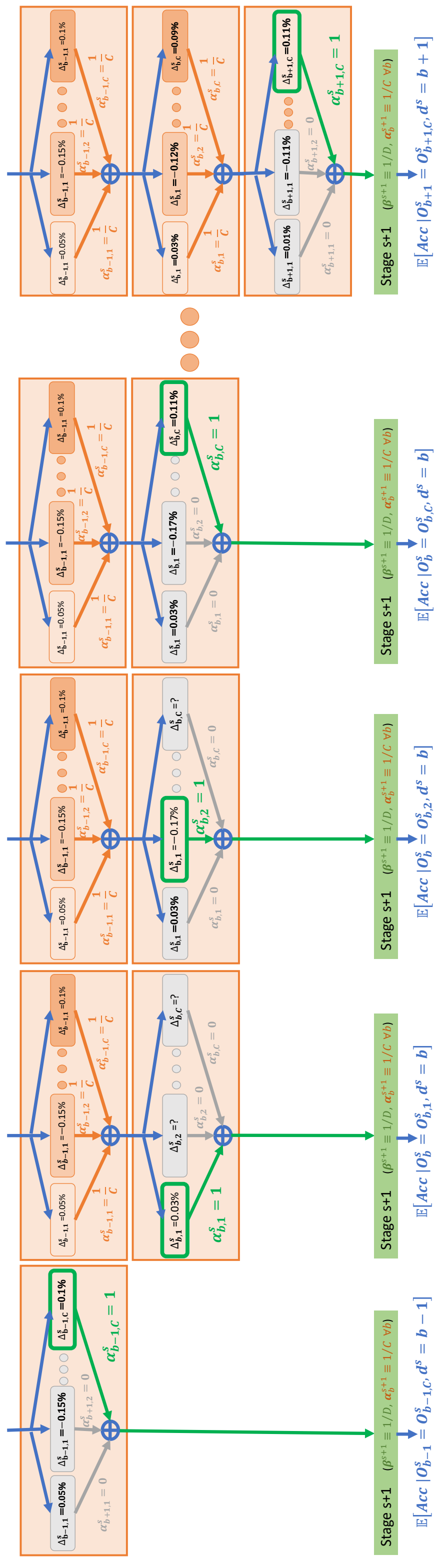


Figure 13: Extension of Figure 5: Estimating the expected accuracy gap caused by microscopic design choices on the operation applied at every block one at a time.