IQNAS: Interpretable Integer Quadratic programming Neural Architecture Search

Anonymous authors
Paper under double-blind review

Abstract

Realistic use of neural networks often requires adhering to multiple constraints on latency, energy and memory among others. A popular approach to find fitting networks is through constrained Neural Architecture Search (NAS). However, previous methods use complicated predictors for the accuracy of the network. Those predictors are hard to interpret and sensitive to many hyperparameters to be tuned, hence, the resulting accuracy of the generated models is often harmed. In this work we resolve this by introducing *Interpretable Integer Quadratic programming Neural Architecture Search (IQNAS)*, that is based on an accurate and simple quadratic formulation of both the accuracy predictor and the expected resource requirement, together with a scalable search method with theoretical guarantees. The simplicity of our proposed predictor together with the intuitive way it is constructed bring interpretability through many insights about the contribution of different design choices. For example, we find that in the examined search space, adding depth and width is more effective at deeper stages of the network and at the beginning of each resolution stage. Our experiments[1] show that IQNAS generates comparable to or better architectures than other state-of-the-art NAS methods within a reduced search cost for each additional generated network, while strictly satisfying the resource constraints.

1 Introduction

With the rise in popularity of Convolutional Neural Networks (CNN), the need for neural networks with fast inference speed and high accuracy, has been growing continuously. At first, manually designed architectures, such as VGG Simonyan & Zisserman (2015) or ResNet He et al. (2015), targeted powerful GPUs as those were the common computing platform for deep CNNs, until the need for deployment on edge devices and standard CPUs emerged. These are more limited computing platforms, requiring lighter architectures that for practical scenarios must comply with hard constraints on real time latency or power consumption. This has spawned a line of research aimed at finding architectures with both high performance and bounded resource demands.

The main approaches to solve this evolved from Neural Architecture Search (NAS) Zoph & Le (2016); Liu et al. (2018); Cai et al. (2018), while adding a constraint on the target latency over various platforms, e.g., TPU, CPU, Edge-GPU, FPGA, etc. The constrained-NAS methods can be grouped into two categories: (i) Reward based methods such as Reinforcement-Learning (RL) or Evolutionary Algorithm (EA) Cai et al. (2019); Tan et al. (2019); Tan & Le (2019); Howard et al. (2019), where the search is performed by sampling networks and predicting their final accuracy and resource demands by evaluation over some validation set. The predictors are typically made of complicated models and hence require many samples and sophisticated fitting techniques White et al. (2021). Overall this makes those oftentimes inaccurate, expensive to acquire, and hard to optimize objective functions due to their complexity. (ii) Resource-aware gradient based methods formulate a differentiable loss function consisting of a trade-off between an accuracy term and either a proxy soft penalty term Hu et al. (2020); Wu et al. (2019) or a hard constraint Nayman et al. (2021). Therefore, the architecture can be directly optimized via bi-level optimization using stochastic gradient descent (SGD) Bottou (1998) or stochastic Frank-Wolfe (SFW) Hazan & Luo (2016).

[1] The full code for search, train and evaluation with trained models will be publicly released.

However, the bi-level nature of the problem introduces many challenges Chen et al. (2019); Liang et al. (2019); Noy et al. (2020); Nayman et al. (2019) and recently Wang et al. (2021) pointed out the inconsistencies associated with using gradient information as a proxy for the quality of the architectures, especially in the presence of skip connections in the search space. This kind of inconsistencies also calls for making NAS more interpretable by extending its scope from finding optimal architectures to interpretable features Ru et al. (2021) and their corresponding impact on the network performance.

In this paper, we propose a fast and scalable search algorithm that produces architectures with high accuracy that strictly satisfy latency constraints. The proposed algorithm is based on two key ideas, with the goal of constructing an intuitive and simple accuracy predictor which is interpretable, easy to optimize and without strong reliance on gradient information:

(1) We propose a simple and well preforming quadratic accuracy predictor which is interpretable, easy to optimize and intuitive by measuring the performance contribution of individual design choices thorough sampling selected sub-networks from an one-shot model (Bender et al. (2018); Chu et al. (2019); Guo et al. (2020); Cai et al. (2019); Nayman et al. (2021)). Many insights about the contribution of various design choices can be extracted due to the way the predictor is constructed. Moreover, we show that the performance of our intuitive and simple predictor is comparable to the performance of its learnable version. Furthermore, its performance matches other more complex predictors, that are not interpretable and expensive and hard to optimize due to many hyperparameters.

(2) The quadratic form of the proposed predictor allows the formulation of the latency constrained NAS problem as an *Integer Quadratic Constrained Quadratic Programming* (IQCQP). Thanks to this, it can be efficiently solved via a simple algorithm with some off-the-shelf components.

The optimization approach we propose has several advantages. First, the outcome networks provide high accuracy and closely comply with the latency constraint. In addition, our solution is highly efficient, which makes it scalable to multiple target devices and latency demands. The efficiency is due to the formulation of the problem as an IQCQP and the deigned algorithm that solves it within few minutes on a common CPU.

2 Related Work

Neural Architecture Search methods automate models' design per provided constraints. Early methods like NASNet Zoph & Le (2016) and AmoebaNet Real et al. (2019) focused solely on accuracy, producing SotA classification models Huang et al. (2019) at the cost of GPU-years per search, with relatively large inference times. DARTS Liu et al. (2018) introduced a differential space for efficient search and reduced the training duration to days, followed by XNAS Nayman et al. (2019) and ASAP Noy et al. (2020) that applied pruning-during-search techniques to further reduce it to hours.

Predictor based methods recently have been proposed based on training a model to predict the accuracy of an architecture just from an encoding of the architecture. Popular choices for these models include Gaussian processes, neural networks, tree-based methods. See Lu et al. (2020) for such utilization and White et al. (2021) for comprehensive survey and comparisons.

Interpretabe NAS was firstly introduced by Ru et al. (2021) through a rather elaborated Bayesian optimisation with Weisfeiler-Lehman kernel to identify beneficial topological features. We propose an intuitive and simpler approach for NAS interpretibiliy for the efficient search space examined. This leads to more understanding and applicable design rules.

Hardware-aware methods such as ProxylessNAS Cai et al. (2018), Mnasnet Tan et al. (2019), FBNet Wu et al. (2019), SPNASNet Stamoulis et al. (2019) and TFNAS Hu et al. (2020) produce architectures that satisfy the required constraints by applying simple heuristics such as soft penalties on the loss function. HardCoRe-NAS Nayman et al. (2021) and OFA Cai et al. (2019) proposed a scalable approach across multiple devices by training an one-shot model Brock et al. (2017); Bender et al. (2018) once. This provides a strong pretrained super-network being highly predictive for the accuracy ranking of extracted sub-networks, e.g. SPOS Guo et al. (2020), FairNAS Guo et al. (2020). HardCore-NAS searches by backpropagation Kelley (1960) over a supernetwork under strict latency constraints for several GPU hours per network. OFA applies evolutionary search Real et al. (2019) over a complicated multilayer perceptron (MLP) Rumelhart et al. (1985) based accuracy predictor with many hyperparameters to be tuned. This work relies on such one-shot model, for intuitively building a simple quadratic accuracy predictor that matches in performance without any tuning and optimized under strict latency constraints by solving an IQCQP problem in several CPU minutes.

3 Method

In this section we propose our method for latency-constrained NAS. We search for an architecture with the highest validation accuracy under a predefined latency constraint, denoted by T. Our architecture search space $\mathcal{S}$ is parametrized by a vector $\zeta \in \mathcal{S} \subset \mathbb{Z}^N$, governing the architecture structure, and w, the convolution weights. We show in Section 3.1 that the latency-constrained NAS problem can be formulated as an IQCQP:

$$\max_{\zeta} ACC(\zeta) = q^T \zeta + \zeta^T Q \zeta \qquad \text{s.t. } LAT(\zeta) = \zeta^T \Theta \zeta \leq T, \quad A_{\mathcal{S}} \cdot \zeta \leq b_{\mathcal{S}}, \quad \zeta \in \mathbb{Z}^N \tag{1}$$

where $q \in \mathbb{R}^N$, $Q \in \mathbb{R}^{N \times N}$,$\Theta \in \mathbb{R}^{N \times N}$, $A_{\mathcal{S}} \in \mathbb{R}^{C \times N}$, $b_{\mathcal{S}} \in \mathbb{R}^C$ and $\zeta \in \mathcal{S}$ can be expressed as a set of C linear equations . We define the accuracy predictor $ACC(\zeta)$ in section 3.2 and present the quadratic formula for the latency computation $LAT(\zeta)$ in section 3.2.1. Finally, in section 3.3 we propose an optimization method to efficiently solve equation 1.

3.1 Search Space

Aiming at latency efficient architectures, we adopt the search space introduced in Nayman et al. (2021) and illustrated in Figure 1, which is closely related to those used by Wu et al. (2019); Howard et al. (2019); Tan et al. (2019); Hu et al. (2020); Cai et al. (2019). It integrates a macro search space and a micro search space. The macro search space is composed of S stages $s \in \{1, .., S = 5\}$, each composed of blocks $b \in \{1, .., D = 4\}$, and defines how the blocks are connected to one another. The micro search space is based on *Mobilenet Inverted Residual* (MBInvRes) blocks Sandler et al. (2018) and controls the internal structures of each block.

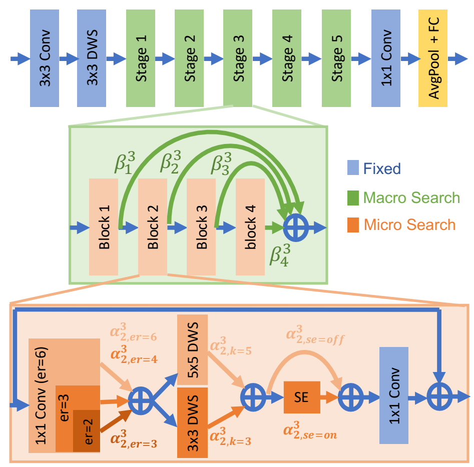

Figure 1: Search space via the one-shot model

Every MBInvRes block is configured by an expansion ratio $er \in \{3, 4, 6\}$ of the point-wise convolution, kernel size $k \in \{3 \times 3, 5 \times 5\}$ of the Depth-Wise Separable convolution (DWS), and Squeeze-and-Excitation (SE) layer Hu et al. (2018) $se \in \{\text{on}, \text{off}\}$ as shown at the bottom of Figure 1 and detailed in Appendix B. Each triplet (er, k, se) implies a block configuration $c \in \mathcal{C}$ (specified in Appendix B) that resides in a micro search space that is parameterized by $\boldsymbol{\alpha} \in \mathcal{A} = \bigotimes_{s=1}^{S} \bigotimes_{b=1}^{D} \bigotimes_{c \in \mathcal{C}} \alpha^s_{b,c}$, where $\otimes$ denotes the Cartesian product.

For each block b of stage s we have $\alpha^s_{b,c} \in \{0,1\}^{|\mathcal{C}|}$ and $\Sigma_{c \in \mathcal{C}} \alpha^s_{b,c} = 1$. An input feature map x^s_b to block b of stage s is processed as follows: $x^s_{b+1} = \sum_{c \in \mathcal{C}} \alpha^s_{b,c} \cdot O^s_{b,c}(x^s_b)$, where $O^s_{b,c}(\cdot)$ is the operation performed by the block configured according to $c = (er, k, se)$. The output of each block of every stage is also directed to the end of the stage as illustrated in the center of Figure 1. Thus, the depth of each stage s is controlled by the parameters $\boldsymbol{\beta} \in \mathcal{B} = \bigotimes_{s=1}^{S} \bigotimes_{b=1}^{D} \beta^s_b$, such that $\beta^s_b \in \{0,1\}^D$ and $\Sigma_{b=1}^{D} \beta^s_b = 1$. The depth is $d^s \in \{b \mid \beta^s_b = 1, b \in \{1, .., D\}\}$, since $x^{s+1}_1 = \Sigma_{b=1}^{D} \beta^s_b \cdot x^s_{b+1}$.

To summarize, the search space is composed of both the micro and macro search spaces parameterized by $\boldsymbol{\alpha} \in \mathcal{A}$ and $\boldsymbol{\beta} \in \mathcal{B}$, respectively, such that for all $s \in \{1, .., S\}, b \in \{1, .., D\}, c \in \mathcal{C}$:

$$\mathcal{S} = \left\{ (\boldsymbol{\alpha}, \boldsymbol{\beta}) \middle| \boldsymbol{\alpha} \in \mathcal{A}, \boldsymbol{\beta} \in \mathcal{B} \, ; \, \alpha^s_{b,c} \in \{0,1\}^{|\mathcal{C}|} \, ; \, \Sigma_{c \in \mathcal{C}} \alpha^s_{b,c} = 1 \, ; \, \beta^s_b \in \{0,1\}^D \, ; \, \Sigma_{b=1}^{D} \beta^s_b = 1 \right\} \tag{2}$$

A continuous probability distribution is induced over the space, by relaxing $\alpha_{b,c}^s \in \{0,1\}^{|\mathcal{C}|} \rightarrow \alpha_{b,c}^s \in \mathbb{R}_+^{|\mathcal{C}|}$ and $\beta_b^s \in \{0,1\}^D \rightarrow \beta_b^s \in \mathbb{R}_+^D$ to be continuous rather than discrete. Therefore, this probability distribution can be expressed by a set of linear equations and one can view the parametrization $\zeta = (\boldsymbol{\alpha}, \boldsymbol{\beta})$ as a composition of probabilities in $\mathcal{P}_\zeta(\mathcal{S}) = \{\zeta \mid A_\mathcal{S}\zeta \leq b_\mathcal{S}\} = \{(\boldsymbol{\alpha}, \boldsymbol{\beta}) \mid A_\mathcal{S}^\alpha \cdot \boldsymbol{\alpha} \leq b_\mathcal{S}^\alpha, A_\mathcal{S}^\beta \cdot \boldsymbol{\beta} \leq b_\mathcal{S}^\beta\}$ or as degenerate one-hot vectors in $\mathcal{S}$.

3.2 Quadratic Accuracy Predictors

While many predictor-based methods utilize complicated forms of accuracy predictors (see White et al. (2021)) that are hard to train and interpret, we introduce a simple predictor of a quadratic form, which can be computed efficiently while maintaining high accuracy. We evaluate its accuracy as commonly done for predictors Chu et al. (2019); White et al. (2021), by showing high ranking correlation between the accuracy of its predictions and the accuracy measured on a sub-network extracted from the one-shot model. The correlation is measured via both Kendall-Tau Maurice (1938) and Spearman Spearman (1961) coefficients.

3.2.1 Deriving a Quadratic Accuracy Estimator

We next propose an intuitive and effective way for estimating the expected accuracy of a given sub-network. We measure the contributions $\Delta_{b,c}^s = \mathbb{E}[Acc|O_{b,c}^s = O_c, d^s = b] - \mathbb{E}[Acc]$ and $\Delta_b^s = \mathbb{E}[Acc|d^s = b] - \mathbb{E}[Acc]$ of each individual decision and then aggregate all the contributions while multiplying each by its probability of participation:

$$ACC(\zeta) = ACC(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \mathbb{E}[Acc] + \sum_{s=1}^{S}\sum_{b=1}^{D} \beta_b^s \cdot \Delta_b^s + \sum_{s=1}^{S}\sum_{b=1}^{D}\sum_{b'=b}^{D}\sum_{c\in\mathcal{C}} \alpha_{b,c}^s \cdot \Delta_{b,c}^s \cdot \beta_{b'}^s \tag{3}$$

The expectations are with respect to a uniform sampling of sub-networks $\zeta \in \mathcal{S}$ excluding decisions specified in the conditional events, as illustrated in Figure 2. In practice, while equation 3 is quadratic in ζ, its vectorized form can be expressed as the following bilinear formula in $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$:

$$ACC(\zeta) = ACC(\boldsymbol{\alpha}, \boldsymbol{\beta}) = r + q_\beta^T \boldsymbol{\beta} + \boldsymbol{\alpha}^T Q_{\alpha\beta} \boldsymbol{\beta} \tag{4}$$

where $r = E[Acc]$, $q_\beta \in \mathbb{R}^{D\cdot S}$ is a vector composed of Δ_b^s and $Q_{\alpha\beta} \in \mathbb{R}^{\mathcal{C}\cdot D\cdot S \times D\cdot S}$ is a matrix composed of $\Delta_{b,c}^s$. We define the latency constraint similarly to Nayman et al. (2021):

$$LAT(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \sum_{s=1}^{S}\sum_{b=1}^{D}\sum_{b'=b}^{D}\sum_{c\in\mathcal{C}} \alpha_{b,c}^s \cdot t_{b,c}^s \cdot \beta_{b'}^s = \boldsymbol{\alpha}^T \Theta \boldsymbol{\beta} \tag{5}$$

where $\Theta \in \mathbb{R}^{\mathcal{C}\cdot D\cdot S \times D\cdot S}$ is a matrix composed of the latency measurements $t_{b,c}^s$ of every possible configuration $c \in \mathcal{C}$ of every block $b \in \{1, \dots, D\}$ in every stage $s \in \{1, \dots, S\}$.

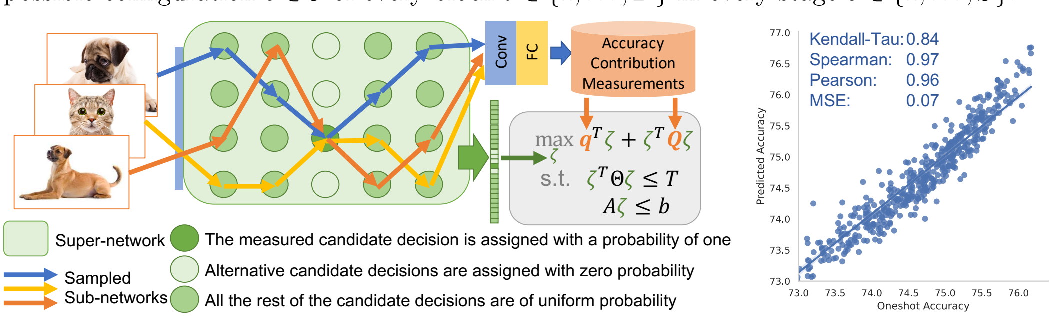

Figure 2: (Left) The IQNAS scheme constructs a quadratic accuracy estimator by measuring the accuracy contribution of individual design choices and then maximizing this objective under quadratic latency constraints. (Right) Subnetworks' accuracy predictions by the proposed estimator vs their measured accuracy. High ranking correlations are achieved.

The expectations $\mathbb{E}[Acc]$, $\mathbb{E}[Acc|O_{b,c}^s = O_c, d^s = b]$ and $\mathbb{E}[Acc|d^s = b]$ are estimated using *Multi-path* sampling Nayman et al. (2021). We average Monte-Carlo samples of distinct sub-networks, sampled uniformly for each input image in the validation set (see Figure 2). Figure 3 compares the effectiveness of *Multi-path* sampling to *Single-path* sampling, where each batch passes through a distinct uniformly sampled subnetwork as discussed in Section 4.2.3. This estimation requires $\mathcal{O}(N)$ validation epochs.

How close is the estimation proposed to the expected accuracy of an architecture?
We next present a theorem (with proof in Appendix D) that states that the estimator in equation 3 approximates well the expected accuracy of an architecture.

Theorem 3.1. *Assume $\{O_b^s, d_s\}$ for $s = 1, \ldots, S$ and $b = 1, \ldots, D$ are conditionally independent with the accuracy Acc. Suppose that there exists a positive real number $0 < \epsilon \ll 1$ such that for any $X \in \{O_b^s, d_s\}$ the following holds $|\mathbb{P}[Acc|X] - \mathbb{P}[Acc]| < \epsilon\mathbb{P}[Acc]$. Then:*

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

Theorem 3.1 and Figure 2 (right) demonstrate the effectiveness of relying on $\Delta_{b,c}^s$, Δ_b^s to express the expected accuracy of networks. Since those terms measure the accuracy contributions of individual design decisions, many insights and design rules can be extracted from those, as discussed in section 4.2.2, making the proposed estimator intuitively interpretable. Furthermore, the transitivity of ranking correlations is used in appendix H for guaranteeing good prediction performance with respect to architectures trained from scratch.

3.2.2 Learning a Quadratic Accuracy Predictor

One can wonder whether setting the coefficients $Q_{\alpha\beta}$, q_β and r of the bilinear equation 4 according to the estimates in equation 3 yields the best prediction of the accuracy of an architecture. An alternative approach is to learn those coefficients by solving a linear regression problem:

$$\min_{\tilde{r},\tilde{q}_\alpha,\tilde{q}_\beta,\tilde{Q}_{\alpha\beta}} \sum_{i=1}^{n} ||\tilde{r} + \boldsymbol{\alpha}_i^T\tilde{q}_\alpha + \boldsymbol{\beta}_i^T\tilde{q}_\beta + \boldsymbol{\alpha}_i^T\tilde{Q}_{\alpha\beta}\boldsymbol{\beta}_i - Acc(\boldsymbol{\alpha}_i,\boldsymbol{\beta}_i)||_2^2 \tag{6}$$

where $\{\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i\}_{i=1}^n$ and $Acc(\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i)$ represent n uniformly sampled subnetworks and their measured accuracy, respectively. Thus the data collection requires n validation epochs.

One can further unlock the full capacity of a quadratic predictor by allowing the coupling of all components and solving the following linear regression problem:

$$\min_{\tilde{r},\tilde{q}_\alpha,\tilde{q}_\beta,\tilde{Q}_{\alpha\beta},\tilde{Q}_\alpha,\tilde{Q}_\beta} \sum_{i=1}^{n} ||\left(\tilde{r} + \boldsymbol{\alpha}_i^T\tilde{q}_\alpha + \boldsymbol{\beta}_i^T\tilde{q}_\beta + \boldsymbol{\alpha}_i^T\tilde{Q}_{\alpha\beta}\boldsymbol{\beta}_i\right) + \boldsymbol{\alpha}_i^T\tilde{Q}_\alpha\boldsymbol{\alpha}_i + \boldsymbol{\beta}_i^T\tilde{Q}_\beta\boldsymbol{\beta}_i - Acc(\boldsymbol{\alpha}_i,\boldsymbol{\beta}_i)||_2^2$$

A closed form solution to these problems is derived in appendix E. While effective, this solution requires avoiding memory issues associated with inverting $N^2 \times N^2$ matrix and also reducing overfitting by tuning regularization effects over train-val splits of the data points. Figure 3 shows that the estimator proposed in section 3.2.1 matches the performance of those learnt predictors while being more sample efficient as discussed in section 4.2.3.

3.2.3 Beyond Quadratic Accuracy Predictors

The reader might question the expressiveness of a simple quadratic predictor and its ability to capture the complexity of architectures. Indeed, White et al. (2021) present many complex accuracy predictors and corresponding sampling techniques. To alleviate the reader's concerns we show in Figure 3 and Section 4.2.3 that the proposed quadratic estimator of Section 3.2.1 matches the performance of the commonly used Multi-Layer-Perceptron (MLP) based accuracy predictor Cai et al. (2019); Lu et al. (2020). The MLP based predictor is more complex and requires extensive hyperparameter tuning, e.g., of the depth, width, learning rate and its scheduling, weight decay, optimizer etc. It is also less efficient, lacks interpretability, and of limited utility as an objective function for NAS, as discussed in Section 3.3.

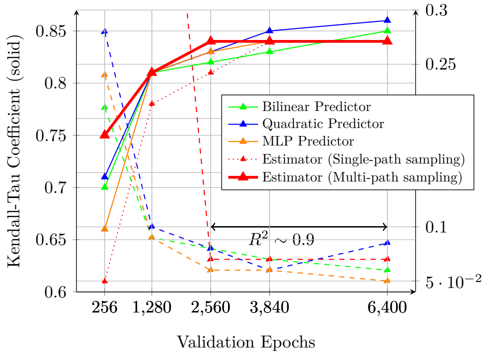

Figure 3: Performance of predictors vs samples. Ours is comparable to complex alternatives and more sample efficient.

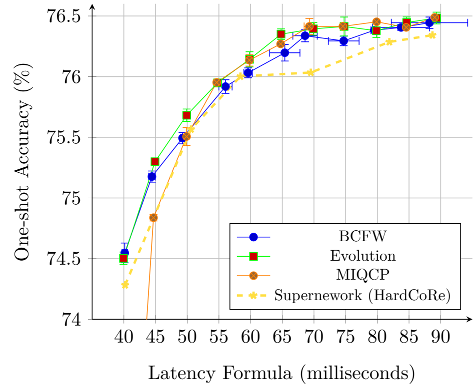

Figure 4: Comparing optimizers for solving the IQCQP over 5 random seeds. All surpass optimizing the supernetwork directly.

3.3 Solving the Integer Quadratic Constraints Quadratic Program

Having defined the quadratic objective function of equation 4, the quadratic latency constraint of equation 5 and the integer (binary) linear constraints of equation 2 that specify the search space, we can now use out-of-the-box *Mixed Integer Quadratic Constraints Programming* (MIQCP) solvers to optimize problem 1. We use IBM ILOG CPLEX that supports non-convex binary QCQP and utilizes the Branch-and-Cut algorithm Padberg & Rinaldi (1991) for this purpose. A heuristic alternative for optimizing an objective function beyond quadratic, e.g., of section 3.2.3, under integer constraints is evolutionary search Real et al. (2019). Next we propose a more theoretically sound alternative.

3.3.1 Utilizing the Block Coordinate Frank-Wolfe Algorithm

As pointed out by Nayman et al. (2021), since Θ is constructed from measured latency in equation 5, it is not guaranteed to be positive semi-definite, hence, the induced quadratic constraint makes the feasible domain in problem 1 non-convex in general. To overcome this we adapt the *Block-Coordinate Frank-Wolfe* (BCFW) Lacoste-Julien et al. (2013) for solving a continuous relaxation of problem 1, such that $\zeta \in \mathbb{R}^N_+$. Essentially BCFW adopts the Frank-Wolfe Frank et al. (1956) update rule for each of coordinates in $\zeta = (\boldsymbol{\alpha}, \boldsymbol{\beta})$ picked up at random at each iteration k for any partially differentiable objective function $ACC(\boldsymbol{\alpha}, \boldsymbol{\beta})$:

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}}{\operatorname{argmax}}\ \nabla_{\boldsymbol{\alpha}} ACC(\boldsymbol{\alpha}, \boldsymbol{\beta}_k)^T \cdot \boldsymbol{\alpha} \text{ s.t. } \boldsymbol{\beta}_k^T \Theta^T \cdot \boldsymbol{\alpha} \leq T; A_{\mathcal{S}}^{\alpha} \cdot \boldsymbol{\alpha} \leq b_{\mathcal{S}}^{\alpha} \tag{7}$$

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmax}}\ \nabla_{\boldsymbol{\beta}} ACC(\boldsymbol{\alpha}_k, \boldsymbol{\beta})^T \cdot \boldsymbol{\beta} \text{ s.t. } \boldsymbol{\alpha}_k^T \Theta \cdot \boldsymbol{\beta} \leq T; A_{\mathcal{S}}^{\beta} \cdot \boldsymbol{\beta} \leq b_{\mathcal{S}}^{\beta} \tag{8}$$

and $\delta_{k+1} = (1-\gamma_k) \cdot \delta_k + \gamma_k \cdot \hat{\delta}$ for $\delta \in \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$, where $0 \leq \gamma_k \leq 1$ and ∇_δ stands for the partial derivatives with respect to δ. This applies for both the differentiable MLP predictor of section 3.2.3 and the quadratic one of equation 7. Convergence guarantees are provided in Lacoste-Julien et al. (2013). Then, we need to project the solution back to the discrete space of architectures, specified in equation 2, as done in Nayman et al. (2021). This step could deviate from the solution and cause degradation in performance.

Algorithm 1 applies the BCFW with line-search for the special case of using the bilinear objective function specified in section 3.2.1 and in equation 6. Together with the bilinear constraints of equation 5, the resulting problem is a *Bilinear Programming* (BLP) Gallo & Ülkücü (1977) with bilinear constraints, i.e., BLCP. For this case, more specific convergence guarantees can be provided together with the sparsity of the solution, hence no additional discretization step is required. The following theorem states that after $\mathcal{O}(1/\epsilon)$ iterations, Algorithm 1 obtains an ϵ-approximate solution to problem 1. The proof is in Appendix F.

Theorem 3.2. *For each $k > 0$ the iterate ζ_k Algorithm 1 satisfies:*

$$E[ACC(\zeta_k)] - ACC(\zeta^*) \leq \frac{4}{k+4}\left(ACC(\zeta_0) - ACC(\zeta^*)\right)$$

where ζ^* is the solution of a continuous relaxation of problem 1 and the expectation is over the random choice of the block $\boldsymbol{\alpha}$ or $\boldsymbol{\beta}$.

Algorithm 1 Block Coordinate Frank-Wolfe (BCFW) with Line Search for BLCP

input $(\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0) \in \{(\boldsymbol{\alpha}, \boldsymbol{\beta}) | \boldsymbol{\alpha}\Theta\boldsymbol{\beta} \leq T, A^{\alpha}_{\mathcal{S}}\boldsymbol{\alpha} \leq b^{\alpha}_{\mathcal{S}}, A^{\beta}_{\mathcal{S}}\boldsymbol{\beta} \leq b^{\beta}_{\mathcal{S}}\}$, $0 < p < 1$

for $k = 0, \ldots, K-1$ **do**
  if $Bernoulli(p) == 1$ **then**
    $\boldsymbol{\alpha}_{k+1} = \text{argmax}_{\,\boldsymbol{\alpha}}(q^T_{\alpha} + \beta^T_k Q^T_{\alpha\beta}) \cdot \boldsymbol{\alpha}$ s.t. $\boldsymbol{\beta}^T_k \Theta^T \cdot \boldsymbol{\alpha} \leq T$; $A^{\alpha}_{\mathcal{S}} \cdot \boldsymbol{\alpha} \leq b^{\alpha}_{\mathcal{S}}$ and $\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k$
  else
    $\boldsymbol{\beta}_{k+1} = \text{argmax}_{\,\boldsymbol{\beta}}(q^T_{\beta} + \alpha^T_k Q_{\alpha\beta}) \cdot \boldsymbol{\beta}$ s.t. $\boldsymbol{\alpha}^T_k \Theta \cdot \boldsymbol{\beta} \leq T$; $A^{\beta}_{\mathcal{S}} \cdot \boldsymbol{\beta} \leq b^{\beta}_{\mathcal{S}}$ and $\boldsymbol{\alpha}_{k+1} = \boldsymbol{\alpha}_k$
  end if
end for

output $\zeta^* = (\boldsymbol{\alpha}_K, \boldsymbol{\beta}_K)$

We next provide a guarantee that Algorithm 1 yields a *sparse solution*, representing a valid sub-network of the one-shot model up to a single probability vector from those composing $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, which contains up to two non-zero entries each, as all the rest are one-hot vectors. Hence, no further discretization step is required. The proof is in Appendix G.

Theorem 3.3. *The output solution $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \zeta^*$ of Algorithm 1 admits:*

$$\sum_{c \in \mathcal{C}} |\alpha^s_{b,c}|^0 = 1 \, \forall (s,b) \in \{1,..,S\} \otimes \{1,..,D\} \setminus \{(s_{\boldsymbol{\alpha}}, b_{\boldsymbol{\alpha}})\} \textit{ and } \sum_{b=1}^{D} |\beta^s_b|^0 = 1 \quad \forall s \in \{1,..,S\} \setminus \{s_{\boldsymbol{\beta}}\}$$

where $|\cdot|^0 = \mathbb{1}\{\cdot > 0\}$ and $(s_{\boldsymbol{\alpha}},\, b_{\boldsymbol{\alpha}})$, $s_{\boldsymbol{\beta}}$ are single block and stage respectively, satisfying:

$$\sum_{c \in \mathcal{C}} |\alpha^{s_{\boldsymbol{\alpha}}}_{b_{\boldsymbol{\alpha}},c}|^0 \leq 2 \quad ; \quad \sum_{b=1}^{D} |\beta^{s_{\boldsymbol{\beta}}}_b|^0 \leq 2 \tag{9}$$

A negligible latency deviation is associated with taking the argmax over the only two couples referred to in equation 9. Experiments supporting this are described in Section 4.2.4.

4 Experimental Results

4.1 Search for State-of-the-Art Architectures

4.1.1 Dataset and Setting

The train data for the accuracy predictors of sections 3.2.2 and 3.2.3 is composed of sub-networks uniformly sampled from the super-network and their corresponding validation accuracy is measured over the same 20% of the Imagenet train set, considered as a validation set. The same validation set is used for the Monte-Carlo sampling mentioned in Section 3.2.1. To avoid overfitting we use regularization when learning accuracy predictors whose coefficient is tuned over 10% of the collected data, see appendix E. The test set for evaluating the ranking correlations of all the accuracy predictors is composed of another 500 samples generated uniformly in the same way. More reproduciblity details are provided in appendix C.

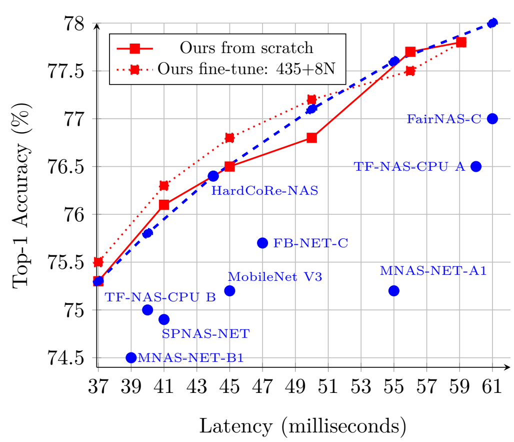

Figure 5: Imagenet Top-1 accuracy vs latency.

4.1.2 Comparisons with other methods

We compare our generated architectures to other state-of-the-art NAS methods in Table 1 and Figure 5. For the purpose of comparing the generated architectures alone, excluding the contribution of evolved pretraining techniques, for each model in Table 1, the official PyTorch implementation Paszke et al. (2019) is trained from a random initialization (besides

Model	Latency (ms)	Top-1 (%)	Total Cost (GPU hours)
MnasNetB1	39	74.5	40,000N
TFNAS-B	40	75.0	263N
SPNASNet	41	74.9	288 + 408N
OFA CPU[2]	42	75.7	1200 + 25N
HardCoRe A	40	75.8	400 + 15N
Ours 40 ms	**40**	**76.1**	435 + **8N**
MobileNetV3	45	75.2	180N
FBNet	47	75.7	576N
MnasNetA1	55	75.2	40,000N
HardCoRe B	44	76.4	400 + 15N
Ours 45 ms	**45**	**76.5**	435 + **8N**
MobileNetV2	70	76.5	150N
TFNAS-A	60	76.5	263N
HardCoRe C	**50**	**77.1**	400 + 15N
Ours 50 ms	**50**	76.8	435 + **8N**
EfficientNetB0	85	77.3	
HardCoRe D	55	77.6	400 + 15N
Ours 55 ms	**55**	**77.7**	435 + **8N**
FairNAS-C	60	77.0	240N
HardCoRe E	61	**78.0**	400 + 15N
Ours 60 ms	**59**	77.8	435 + **8N**

Model	Latency (ms)	Top-1 (%)
MobileNetV3	28	75.2
TFNAS-D	30	74.2
HardCoRe A	27	75.7
Ours 25 ms	**26**	**76.4**
MnasNetA1	37	75.2
MnasNetB1	34	74.5
FBNet	41	75.7
SPNASNet	36	74.9
TFNAS-B	44	76.3
TFNAS-C	37	75.2
HardCoRe B	32	**77.3**
Ours 30 ms	**31**	76.8
TFNAS-A	54	76.9
EfficientNetB0	48	77.3
MobileNetV2	50	76.5
HardCoRe C	41	**77.9**
Ours 40 ms	**40**	77.5

Table 1: ImageNet top-1 accuracy, latency and cost comparison with other methods. The total cost stands for the search and training cost of N networks. Latency is reported for (Left) Intel Xeon CPU and (Right) NVIDIA P100 GPU with a batch size of 1 and 64 respectively.

OFA [2]) using the exact same techniques and code, as specified in section 4.1.1. We report the maximum accuracy between the original paper and our training. We emphasize that all latency values presented are actual time measurements of the models, running on a single thread with the exact same settings and on the same hardware. We excluded further optimizations, such as Intel MKL-DNN Intel (R), therefore, the latency we report may differ from the one originally reported. It can be seen that networks generated by our method meet the latency target closely, while at the same time is comparable to or surpassing all the other methods on the top-1 accuracy over ImageNet with a reduced scalable search time. The total search time consists of 435 GPU hours computed only once as preprocessing and additional 8 GPU hours for fine-tuning each generated network, while the search itself requires negligible several CPU minutes, see appendix A for more details.

4.2 Empirical Analysis of Key Components

In this section we analyze and discuss different aspects of the proposed method.

4.2.1 The Contribution of Different Terms of the Accuracy Estimator

The accuracy estimator in equation 3 aggregates the contributions of multiple architectural decisions. In equation 4, those decisions are grouped into two groups: (1) macroscopic decisions about the depth of each stage are expressed by q_β and (2) microscopic decisions about the configuration of each block are expressed by $Q_{\alpha\beta}$.

Variant	Kendall-Tau	Spearman
$q_\beta \equiv 0$	0.29	0.42
$Q_{\alpha\beta} \equiv 0$	0.66	0.85
$ACC(\boldsymbol{\alpha}, \boldsymbol{\beta})$	0.84	0.97

Table 2: Contribution of terms.

Table 2 quantifies the contribution of each of those terms to the ranking correlations by setting the corresponding terms to zero. We conclude that the depth of the network is very significant for estimating the accuracy of architectures, as setting q_β to zero specifically decreases the Kendall-Tau and Spearman's correlation coefficients from 0.84 and 0.97 to 0.29 and 0.42 respectively. The significance of microscopic decisions about the configuration of blocks is also viable but not as much, as setting $Q_{\alpha\beta}$ to zero decreases the Kendall-Tau and Spearman's correlation coefficients to 0.66 and 0.85 respectively.

[2]Finetuning a model obtained by 1200 GPU hours.

4.2.2 Interpretability of the Accuracy Estimator

The way the accuracy estimator in section 3.2.1 is constructed brings many insights about the contribution of different design choices to the accuracy, as shown in Figure 6.

Deepen later stages: In the left figure $\Delta_b^s - \Delta_{b-1}^s$ are presented for $b = 3, 4$ and $s = 1, \ldots, 5$. This graph shows that increasing the depth of deeper stages is more beneficial than doing so for shallower stages. Showing also the latency cost for adding a block to each stage, we see that there is a strong motivation to make later stages deeper.

Add width and S&E to later stages and shallower blocks: In the middle and right figures, $\Delta_{b,c}^s$ are averaged over different configurations and block or stages respectively for showing the contribution of microscopic design choices. Those show that increasing the expansion ratio and adding S&E is more significant in deeper stages and at sooner blocks within each stage. **Width and S&E over bigger kernels**: Increasing the kernel size is relatively less significant and is more effective at intermediate stages.

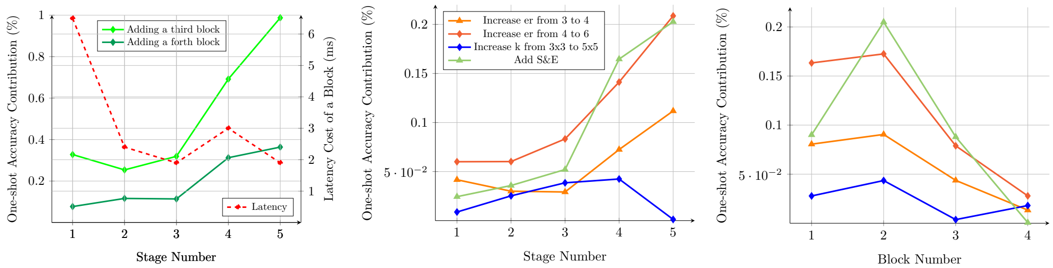

Figure 6: Design choices insights deduced from the accuracy estimator: The contribution of (Left) depth for different stages, (Middle) expansion ration, kernel size and S&E for different stages and (Right) for different blocks within a stage.

4.2.3 Ranking Correlation Per Cost for Different Accuracy Predictors

Figure 3 presents the Kendall-Tau ranking correlation coefficients and MSE of different accuracy predictors that are introduced in Section 3.2 versus the number of epochs of the validation set required for obtaining their parameters. It is noticable that the simple bilinear accuracy estimator (section 3.2.1) is more sample efficient, as its parameters are estimated rather than learned. The effectiveness of the Multi-path sampling Nayman et al. (2021) of driving each image through a distinct subnetwork for obtaining the accuracy estimator is very clear comparing to the Single-path counterpart of driving each batch through the same subnetwork. With access to enough samples, the performance of these all are comparable.

4.2.4 Comparison of Optimization Algorithms

Formulating the NAS problem as IQCQP affords the utilization of a variety of optimization algorithms. Figure 4 compares the one-shot accuracy and latency of networks generated by utilizing the algorithms suggested in section 3.3 for solving problem 1 with the bilinear estimator introduced in section 3.2.1 serving as the objective function. Error bars for both accuracy and latency are presented for 5 different seeds. All algorithms satisfy the latency constraints up to a reasonable error of less than 10%. While all of them surpass the performance of BCSFW Nayman et al. (2021), given as reference, BCFW is superior at low latency, evolutionary search does well over all and MIQCP is superior at high latency. Hence, for practical purposes we apply the three of them for search and take the best one, with negligible computational cost of less than three CPU minutes overall.

5 Conclusion

The problem of resource-aware NAS is formulated as an IQCQP optimization problem. The quadratic constraints express resource requirements and a quadratic accuracy estimator serves as the objective function. This estimator is constructed by measuring the individual contribution of design choices, which makes it intuitive and interpretable. Indeed, its interpretability brings several insights and design rules. Its performance is comparable to complex predictors that are more expensive to acquire and harder to optimize. Efficient optimization algorithms are proposed for solving the resulted IQCQP problem. IQNAS is a faster search method, scalable to many devices and requirements, while generating comparable or better architectures than those of other state-of-the-art NAS methods.

6 Reproducibility Statement

In this paper, we have made efforts to produce explanations and present clear algorithms describing every step of the different components of our method. In addition, a source code will be released upon publication that will allow to reproduce every result presented in the paper. In the appendix, we present an overview of our method and its computational cost (Appendix A), a clear description of our search space (Appendix B), the experimental setting with training procedure details to obtain our results (Appendix C), as well as proof of Theorem 3.1 (Appendix D), and the concise description of the algorithm we used to find the closed form regularized solution of the linear regression problems described in section 3.2 (Appendix E), to ensure full reproducibility. In addition, we present more theoretical results related to the convergence of Algorithm 1 and the sparsity of it solution (Appendix F, G, H).

References

Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, pp. 550–559. PMLR, 2018.

Léon Bottou. Online algorithms and stochastic approxima-p tions. *Online learning and neural networks*, 1998.

Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*, 2017.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019.

Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive differentiable architecture search: Bridging the depth gap between search and evaluation. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1294–1303, 2019.

Xiangxiang Chu, Bo Zhang, Ruijun Xu, and Jixiang Li. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. *arXiv preprint arXiv:1907.01845*, 2019.

Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Giorgio Gallo and Aydin Ülkücü. Bilinear programming: an exact algorithm. *Mathematical Programming*, 12(1):173–194, 1977.

Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. In *European Conference on Computer Vision*, pp. 544–560. Springer, 2020.

Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, pp. 1263–1271. PMLR, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*, 2015. URL http://arxiv.org/abs/1503.02531.

Andrew Howard, Ruoming Pang, Hartwig Adam, Quoc V. Le, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, and Yukun Zhu. Searching for mobilenetv3. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 1314–1324. IEEE, 2019. doi: 10.1109/ICCV.2019.00140. URL https://doi.org/10.1109/ICCV.2019.00140.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Yibo Hu, Xiang Wu, and Ran He. Tf-nas: Rethinking three search freedoms of latency-constrained differentiable neural architecture search. *arXiv preprint arXiv:2008.05314*, 2020.

Yanping Huang, Youlong Cheng, Ankur Bapna, Orhan Firat, Dehao Chen, Mia Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V Le, Yonghui Wu, et al. Gpipe: Efficient training of giant neural networks using pipeline parallelism. In *Advances in neural information processing systems*, pp. 103–112, 2019.

IBM ILOG CPLEX. Ibm ilog cplex miqcp optimizer. https://www.ibm.com/docs/en/icos/12.7.1.0?topic=smippqt-miqcp-mixed-integer-programs-quadratic-terms-in-constraints.

Intel(R). Intel(r) math kernel library for deep neural networks (intel(r) mkl-dnn), 2019. URL https://github.com/rsdubtso/mkl-dnn.

Hans Kellerer, Ulrich Pferschy, and David Pisinger. *The Multiple-Choice Knapsack Problem*, pp. 317–347. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-24777-7. doi: 10.1007/978-3-540-24777-7_11. URL https://doi.org/10.1007/978-3-540-24777-7_11.

Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954, 1960.

Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *International Conference on Machine Learning*, pp. 53–61. PMLR, 2013.

Eric Langford, Neil Schwertman, and Margaret Owens. Is the property of being positively correlated transitive? *The American Statistician*, 55(4):322–325, 2001.

Hanwen Liang, Shifeng Zhang, Jiacheng Sun, Xingqiu He, Weiran Huang, Kechen Zhuang, and Zhenguo Li. Darts+: Improved differentiable architecture search with early stopping. *arXiv preprint arXiv:1909.06035*, 2019.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. NSGANetV2: Evolutionary multi-objective surrogate-assisted neural architecture search. In *European Conference on Computer Vision (ECCV)*, 2020.

Kendall Maurice. A new measure of rank correlation. *Biometrika*, 30(1-2):81–89, 1938.

Niv Nayman, Asaf Noy, Tal Ridnik, Itamar Friedman, Rong Jin, and Lihi Zelnik. Xnas: Neural architecture search with expert advice. In *Advances in Neural Information Processing Systems*, pp. 1977–1987, 2019.

Niv Nayman, Yonathan Aflalo, Asaf Noy, and Lihi Zelnik-Manor. Hardcore-nas: Hard constrained differentiable neural architecture search. *arXiv preprint arXiv:2102.11646*, 2021.

Asaf Noy, Niv Nayman, Tal Ridnik, Nadav Zamir, Sivan Doveh, Itamar Friedman, Raja Giryes, and Lihi Zelnik. Asap: Architecture search, anneal and prune. In *International Conference on Artificial Intelligence and Statistics*, pp. 493–503. PMLR, 2020.

Manfred Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100, 1991.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=j9Rv7qdXjd`.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Charles Spearman. "general intelligence" objectively determined and measured. 1961.

Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 481–497. Springer, 2019.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114. PMLR, 2019. URL `http://proceedings.mlr.press/v97/tan19a.html`.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2820–2828, 2019.

Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable nas. *arXiv preprint arXiv:2108.04392*, 2021.

Colin White, Arber Zela, Binxin Ru, Yang Liu, and Frank Hutter. How powerful are performance predictors in neural architecture search? *arXiv preprint arXiv:2104.01177*, 2021.

Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 10734–10742. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.01099.

Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

Appendix

A An Overview of the Method and Computational Costs

Figure 7 presents and overview scheme of the method:

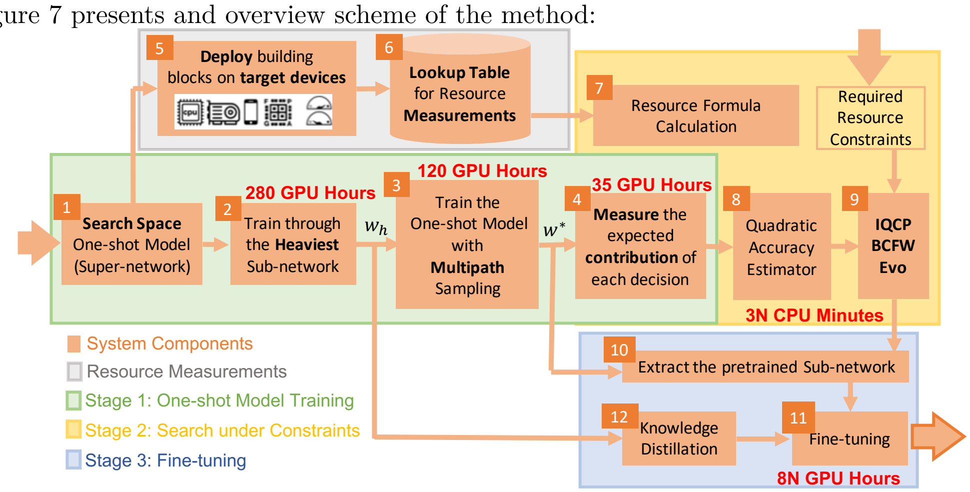

Figure 7: An Overview scheme of the IQNAS method with computational costs

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, as shown in figure 5 by comparing the dashed red line with the solid one, 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 7) that have to do with constructing the quadratic accuracy estimator of section 3.2.1 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 3 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. This makes our method scalable for many devices and latency requirements.

B More Specifications of the Search Space

Inspired by EfficientNet Tan & 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 depicted in Figure 1 and in Table 3a. 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

(a) Macro architecture of the one-shot model.

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

(b) Configurations.

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.

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.

D Proof of Theorem 3.1

Theorem D.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:*

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

Then we have that:

$$\mathbb{E}\left[A|X_1 = x_1, \dots, X_n = x_n\right] = \mathbb{E}\left[A\right] + \sum_i \left(\mathbb{E}\left[A|X_i = x_i\right] - \mathbb{E}\left[A\right]\right)(1 + O(n\epsilon)). \tag{10}$$

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

$$\begin{aligned}\mathbb{P}\left[A = a|X = x, Y = y\right] &= \frac{\mathbb{P}\left[X = x|A = a\right]\mathbb{P}\left[Y = y|A = a\right]\mathbb{P}\left[A = a\right]}{\mathbb{P}\left[X = x\right]\mathbb{P}\left[Y = y\right]} \\ &= \frac{\mathbb{P}\left[A = a|X = x\right]\mathbb{P}\left[A = a|Y = y\right]}{\mathbb{P}\left[A = a\right]}\end{aligned} \tag{11}$$

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 (11), 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:

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

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

A simple induction shows that:

$$v_n+(1-\epsilon)v_{n-1}+(1-\epsilon)^2v_{n-2}+\dots+(1-\epsilon)^n v_1 < u_n < v_n+(1+\epsilon)v_{n-1}+(1+\epsilon)^2v_{n-2}+\dots+(1+\epsilon)^n v_1$$

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 D.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 $s=1,\dots,S$ and $b=1,\dots,D$ are conditionally independent with the accuracy Acc. In our setting, we have

$$\begin{aligned}\mathbb{E}\left[Acc\middle|\cap_{s=1}^S\cap_{b=1}^D O_b^s;\cap_{s=1}^S d^s\right]=&\,\mathbb{E}\left[Acc\middle|\cap_{s=1}^S\cap_{b=1}^{d^s}\{O_b^s,d^s\};\cap_{s=1}^S d^s\right] &(12)\\ \approx&\,\mathbb{E}\left[Acc\right]\\ &+\sum_{s=1}^S\sum_{b=1}^D\left(\mathbb{E}\left[Acc|O_b^s,d^s\right]-\mathbb{E}\left[Acc\right]\right)\mathbb{1}_{b\le d_s}\left(1+O(N\epsilon)\right) &(13)\\ &+\sum_{s=1}^S\left(\mathbb{E}\left[Acc|d^s\right]-\mathbb{E}\left[Acc\right]\right)\mathbb{1}_{b\le d^s}\left(1+O(N\epsilon)\right) &(14)\end{aligned}$$

where equation 12 is since the accuracy is independent of blocks that are not participating in the subnetowrk, i.e. with $b>d^s$, and equations 13 and 14 are by utilizing Theorem D.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 $b=1,\dots,D$ 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,

$$\begin{aligned}\mathbb{E}\left[Acc|d^s=b^s\right]-\mathbb{E}\left[Acc\right]=&\sum_{b=1}^D\mathbb{1}_{b=b^s}(\mathbb{E}\left[Acc|d^s=b\right]-\mathbb{E}\left[Acc\right])\\ =&\sum_{b=1}^D\beta_b^s(\mathbb{E}\left[Acc|d^s=b\right]-\mathbb{E}\left[Acc\right])=\sum_{b=1}^D\beta_b^s\Delta_b^s &(15)\end{aligned}$$

Similarly,

$$
\begin{aligned}
\mathbb{E}\left[Acc|O_b^s = O_{c_b^s}, d_s = b\right] - \mathbb{E}\left[Acc\right] &= \sum_{c\in\mathcal{C}} \mathbb{1}_{c=c_b^s}(\mathbb{E}\left[Acc|O_b^s = O_c, d_s = b\right] - \mathbb{E}\left[Acc\right]) \\
&= \sum_{c\in\mathcal{C}} \alpha_{b,c}^s(\mathbb{E}\left[Acc|O_b^s = O_c, d_s = b\right] - \mathbb{E}\left[Acc\right]) \\
&= \sum_{c\in\mathcal{C}} \alpha_{b,c}^s \Delta_{b,c}^s
\end{aligned} \tag{16}
$$

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

$$
\begin{aligned}
\left(\mathbb{E}\left[Acc|O_b^s = O_{c_b^s}, d_s = b\right] - \mathbb{E}\left[Acc\right]\right)\mathbb{1}_{b\leq d_s} &= \sum_{c\in\mathcal{C}} \alpha_{b,c}^s \Delta_{b,c}^s \cdot \mathbb{1}_{b\leq d_s} \\
&= \sum_{b'=b}^{D}\sum_{c\in\mathcal{C}} \alpha_{b,c}^s \cdot \Delta_{b,c}^s \cdot \beta_{b'}^s
\end{aligned} \tag{17}
$$

Finally by setting equations 15 and 17 into 13 and 14 respectively, we have,

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

E 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:

$$\min_{\mathbf{Q},\mathbf{a},b} \sum_{(\mathbf{x},\mathbf{y})\in(X_{\text{train}},Y_{\text{train}})} \|\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{x}^T\mathbf{a} + b - \mathbf{y}\|^2 \tag{18}$$

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 18 can be expressed as:

$$\min_{\mathbf{q},\mathbf{a},b} \sum_{(\mathbf{x},y)\in(X_{\text{train}},Y_{\text{train}})} \|\left(\mathbf{x},\mathbf{x}\otimes\mathbf{x}\right)^T(\mathbf{a},\mathbf{q}) + b - y\|^2. \tag{19}$$

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:

$$\min_{\mathbf{v},b} \sum_{(\tilde{\mathbf{x}},\mathbf{y})\in(\tilde{X}_{\text{train}},Y_{\text{train}})} \|\tilde{\mathbf{x}}^T\mathbf{v} + b - y\|^2. \tag{20}$$

We rewrite the objective function of equation 20 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:

$$\mathbf{v}^T\tilde{\mathbf{X}}\tilde{\mathbf{X}}^T\mathbf{v} + 2(b\mathbb{1} - \mathbf{Y})^T\tilde{\mathbf{X}}^T\mathbf{v} + (b\mathbb{1} - \mathbf{Y})^T(b\mathbb{1} - \mathbf{Y}) \tag{21}$$

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 21 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:

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 $\hat{x}_i = \tilde{x}_i - \text{mean}_{i=1,\ldots,N}\left(\tilde{x}_i\right),\ \forall i \in \{1, \ldots, N\}$
3: Perform a centering on y_i computing $\hat{y}_i = y_i - \text{mean}_{i=1,\ldots,N}\left(y_i\right),\ \forall i \in \{1, \ldots, N\}$
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

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 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.

F 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.

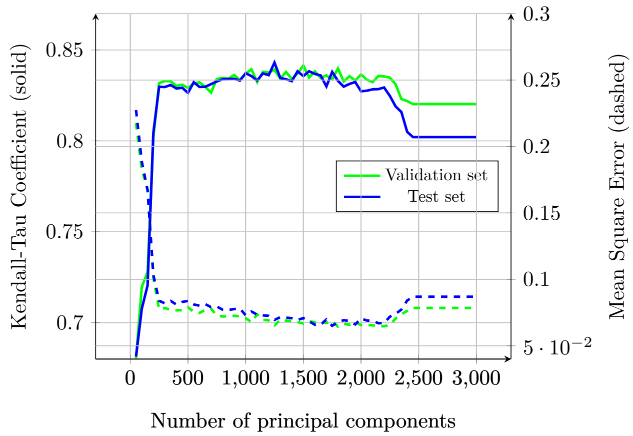

Figure 8: Kendall-Tau correlation coefficients and MSE of different predictors vs number of principal components

F.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:

$$\min_{\zeta \in \mathcal{M}^{(1)} \times \cdots \times \mathcal{M}^{(n)}} f(\zeta) \tag{22}$$

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** $k = 0, \ldots, K$ **do**
2: Pick i at random in $\{1, \ldots, n\}$
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 22, and guaranteed ϵ-small duality gap.

Theorem F.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 22 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:

$$g(\zeta) = \max_{s \in \mathcal{M}^{(1)} \times \cdots \times \mathcal{M}^{(n)}} (\zeta - s)^T \cdot \nabla f(\zeta) \tag{23}$$

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 $\mathcal{M}^{(i)}$:

$$C_f^{(i)} = \sup_{\substack{x \in \mathcal{M}^{(1)} \times \cdots \times \mathcal{M}^{(n)}, \\ s^{(i)} \in \mathcal{M}^{(i)}, \gamma \in [0,1], \\ y^{(i)} = (1-\gamma)x^{(i)} + \gamma s^{(i)}, \\ y^{\backslash(i)} = x^{\backslash(i)}}} \frac{2}{\gamma^2} \left(f(y) - f(x) - \left(y^{(i)} - x^{(i)} \right)^T \nabla^{(i)} f(x) \right) \tag{24}$$

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 F.1 is given in Lacoste-Julien et al. (2013).

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

$$f(\zeta) = \sum_{i=1}^{n} \left(\zeta^{(i)}\right)^T \cdot p_f^{(i)} + \sum_{i=1}^{n} \sum_{j=i}^{n} \left(\zeta^{(i)}\right)^T \cdot Q_f^{(i,j)} \cdot \zeta^{(j)} \tag{25}$$

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:

$$\begin{aligned} s &= \underset{s' \in \mathcal{M}^{(i)}}{\operatorname{argmin}} \ \nabla^{(i)} f(\zeta)^T \cdot s' && (26) \\ &= \underset{s' \in \mathcal{M}^{(i)}}{\operatorname{argmin}} \left(\left(p_f^{(i)}\right)^T + \sum_{j \in \{1,\ldots,i-1\}} \left(\zeta^{(j)}\right)^T \cdot Q_f^{(i,j)} + \sum_{j \in \{i+1,\ldots,n\}} \left(\zeta^{(j)}\right)^T \cdot \left(Q_f^{(i,j)}\right)^T \right) \cdot s' \end{aligned}$$

and the Line-Search at line 5 reads:

$$\begin{aligned} \gamma =: &\underset{\gamma' \in [0,1]}{\operatorname{argmin}} \ f\left((1-\gamma') \cdot \zeta + \gamma' \cdot \tilde{s}\right) \\ = &\underset{\substack{\gamma' \in [0,1] \\ y = (1-\gamma') \cdot \zeta^{(i)} + \gamma' \cdot s}}{\operatorname{argmin}} \left(\left(p_f^{(i)}\right)^T + \sum_{j \in \{1,\ldots,i-1\}} \left(\zeta^{(j)}\right)^T \cdot Q_f^{(i,j)} + \sum_{j \in \{i+1,\ldots,n\}} \left(\zeta^{(j)}\right)^T \cdot \left(Q_f^{(i,j)}\right)^T \right) \cdot y \\ = &\underset{\substack{\gamma' \in [0,1] \\ y = (1-\gamma') \cdot \zeta^{(i)} + \gamma' \cdot s}}{\operatorname{argmin}} \nabla^{(i)} f(\zeta)^T \cdot y && (27) \end{aligned}$$

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 26 the solution to 27 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. □

F.3 Solving BLCP by BCFW with Line-Search

In addition to a bilinear objective function as in equation 25, consider also a domain that is specified by the following bilinear constraints:

$$\sum_{i=1}^{n} \left(\zeta^{(i)}\right)^T \cdot p_{\mathcal{M}}^{(i)} + \sum_{i=1}^{n} \sum_{j=i}^{n} \left(\zeta^{(i)}\right)^T \cdot Q_{\mathcal{M}}^{(i,j)} \cdot \zeta^{(j)} \leq T \quad ; \quad A \cdot \zeta \leq b \tag{28}$$

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:

$$\left(\left(p_{\mathcal{M}}^{(i)}\right)^T + \sum_{j \in \{1,\ldots,i-1\}} \left(\zeta^{(j)}\right)^T \cdot Q_{\mathcal{M}}^{(i,j)} + \sum_{j \in \{i+1,\ldots,n\}} \left(\zeta^{(j)}\right)^T \cdot \left(Q_{\mathcal{M}}^{(i,j)}\right)^T \right) \cdot \zeta^{(i)} \leq T \tag{29}$$

$$A^{(i)} \cdot \zeta^{(i)} \leq b^{(i)} \tag{30}$$

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:

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

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

Algorithm 4 BCFW with Line-Search on QCQP Product Domain

input $\zeta_0 \in \{\zeta \mid \sum_{i=1}^{n} \left(\zeta^{(i)}\right)^T \cdot p_{\mathcal{M}}^{(i)} + \sum_{i=1}^{n} \sum_{j=i}^{n} \left(\zeta^{(i)}\right)^T \cdot Q_{\mathcal{M}}^{(i,j)} \cdot \zeta^{(j)} \leq T \quad ; \quad A \cdot \zeta \leq b\}$

for $k = 0, \ldots, K$ **do**

Pick i at random in $\{1, \ldots, n\}$

Keep the same values for all other coordinate blocks $\zeta_{k+1}^{\backslash(i)} = \zeta_k^{\backslash(i)}$ and update:

$$\begin{aligned}
\zeta_{k+1}^{(i)} = \operatorname*{argmin}_{s} & \left(\left(p_f^{(i)}\right)^T + \sum_{j \in \{1,\ldots,i-1\}} \left(\zeta^{(j)}\right)^T \cdot Q_f^{(i,j)} + \sum_{j \in \{i+1,\ldots,n\}} \left(\zeta^{(j)}\right)^T \cdot \left(Q_f^{(i,j)}\right)^T \right) \cdot s \\
s.t. & \left(\left(p_{\mathcal{M}}^{(i)}\right)^T + \sum_{j \in \{1,\ldots,i-1\}} \left(\zeta^{(j)}\right)^T \cdot Q_{\mathcal{M}}^{(i,j)} + \sum_{j \in \{i+1,\ldots,n\}} \left(\zeta^{(j)}\right)^T \cdot \left(Q_{\mathcal{M}}^{(i,j)}\right)^T \right) \cdot s \leq T \\
& A^{(i)} \cdot s \leq b^{(i)}
\end{aligned}$$

end for

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

$$\begin{array}{lllllll}
\zeta^{(1)} = \boldsymbol{\alpha} & m_1 = D \cdot S \cdot |\mathcal{C}| & p_f^{(1)} = p_\alpha & p_{\mathcal{M}}^{(1)} = 0 & A^{(1)} = A_{\mathcal{S}}^{\alpha} & b^{(1)} = b_{\mathcal{S}}^{\alpha} & Q_f^{(1,2)} = Q_{\alpha\beta} \\
\zeta^{(2)} = \boldsymbol{\beta} & m_2 = D \cdot S & p_f^{(2)} = p_\beta & p_{\mathcal{M}}^{(2)} = 0 & A^{(2)} = A_{\mathcal{S}}^{\beta} & b^{(2)} = b_{\mathcal{S}}^{\beta} & Q_{\mathcal{M}}^{(1,2)} = \Theta
\end{array} \tag{31}$$

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

F.3.1 Proof of Theorem 1

Let us first compute the curvature constants $C_f^{(i)}$ (equation 24) and $C_f^{\otimes}$ for the bilinear objective function as in equation 25.

Lemma F.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:

$$f(x) = \sum_{l=1}^{n} \left(x^{(l)}\right)^T \cdot p_f^{(l)} + \sum_{l=1}^{n}\sum_{j=l}^{n} \left(x^{(l)}\right)^T \cdot Q_f^{(l,j)} \cdot x^{(j)} \tag{32}$$

$$= \left(x^{(i)}\right)^T \cdot p_f^{(i)} + \sum_{j\in\{1,\dots,i-1\}} \left(x^{(j)}\right)^T \cdot Q_f^{(i,j)} \cdot x^{(i)} + \sum_{j\in\{i+1,\dots,n\}} x^{(i)} \cdot Q_f^{(i,j)} \cdot x^{(j)} \tag{33}$$

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

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:

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

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

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

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

Hence,

$$f(y) - f(x) = \nabla^{(i)} f(x) \cdot \left(y^{(i)} - x^{(i)}\right) \tag{39}$$

since 34 and 38 cancel out and,

$$\nabla^{(i)} f(x) = \left(\left(p_f^{(i)}\right)^T + \sum_{j\in\{1,\dots,i-1\}} \left(x^{(j)}\right)^T \cdot Q_f^{(i,j)} + \sum_{j\in\{i+1,\dots,n\}} \left(x^{(j)}\right)^T \cdot \left(Q_f^{(i,j)}\right)^T\right) \tag{40}$$

Hence we have,

$$C_f^{(i)} = 0 \quad \forall i \in \{1,\dots,n\} \qquad ; \qquad C_f^{\otimes} = \sum_{i=1}^{n} C_f^{(i)} = 0 \tag{41}$$

□

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

Theorem F.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 22 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 31 for $f(\zeta) := ACC(\zeta)$, theorem 3.2 follows.

G 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 G.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} $\forall i \in \{1, \ldots, k\}, j \in N_i$ respectively, the relaxed Multiple Choice Knapsack Problem (MCKP) is formulated as following:*

$$
\begin{aligned}
&\max_{vu} \sum_{i=1}^{k} \sum_{j \in N_i} p_{ij} \boldsymbol{u}_{ij} \\
&s.t. \sum_{i=1}^{k} \sum_{j \in N_i} t_{ij} \boldsymbol{u}_{ij} \leq T \\
&\sum_{j \in N_i} \boldsymbol{u}_{ij} = 1 \qquad \forall i \in \{1, \ldots, k\} \\
&\boldsymbol{u}_{ij} \geq 0 \qquad \forall i \in \{1, \ldots, k\}, j \in N_i
\end{aligned}
\tag{42}
$$

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 G.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 G.1. *The solution $\boldsymbol{u}^*$ of the relaxed MCKP equation 42 is composed of vectors $\boldsymbol{u}_i^*$ that are all one-hot but a single one.*

Lemma G.2. *The single non one-hot vector of the solution $\boldsymbol{u}^*$ of the relaxed MCKP equation 42 has at most two nonzero elements.*

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

In order to prove Theorem 3.3, we use Lemmas G.1 and G.1 for each coordinate block $\zeta^{(i)}$ for $i \in \{1, \ldots, n\}$ separately, based on the observation that at every iteration $k = 0, \ldots, K$ of algorithm 4, each sub-problem (lines 3,5) forms a *relaxed* MCKP equation 42. Thus replacing

- $\boldsymbol{u}$ in equation 42 with $\zeta^{(i)}$.
- p with $\left(p_f^{(i)}\right)^T + \sum_{j \in \{1, \ldots, i-1\}} \left(\zeta^{(j)}\right)^T \cdot Q_f^{(i,j)} + \sum_{j \in \{i+1, \ldots, n\}} \left(\zeta^{(j)}\right)^T \cdot \left(Q_f^{(i,j)}\right)^T$.
- The elements of t with the elements of
$$
\left(p_{\mathcal{M}}^{(i)}\right)^T + \sum_{j \in \{1, \ldots, i-1\}} \left(\zeta^{(j)}\right)^T \cdot Q_{\mathcal{M}}^{(i,j)} + \sum_{j \in \{i+1, \ldots, n\}} \left(\zeta^{(j)}\right)^T \cdot \left(Q_{\mathcal{M}}^{(i,j)}\right)^T
$$
- 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 31 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.

H On Transitivity of Ranking Correlations

While the predictors in section 3.2 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:

$$
|Cor(P, S) - Cor(P, O) \cdot Cor(O, S)| \leq \sqrt{(1 - Cor(P, O)^2) \cdot (1 - Cor(O, S)^2)} \tag{43}
$$

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 $\rho_{P,S} \geq 0.93$.