Supplementary Material

In the supplement material, we elaborate our formal arguments and provide additional results and ablations.

7. Fisher Information

In this section, we provide a more detailed inspection on the Fisher Information matrix (FIM) in the context of neural networks as an extension of Sec. 2.1. The Section 7.1 extends some results from the main paper, the Section 7.2 provides another motivation on why the FIM should be considered as a tool for analysis of neural networks and finally Section 7.3 summarizes terminology issues within the community.

7.1. Cramér-Rao bound

Consider the same setting as in Sec. 2.1, that is a deep network f is given with the unknown (deterministic) weight vector $\theta \in \mathbb{R}^p$ for some parameters $p \in \mathbb{N}$, whose estimation is the subject of our interest. Take any estimator $\hat{\theta}_n$ that is computed from n independently-drawn input images and which is unbiased, i.e.

$$\mathbb{E}\hat{\theta}_n = \theta. \tag{16}$$

Then we have the lower bound for the variance matrix of $\hat{\theta}_n$ given as

$$\text{Var}\left(\hat{\theta}_n\right) \geq \frac{1}{n} F^{-1}(\theta), \tag{17}$$

which in turn gives also an estimate for the diagonal elements

$$\left(\text{Var}\left(\hat{\theta}_n\right)\right)_{jj} \geq \frac{1}{n}\left(F^{-1}(\theta)\right)_{jj}, \tag{18}$$

where we use the standard notation A_{ij} for the entry at the position i, j for any matrix A.

We now show the relation between (17) and mean-square error of the weight estimator. First, consider a weight $\theta(j)$ for some $j \in \{1, \ldots, p\}$. Then we shall write $\theta(j) = e_j^T \theta$, where e_j is the jth unit vector consisting only of zeros and single one at the jth position: $e_j = (0, \ldots, 1, \ldots, 0)$. Now recall that that if a random d-dimensional vector X has a variance matrix $\text{Var}(X)$ and $e \in \mathbb{R}^d$ then the linear combination $e^T X$ has the variance

$$\text{Var}\left(e^T X\right) = e\text{Var}(X)\, e^T, \tag{19}$$

from which it easily follows that the variance of the random scalar $\hat{\theta}_n(j)$ is

$$\begin{aligned}\text{Var}\left(\hat{\theta}_n(j)\right) =& \text{Var}\left(e_j^T \hat{\theta}_n\right) \\ =& e_j \text{Var}\left(\hat{\theta}_n\right) e_j^T \\ =& \left(\text{Var}\left(\hat{\theta}_n\right)\right)_{jj}.\end{aligned} \tag{20}$$

Next, we use the bias-variance decomposition of the mean-square error: if $\hat{X}$ is an estimator of an unknown scalar value $X \in \mathbb{R}$, then

$$\begin{aligned}\mathbb{E}\left(\hat{X} - X\right)^2 =& \mathbb{E}\left(\hat{X} - \mathbb{E}\hat{X} + \mathbb{E}\hat{X} - X\right)^2 \\ =& \mathbb{E}\left(\hat{X} - \mathbb{E}\hat{X}\right)^2 + 2\mathbb{E}\left(\hat{X} - \mathbb{E}\hat{X}\right)\mathbb{E}\left(\hat{X} - X\right) \\ &+ \mathbb{E}\left(\mathbb{E}\hat{X} - X\right)^2 \\ =& \text{Var}\hat{X} + \left(\mathbb{E}\hat{X} - X\right)^2 \\ =& \text{Var}\hat{X} + \left(\text{Bias}\hat{X}\right)^2.\end{aligned} \tag{21}$$

Combining (16) with (20), (21) and the Cramér-Rao bound (17) we obtain

$$\begin{aligned}\mathbb{E}\left(\hat{\theta}_n(j) - \theta(j)\right)^2 =& \text{Var}\left(\hat{\theta}_n(j)\right) + \left(\text{Bias}\left(\hat{\theta}_n(j)\right)\right)^2 \\ =& \text{Var}\left(\hat{\theta}_n(j)\right) \\ =& \left(\text{Var}\left(\hat{\theta}_n\right)\right)_{jj} \\ \geq& \frac{1}{n}\left(F^{-1}(\theta)\right)_{jj}.\end{aligned} \tag{22}$$

Summing now over all indices j we finally obtain a lower bound of for the mean-square error of the entire weight vector $\hat{\theta}_n$

$$\begin{aligned}\mathbb{E}\|\theta_1 - \theta_2\|^2 =& \sum_{j=1}^{p} \mathbb{E}\left(\hat{\theta}_n(j) - \theta(j)\right)^2 \\ \geq& \frac{1}{n}\sum_{j=1}^{p}\left(F^{-1}(\theta)\right)_{jj}.\end{aligned} \tag{23}$$

The quantity on the right-hand-side of (23) is the trace of the matrix $F^{-1}(\theta)$ and it coincides with sum of the eigenvalues of $F^{-1}(\theta)$, which are just the reciprocals of the eigenvalues of $F(\theta)$ [24].

We have shown that eigenvalues of the FIM determine the least-possible mean-square error for any unbiased estimator of the network weight vector θ and its components. The case of a biased estimator is more delicate and we kindly refer the reader to [11].

7.2. Natural Gradient Descent

Natural Gradient Descent (see [29]) is an improvement of the classical Stochastic Gradient Descent that is proven to have faster and more stable convergence, but for the price of significantly increased computation costs. In Natural Gradient Descent, the weight updates are governed by a transformed loss gradient as

$$\theta_{n+1} = \theta_n - F^{-1}(\theta_n)\nabla_\theta \mathcal{L}(\theta_n), \tag{24}$$

where $F^{-1}(\theta)$ is the inverse of the FIM and $\mathcal{L}(\theta)$ is the loss function.

We also take the steepest descent direction of the loss function, but now we do not measure the distance in the space of weights by means of the Euclidean distance but we adjust the curvature by measuring the KL-divergence of the output distributions. In other words, Natural Gradient Descent is just what happens to Stochastic Gradient Descent if we say that two weight vectors θ_1, θ_2 are close to each other if

$$D_{KL}(\sigma_{\theta_1}(\cdot|x), \sigma_{\theta_2}(\cdot|x)) \tag{25}$$

is small, in contrast to the usual case when we consider $\|\theta_1 - \theta_2\|$ instead. And again similarly as above, after inspecting the eigenvalues of the FIM we can conclude that the more different the eigenvalues are the more difficult is to train the model as the weight updates are much larger in some directions than in others.

Even though we later use the FIM in the applications where the networks have been trained with the classical Stochastic Gradient Descent, the curvature given by the FIM still provides a valuable information – if the network is more difficult to train using Natural Gradient Descent, it's unlikely that when using a simpler optimisation method, the network would yield stronger performance after training.

7.3. Monte Carlo estimation of the Fisher Information Matrix (FIM)

We now follow [19] and outline details of the common misconception in the terminology within the community that might lead to incorrect estimation of the FIM.

In our setting, the FIM is given as

$$F(\theta) \coloneqq \mathbb{E}\left[\nabla_\theta \sigma_\theta(c\,|\,x)\, \nabla_\theta \sigma_\theta(c\,|\,x)^T\right] \in \mathbb{R}^{p\times p}, \tag{26}$$

where the expectation $\mathbb{E}$ is taken with respect to the join distribution of the image-label pair (x, c). Recall that the joint distribution can be decomposed into the prior distribution for x, usually unknown, and the conditional distribution distribution for the label c given x

$$\sigma_\theta(c\,|\,x) = \frac{\exp\left(\Psi_c(x,\theta)\right)}{\sum_{d=1}^{C}\exp\left(\Psi_d(x,\theta)\right)}, \quad c = 1,\dots,C \tag{27}$$

where $\Psi(x,\theta) \in \mathbb{R}^C = (\Psi_1(x,\theta),\dots,\Psi_C(x,\theta))$ is the network output (logits) given the weight vector $\theta \in \mathbb{R}^p$, i.e. the network weights. We might deal with missing information on the prior distribution of x by simply replacing it with the empirical distribution given by independently drawn examples $x_1,\dots,x_n$ which then yields a Monte Carlo estimate

$$\hat{F}(\theta) \coloneqq \frac{1}{n}\sum_{n=1}^{N} \mathbb{E}_{\sigma_\theta}\left[\nabla_\theta \sigma_\theta(c\,|\,x_n)\, \nabla_\theta \sigma_\theta(c|\,x_n)^T\right] \tag{28}$$

where $\mathbb{E}_{\sigma_\theta}$ now denotes the expectation with respect to the model prediction σ_θ, which is in the statistical community denoted as the empirical FIM. From the strong law of large numbers it follows that the empirical FIM converges to the FIM almost surely as the number of samples n tends to infinity. Therefore, it is reasonable to replace (26) in the applications by (28).

However, in some methods (see [19] and references therein) the expectation in (28) with respect to the model prediction σ_θ is often replaced by the empirical distribution σ of the labels given the images which leads to a different definition

$$\begin{aligned} G(\theta) &\coloneqq \frac{1}{n}\sum_{n=1}^{N} \mathbb{E}_{\sigma}\left[\nabla_\theta \sigma_\theta(c\,|\,x_n)\, \nabla_\theta \sigma_\theta(c|\,x_n)^T\right] \\ &= \frac{1}{n}\sum_{n=1}^{N} \left[\nabla_\theta \sigma_\theta(c_n\,|\,x_n)\, \nabla_\theta \sigma_\theta(c_n|\,x_n)^T\right], \end{aligned} \tag{29}$$

where now $(x_j\, c_j)$ are the observed image-label pairs. The difference between (28) and (29) is that in the former we sum the multiplied gradients over all categories c weighted by the network-predicted probability, while in the later we use only single class as if the network correctly classified the sample with zero error. At the initialization stage, the network prediction is however far from the ground-truth distribution and therefore $G(\theta)$ is indeed very different from the Monte Carlo approximation $\hat{F}(\theta)$ (and also from the FIM $F(\theta)$ itself). In our method we used $\hat{F}(\theta)$.

8. Experiments

NAS-Bench-201. In Tab. 1, we provide results for the NAS-Bench-201 architecture search space where we adopted the practice of NAS-Bench-101 [14, 30, 46] where only unique graph structures are considered. As we described in Sec. 5, the entire search space in NAS-Bench-201 contains also networks where some computation edges don't receive any input or their output cannot be propagated through the network due to the existence of zero operation nodes. By filtering these networks, the number of architectures drops from 15,625 to 9,445 unique architectures. We argue that this is indeed good practice as networks with unreachable parameters should not be used in practice as the energy costs rise without improved performance.

	Type	CIFAR-10			CIFAR-100			ImageNet16-120		
		KT	SPR	nDCG	KT	SPR	nDCG	KT	SPR	nDCG
Simple rankings										
FLOPs	S	0.578	0.753	0.729	0.551	0.727	0.565	0.517	0.691	0.386
GradNorm [1]	S	0.356	0.483	0.407	0.359	0.489	0.202	0.322	0.441	0.192
GraSP [1, 42]	S	0.315	0.454	0.439	0.322	0.461	0.224	0.333	0.470	0.207
SNIP [1, 23]	S	0.454	0.615	0.433	0.462	0.620	0.221	0.403	0.539	0.212
SynFlow [1, 41]	S	0.571	0.769	0.691	0.565	0.761	0.584	0.555	0.747	0.504
Jacov [1]	S	0.545	0.712	0.362	0.554	0.720	0.249	0.537	0.701	0.240
NASWOT [31]	S	0.556	0.742	0.572	0.579	0.768	0.449	0.583	0.768	0.459
ZenNAS [26]	S	0.244	0.321	0.110	0.232	0.300	0.110	0.250	0.344	0.065
GradSign† [49]	S	·	0.765	·	·	0.793	·	·	0.783	·
ZiCo [25]	S	0.590	0.785	0.732	0.600	0.794	0.597	0.594	0.787	0.516
TE-NAS [6]	A	0.489	0.676	0.481	0.481	0.664	0.214	0.459	0.641	0.143
AZ-NAS [21]	A	0.739	**0.912**	0.702	0.722	0.899	0.473	0.694	0.876	0.482
No. of trainable layers ($\aleph$)	S	0.580	0.723	0.631	0.594	0.737	0.491	0.574	0.716	0.483
$\text{VKDNW}_{\text{single}}$ (ours)	S	0.606	0.800	0.724	0.613	0.807	0.592	0.605	0.795	0.583
$\text{VKDNW}_{\text{agg}}$ (ours)	A	**0.740**	0.911	**0.743**	**0.736**	**0.906**	**0.578**	**0.723**	**0.893**	**0.614**
Model-driven rankings										
GRAF [14]	A	0.832	0.957	0.921	0.818	0.952	0.859	0.812	0.946	0.832
VKDNW_m (ours)	A	0.682	0.864	0.757	0.655	0.840	0.573	0.642	0.826	0.506
$(\text{VKDNW+ZCS})_m$ (ours)	A	0.865	0.973	0.906	0.859	0.970	0.861	0.863	0.971	0.828
$(\text{VKDNW+ZCS+GRAF})_m$ (ours)	A	**0.878**	**0.978**	**0.927**	**0.869**	**0.975**	**0.871**	**0.874**	**0.975**	**0.856**

Table 6. Training-free NAS methods in the NAS-Bench-201 [10] search space with inaccessible nodes (see discussion in Sec. 5.2, evaluated on three public datasets. Kendall's τ (KT), Spearman's ρ (SPR) and Normalized Discounted Cumulative Gain (nDCG) are reported, results are averages of 5 independent runs. The Type column differentiates single (S) and aggregated (A) rankings. Results are reproduced with code published by their authors, except those marked†, where results from the original paper are taken.

Moreover, many of the ranking scores (such as FLOPs or #params) do not make sense in such cases, because parameters/operations are not used in network output yet they are still included in these metrics.

For the sake of completeness however, in Table 6 we provide results of our experiments on full NAS-Bench-201 search space, using all 15,625 architectures. We can see that both $\text{VKDNW}_{\text{single}}$ and $\text{VKDNW}_{\text{agg}}$ outperform all other simple rankings in all metrics on CIFAR-100 and ImageNet16-120. On CIFAR-10 dataset AZ-NAS[21] achieves similar Kendall's τ and Spearman's ρ correlations as $\text{VKDNW}_{\text{agg}}$, however $\text{VKDNW}_{\text{agg}}$ leads in nDCG with a considerable margin. In Table **??** we also present the accuracies of the top networks chosen by training-free methods.

MobileNetV2. In this experiment, we search for the best network configuration in the MobileNetV2 space [38]. The search space is much larger as it consists of different architectures with inverted residual blocks, where depth, width, and expansion ratio of the blocks is altered. We constrained the model size to approximately 450M FLOPs and number of layers to 14. We adapt the evolutionary search algorithm [21] by replacing the objective function in the search algorithm with our $\text{VKDNW}_{\text{agg}}$. We then ran 100,000 iterations of the algorithm, always keeping top 1,024 best architectures, measured by $\text{VKDNW}_{\text{agg}}$. In each iteration, one mutation operation randomly changes one element in one of the top 1,024 architectures, and the newly created architecture is again ranked using $\text{VKDNW}_{\text{agg}}$. As a result, 100,000 iterations of architecture evaluations were made in the search process, leaving us with a shortlist of 1,024 architectures in the end.

Out of these final 1,024 architectures, we then again picked the one with the highest $\text{VKDNW}_{\text{agg}}$ rank and trained it for 480 epochs on ImageNet-1K [9] in the same teacher-student setting as [21, 25]. We used vanilla SGD optimizer with LR=0.2 and single-cycle cosine learning rate schedule. The final training of the model took 7 days on 8xNVidia A100 GPUs.

9. Ablations

Fisher Information matrix size. In Tab. 8, we evaluate our method with varying number of trainable layers considered in the computation of the FIM (see Eq. (2)). We can see that initial 16 layers of the network already carry enough information, even comparable to when we use 256 layers. In our method, we set this parameter to 128 to maximize for (nDCG) while keeping other metrics high.

	CIFAR-10	CIFAR-100	ImageNet16-120
FLOPs	93.76 ± 0.15	71.11 ± 0.28	41.44 ± 0.72
ZiCo [25]	93.80 ± 0.18	71.21 ± 0.30	42.12 ± 0.79
TE-NAS [6]	92.47 ± 0.30	67.20 ± 1.01	39.22 ± 1.99
AZ-NAS [21]	93.51 ± 0.15	71.04 ± 0.61	45.57 ± 0.48
$\text{VKDNW}_{\text{single}}$	93.50 ± 0.18	70.94 ± 0.48	44.20 ± 0.49
$\text{VKDNW}_{\text{agg}}$	93.46 ± 0.15	70.91 ± 0.28	44.51 ± 0.41
GRAF [14]	94.10 ± 0.20	72.78 ± 0.45	45.94 ± 0.42
VKDNW_m	93.57 ± 0.12	70.94 ± 0.20	43.64 ± 0.22
$(\text{VKDNW+ZCS})_m$	94.07 ± 0.11	**72.79** ± 0.21	**46.48** ± 0.23
$(\text{VKDNW+ZCS+GRAF})_m$	**94.19** ± 0.11	72.58 ± 0.20	46.29 ± 0.25

Table 7. Training-free NAS methods in the NAS-Bench-201 [10] search space, evaluated on three public datasets. Accuracy of top model chosen by given method is shown. Results are averages over 5 independent runs.

FIM Dimension	KT	SPR	nDCG
8	0.590	0.782	0.579
16	0.619	0.810	0.592
32	0.621	0.813	0.600
64	0.621	0.814	0.607
128	0.619	0.812	0.611
256	0.619	0.812	0.606

Table 8. Our method $\text{VKDNW}_{\text{single}}$ evaluated for different FIM (see Sec 2.1) sizes with respect to Kendall's τ (KT), Spearman's ρ (SPR) and Normalized Discounted Cumulative Gain with $P = 1000$ (nDCG).

One weight per layer			
Policy	KT	SPR	nDCG
random	0.618	0.811	0.586
0	0.622	0.814	0.608
0.2	0.622	0.815	0.602
0.4	0.625	0.817	0.606
0.6	0.623	0.814	0.598
0.8	0.622	0.814	0.609
1	0.621	0.813	0.608
Multiple weights per layer			
No. weights	KT	SPR	nDCG
1	0.621	0.813	0.600
2	0.634	0.824	0.608
4	0.626	0.817	0.600
8	0.605	0.797	0.594

Table 9. Our method $\text{VKDNW}_{\text{single}}$ evaluated for different parameter sampling policies within each trainable layer. Two types of sampling methods are presented. In *One weight per layer* we take 128 initial network layers and either sample one weight per layer randomly or we take for $p = 0, 0.2, \ldots, 1$ the pth index relative within the weight vector. In *Multiple weights per layer* we take 32 initial network layers and sample uniformly k weights for $k = 1, 2, 4, 8$. We evaluate Kendall's τ (KT), Spearman's ρ (SPR) and Normalized Discounted Cumulative Gain with $P = 1000$ (nDCG).

Parameter sampling policy. To make the dimension of the FIM feasible for computation of eigenvalues, we use only a small portion of the network weights. More specifically, instead of taking the full matrix of dimension p (number of trainable parameters), we only sample one weight from each trainable layer from the first 128 layers and compute the FIM as if the network did not have any other parameters. In Table 9, we compare performance of our method $\text{VKDNW}_{\text{single}}$ as we vary the number of weights per layer and their sampling policy. We can see that the performance as measured by (nDCG) is roughly the same when taking anything between one and four weights per layer, and then starts to slowly decrease with a higher number of weights per layer. Secondly, our method is robust against choice of the policy as the performance for the case of one weight per layer with changing position of the weight within each layer does not change significantly. To further show that we do not lose any performance when dealing only with limited number of initial layers, we show in Tab. 8 that our method is also robust against change of number of considered layers (the highest number of layers we tested was 256 as the number of larger networks in NAS-Bench-201 is small).

Orthogonality of VKDNW. Our score VKDNW is based on information orthogonal to the size of the network: in Fig. 3, we show that unlike previous work, VKDNW is not correlated with the network size measured by $\aleph$ (number of trainable layers). In Fig. 4, we present similar results where we now measure the network size by the number of trainable parameters. We can see that VKDNW keeps the orthogonality property even after change of the size proxy.

Components of the aggregated rank. Our aggregated rank $\text{VKDNW}_{\text{agg}}$ combines information from 5 different sources: our $\text{VKDNW}_{\text{single}}$, Jacov, expressivity, trainability and FLOPs (see Sec. 5.1). In Table 10, all 2^5 combinations of keeping/dropping every of the 5 sources are evaluated on ImageNet16-120. We can see that our ranking $\text{VKDNW}_{\text{single}}$ is the strongest component as it has the highest marginal performance in all three considered metrics. The lowest performance drop is observed for expres-

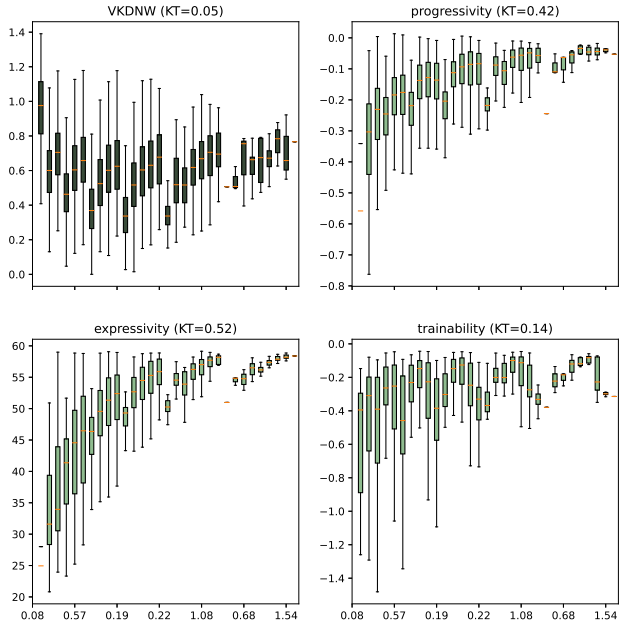

Figure 4. Components of AZ-NAS [21] and our VKDNW are compared w.r.t. correlation with number of model parameters, in the NAS-Bench-201 search space [10] on ImageNet16-120 [7] dataset. Our VKDNW proxy has the lowest correlation, ie. is the most invariant to the size of the model.

sivity: without this component the method would even perform better in the (nDCG) metric than the original variant $\text{VKDNW}_{\text{agg}}$. We decided to include expressivity in the final ranking as we optimized for all three metrics (KT), (SPR) and (nDCG) simultaneously.

V	J	E	T	F	(KT)	(SPR)	(nDCG)
✓					**0.622**	**0.814**	**0.608**
	✓				0.603	0.781	0.339
		✓			0.588	0.779	0.274
			✓		0.353	0.517	0.233
				✓	0.545	0.718	0.403
✓	✓				0.677	0.851	0.565
✓		✓			0.675	0.851	0.489
✓			✓		0.622	0.821	0.552
✓				✓	0.619	0.811	0.557
	✓	✓			0.630	0.815	0.349
	✓		✓		0.621	0.818	0.505
	✓			✓	0.695	0.863	0.574
		✓	✓		0.616	0.818	0.463
		✓		✓	0.642	0.818	0.434
			✓	✓	0.617	0.815	0.580
✓	✓		✓		0.698	0.879	0.632
✓	✓	✓			0.686	0.858	0.515
✓	✓			✓	0.696	0.868	0.616
✓		✓	✓		0.698	0.882	0.612
✓		✓		✓	0.672	0.848	0.512
✓			✓	✓	0.681	0.870	0.675
	✓	✓	✓		0.674	0.862	0.527
	✓	✓		✓	0.695	0.859	0.484
	✓		✓	✓	0.726	0.899	0.673
		✓	✓	✓	0.702	0.883	0.630
✓	✓	✓	✓		0.717	0.891	0.623
✓	✓	✓		✓	0.706	0.871	0.553
✓	✓		✓	✓	0.736	0.905	0.695
✓		✓	✓	✓	0.722	0.896	0.658
	✓	✓	✓	✓	0.735	0.901	0.646
✓	✓	✓	✓	✓	**0.743**	**0.906**	**0.664**

Table 10. Components of $\text{VKDNW}_{\text{agg}}$ rank with non-linear aggregation. Consistency is shown with respect to Kendall's τ (KT), Spearman's ρ (SPR) and Normalized Discounted Cumulative Gain (nDCG) with $P = 1000$ on ImageNet16-120 image dataset [7]. Here V, J, E, T and F stand for $\text{VKDNW}_{\text{single}}$, Jacov, expressivity, trainability and FLOPs respectively (see Sec. 5.1).