A Detailed proofs

A.1 Proof of Theorem 2

In this section, we present the proof for our main theorem (Theorem 2 in Section 3.4). We first introduce the following definitions and lemmas to better illustrate our proof.

Lemma 1 (Lemma 3.1 in [1]). *If* $m = \Omega\left(\frac{n^2}{\lambda_0^2}\log\left(\frac{n}{\delta}\right)\right)$, *then with probability at least* $1-\delta$, $\|\mathbf{H}(0)-\mathbf{H}^{\infty}\|_2 \leq \frac{\lambda_0}{4}$ *and* $\frac{3}{4}\lambda_0 \leq \lambda_{\min}(\mathbf{H}(0)) \leq \frac{5}{4}\lambda_0$.

Definition 1. *We denote the empirical indicator matrix as* $\mathbf{W}(0)$ *with entry* $[\mathbf{W}(0)]_{ij}$ *such that*

$$[\mathbf{W}(0)]_{ij} := \frac{1}{m}\sum_{r=1}^{m}\mathbb{I}\{\mathbf{w}_r^T(0)\mathbf{x}_i \geq 0, \mathbf{w}_r^T(0)\mathbf{x}_j \geq 0\}$$

and indicator matrix as $\bar{\mathbf{W}}$ *with entry* $[\bar{\mathbf{W}}]_{ij}$

$$[\bar{\mathbf{W}}]_{ij} := \mathbb{E}_{\mathbf{w}\sim N(\mathbf{0},\mathbf{I})}[\mathbb{I}\{\mathbf{w}^T\mathbf{x}_i \geq 0, \mathbf{w}^T\mathbf{x}_j \geq 0\}]$$

The indicator matrix $\bar{\mathbf{W}}$ has the following property.

Proposition 1. *If* $\mathbf{x}_i \nparallel \mathbf{x}_j$, *then* $[\bar{\mathbf{W}}]_{ij} > 0$.

Proof. If Proposition 1 does not hold, then $\exists i_0, j_0 \in [n]\times[n]$, satisfying

$$\begin{aligned}0 =&[\bar{\mathbf{W}}]_{i_0 j_0}\\ =&\mathbb{E}_{\mathbf{w}\sim N(\mathbf{0},\mathbf{I})}[\mathbb{I}\{\mathbf{w}^T\mathbf{x}_{i_0} \geq 0, \mathbf{w}^T\mathbf{x}_{j_0} \geq 0\}]\\ =&P_{\mathbf{w}\sim N(\mathbf{0},\mathbf{I})}(\mathbf{w}^T\mathbf{x}_{i_0} \geq 0, \mathbf{w}^T\mathbf{x}_{j_0} \geq 0)\end{aligned}$$

Thus $\mathbf{x}_{i_0} = -\mathbf{x}_{j_0}$, which contradicts the hypothesis $\mathbf{x}_{i_0} \nparallel \mathbf{x}_{j_0}$. ■

In real-world datasets, the possibility of two different data being parallel is slight. Thus, proposition 1 holds in general. We further introduce Weyl inequality as follows:

Lemma 2 (Weyl inequality [2]). *Let* $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ *be Hermitian matrices, and let the eigenvalues of* $\mathbf{A}$, $\mathbf{B}$, *and* $\mathbf{A}+\mathbf{B}$ *be* $\{\lambda_i(\mathbf{A})\}_{i=1}^n$, $\{\lambda_i(\mathbf{B})\}_{i=1}^n$ *and* $\{\lambda_i(\mathbf{A}+\mathbf{B})\}_{i=1}^n$, *respectively. The eigenvalues of each matrix are arranged in ascending order. Then we have*

$$\lambda_i(\mathbf{A}+\mathbf{B}) \leq \lambda_{i+j}(\mathbf{A}) + \lambda_{n-j}(\mathbf{B}), \quad j = 0, 1, \ldots, n-i \tag{1}$$

for each $i = 1, \ldots, n$, *with equality for some pair* i, j *if and only if there is a nonzero vector* $\mathbf{x}$ *such that* $\mathbf{A}\mathbf{x} = \lambda_{i+j}(\mathbf{A})\mathbf{x}$, $\mathbf{B}\mathbf{x} = \lambda_{n-j}(\mathbf{B})\mathbf{x}$, *and* $(\mathbf{A}+\mathbf{B})\mathbf{x} = \lambda_i(\mathbf{A}+\mathbf{B})\mathbf{x}$. *Also,*

$$\lambda_{i-j+1}(\mathbf{A}) + \lambda_j(\mathbf{B}) \leq \lambda_i(\mathbf{A}+\mathbf{B}), \quad j = 1, \ldots, i \tag{2}$$

for each $i = 1, \ldots, n$, *with equality for some pair* i, j *if and only if there is a nonzero vector* $\mathbf{x}$ *such that* $\mathbf{A}\mathbf{x} = \lambda_{i-j+1}(\mathbf{A})\mathbf{x}$, $\mathbf{B}\mathbf{x} = \lambda_j(\mathbf{B})\mathbf{x}$, *and* $(\mathbf{A}+\mathbf{B})\mathbf{x} = \lambda_i(\mathbf{A}+\mathbf{B})\mathbf{x}$. *If* $\mathbf{A}$ *and* $\mathbf{B}$ *have no common eigenvector, then inequality (1) and (2) are strict inequality.*

Now, we provide the full proof of Theorem 2.

Theorem 2. Suppose f is an NN with a single hidden layer and ReLU activation function. Assume $\mathbf{X} \in \mathbb{R}^{d\times n}$, $\mathbf{w}(0) \sim N(\mathbf{0}, \mathbf{I})$, $P(\mathbf{X}) \succ 0$, $p_0 := \lambda_{\min}(P(\mathbf{X}))$, and hidden nodes $m = \Omega\left(\frac{n^6 d^2}{\lambda_0^4 \delta^3}\right)$, then the following formula holds with probability at least $1-\delta$ over the initialization

$$\|f(\mathbf{W}(t), \mathbf{a}, \mathbf{X}) - \mathbf{y}\|_2^2 \leq \exp\left(-cp_0 t\right)\|f(\mathbf{W}(0), \mathbf{a}, \mathbf{X}) - \mathbf{y}\|_2^2$$

where c is a constant depending on m and d.

Proof. To simplify our proof, we assume $\mu_{\mathbf{x}} = 0$, $\sigma_{\mathbf{x}} = 1$, and $\|\mathbf{x}\|_2 \leq C$ for all $\mathbf{x} \in \mathbf{X}$. Recall that

$$[\mathbf{H}(0)]_{ij} = \mathbf{x}_i^T \mathbf{x}_j \frac{1}{m} \sum_{r=1}^{m} \mathbb{I}\{\mathbf{w}_r^T(0)\mathbf{x}_i \geq 0, \mathbf{w}_r^T(0)\mathbf{x}_j \geq 0\} = \mathbf{x}_i^T \mathbf{x}_j [\mathbf{W}(0)]_{ij}$$

Due to $\mathbb{I}\{\mathbf{w}_r^T(0)\mathbf{x}_i \geq 0, \mathbf{w}_r^T(0)\mathbf{x}_j \geq 0\}$ is an independent random variable between 0 and 1, then by Hoeffding's inequality [3], the following inequality holds with probability $1 - \delta$:

$$[\mathbf{W}(0)]_{ij} \geq [\bar{\mathbf{W}}]_{ij} - \frac{2\sqrt{\log(1/\delta)}}{\sqrt{m}}$$

Let $\mu_0 = \min_{(i,j)\in[n]\times[n]} [\bar{\mathbf{W}}]_{ij}$, and choose $m > \frac{16 \log(1/\delta)}{\mu_0^2}$, then we have

$$[\mathbf{W}(0)]_{ij} \geq \mu_0 - \frac{2\sqrt{\log(1/\delta)}}{\sqrt{m}} \geq \frac{2\sqrt{\log(1/\delta)}}{\sqrt{m}} =: c(m)$$

Define the matrix $\mathbf{M}$ with entry $[\mathbf{M}]_{ij}$ as

$$[\mathbf{M}]_{ij} := [\mathbf{H}(0)]_{ij} - c(m,d)[P(\mathbf{X})]_{ij}$$

where $c(m,d) = c(m)(d-1)$. We claim that if m is large enough, then $\mathbf{M}$ is positive definite. To clarify this statement, we consider the gap between $\mathbf{H}(0)$ and $\mathbf{M}$:

$$\|\mathbf{H}(0) - \mathbf{M}\|_2 = \|c(m,d)P(\mathbf{X})\|_2 \leq c(m,d)\|P(X)\|_F \leq c(m,d)n^2C^2$$

If we choose $m > \frac{64 \log(1/\delta)(d-1)^2 n^4 C^4}{\lambda_0^2}$, we have

$$\|\mathbf{H}(0) - \mathbf{M}\|_2 \leq \frac{\lambda_0}{4}$$

Then the following formula holds by matrix perturbation theory (Corollary 6.3.8 in [2])

$$0 < \frac{\lambda_0}{2} \leq \lambda_{\min}(\mathbf{H}(0)) - \frac{\lambda_0}{4} \leq \lambda_{\min}(\mathbf{M})$$

which indicates that $\mathbf{M}$ is positive definite. We next exert Lemma 2 on $\mathbf{M}$ and get

$$0 < \lambda_{\min}(\mathbf{H}(0) - c(m,d)P(\mathbf{X})) < \lambda_{\min}(\mathbf{H}(0)) - \lambda_{\min}(c(m,d)\mathbf{P}(\mathbf{X}))$$

That means

$$0 < c(m,d)p_0 \leq \lambda_{\min}(\mathbf{H}(0)) \leq \frac{5}{4}\lambda_0 \tag{3}$$

Therefore, combined with Theorem 1, we have

$$\begin{aligned}
&\|f(\mathbf{W}(t), \mathbf{a}, \mathbf{X}) - \mathbf{y}\|_2^2 \\
\leq &\exp(-\lambda_0 t)\|f(\mathbf{W}(0), \mathbf{a}, \mathbf{X}) - \mathbf{y}\|_2^2 \\
\leq &\exp(-c(m,d)p_0 t)\|f(\mathbf{W}(0), \mathbf{a}, \mathbf{X}) - \mathbf{y}\|_2^2
\end{aligned}$$

where $c(m,d)$ is a constant depending on m and d. This completes the proof of **Theorem 2**. ■

A.2 Proof of Theorem 3

Proof. By Corollary 3.10 and Remark 3.11 in [4], we have

$$\mathbb{E}[L(W)] \leq O\left(c \cdot \sqrt{\frac{\mathbf{y}^T(\mathbf{H}^\infty)^{-1}\mathbf{y}}{N}}\right) + O\left(\sqrt{\frac{\log(1/\delta)}{N}}\right)$$

According to the courant minimax principle[5], D.2 in [6], and inequality 3, we get

$$\mathbf{y}^T(\mathbf{H}^\infty)^{-1}\mathbf{y} \leq \frac{\mathbf{y}^T\mathbf{y}}{\lambda_{\min}(\mathbf{H}^\infty)} \leq c \cdot \frac{\mathbf{y}^T\mathbf{y}}{p_0}.$$

Thus, we have

$$\begin{aligned}\mathbb{E}[L(W)] &\leq O\left(c \cdot \sqrt{\frac{\mathbf{y}^T(\mathbf{H}^\infty)^{-1}\mathbf{y}}{N}}\right) + O\left(\sqrt{\frac{\log(1/\delta)}{N}}\right) \\ &\leq O\left(c \cdot \sqrt{\frac{\mathbf{y}^T\mathbf{y}}{Np_0}}\right) + O\left(\sqrt{\frac{\log(1/\delta)}{N}}\right)\end{aligned}$$

■

B Zero-cost proxies

In this section, we provide the details of the baseline zero-cost proxies. Suppose L is the loss function and θ is the parameters of an initialized network. We denote $\circ$ as the Hadamard product. The concrete formulations of the existing zero-cost proxies are as follows.

- snip. Lee et al. [7] use the changes in loss caused by the parameter perturbations to measure the importance of the parameters in an initialized network, such that

$$\mathcal{S}_{\mathsf{snip}}(\theta) = \left|\frac{\partial L}{\partial \theta} \circ \theta\right|$$

- grasp. Wang et al. [8] replace the loss change in snip with the change of the gradient norm to establish the proxy, such that

$$\mathcal{S}_{\mathsf{grasp}}(\theta) = -(H\frac{\partial L}{\partial \theta}) \circ \theta$$

  where H is the Hessian.

- synflow. To avoid layer collapse, Tanaka et al. [9] utilize the product of all parameters in the network during the parameter perturbation to represent the loss, such that

$$\mathcal{S}_{\mathsf{synflow}}(\theta) = \frac{\partial L}{\partial \theta} \circ \theta$$

- grad_norm. Abdelfattah et al. [10] adopt the l_2 norm of the gradients in an initialized work as a proxy, such that

$$\mathcal{S}_{\mathsf{grad\_norm}} = \left\|\frac{\partial L}{\partial \theta}\right\|_2$$

- jacov/NWOT. Mellor et al. [11] use the correlation of activations within a network as a proxy to evaluate the performance of the network, such that

$$\mathcal{S}_{\mathsf{jacov}} = \log|\mathbf{K}_H|, \quad \mathbf{K}_H = \begin{pmatrix} N_A - d_H(\mathbf{c}_1, \mathbf{c}_1) & \cdots & N_A - d_H(\mathbf{c}_1, \mathbf{c}_N) \\ \vdots & \ddots & \vdots \\ N_A - d_H(\mathbf{c}_N, \mathbf{c}_1) & \cdots & N_A - d_H(\mathbf{c}_N, \mathbf{c}_N) \end{pmatrix}$$

where N_A is the number of rectified linear units, $d_H(\mathbf{c}_i, \mathbf{c}_j)$ represents the Hamming distance between two binary codes $\mathbf{c}_i$ and $\mathbf{c}_j$.

- NTK. Chen et al. [12] propose to use the condition number of NTK to measure the trainability of the networks, such that

$$\kappa_{\mathcal{N}} = \frac{\lambda_{\max}(\hat{\Theta}_{train})}{\lambda_{\min}(\hat{\Theta}_{train})}$$

where $\hat{\Theta}_{train}$ stands for NTK of the networks. In our paper, we calculate the Spearman correlation coefficient between $1/\kappa_{\mathcal{N}}$ and the test accuracy of the networks.

- zen. Lin et al. [13] propose Zen-Score, in which they design an efficient zero-cost proxy with Gaussian random inputs, such that

$$\mathcal{S}_{\mathsf{zen}} = \log\left(\mathbb{E}_{\mathbf{x},\epsilon}\|f(\mathbf{x};\theta) - f(\mathbf{x}+\alpha\epsilon;\theta)\|_F\right) + \sum_i \log\left(\sqrt{\sum_j \sigma_{i,j}^2/m}\right)$$

where $\sigma_{i,j}$ is the mini-batch standard deviation statistic of the j-th channel in BN.

- NASI. Shu et al. [14] propose NASI to evaluate the networks by approximating the trace of the NTK, such that

$$\mathsf{NASI} = m\gamma^{-1}\left\| b^{-1}\sum_{x\in\mathcal{X}_j} \nabla_{\theta_0(\mathcal{A})}\mathcal{L}_x \right\|_2^2$$

where $\mathcal{X}_j$ is a mini-batch of data with size $|\mathcal{X}_j| = b$.

- KNAS. Xu et al. [15] propose to use gradient kernel to evaluate the networks, such that

$$g = \frac{1}{n^2}\sum_{i,j}\left(\frac{\partial y_j^L(t)}{\partial \mathbf{w}(t)}\right)\left(\frac{\partial y_i^L(t)}{\partial \mathbf{w}(t)}\right)^T$$

where y^L is the output of L-th layer.

- GradSign. Zhang and Jia [16] propose GradSign, in which they analyze the sample-wise optimization landscape of the networks, such that

$$\mathsf{GradSign} = \sum_k |\sum_i sign([\nabla_\theta l(f_\theta(x_i), y_i)|_{\theta_0}]_k)|$$

- ZiCo. Li et al. [17] explore the effect of gradient properties on network performance. They use absolute mean and standard deviation values of gradients to evaluate network performance:

$$\mathsf{ZiCo} = \sum_{l=1}^{D}\log\left(\sum_{\omega\in\theta_l}\frac{\mathbb{E}[|\partial L(\mathbf{X}_i, \mathbf{y}_i;\theta)/\partial\omega|]}{\sqrt{Var(|\partial L(\mathbf{X}_i, \mathbf{y}_i;\theta)/\partial\omega|)}}\right), \quad i \in [N]$$

where N and D represent the number of batches and network layers, respectively. θ_l represents the parameters of the l-th layers.

C Algorithm of MeCo-based NAS

We slightly abuse the notation and denote MeCo as our proxy function for the network. We adopt the Zero-Cost-PT [18] to integrate MeCo to a zero-shot NAS. The algorithm of our MeCo-based NAS is summarized in Algorithm 1. We denote A_0 as an untrained supernet, e_t represents the t-th edge, which stands for mixed operation in the search cells, and $e_{t,k}$ is the k-th operation of t-th edge. We denote $\mathcal{E}$, $\mathcal{N}$, and $\mathcal{O}$ as the set of edges, nodes, and candidate operations in the search cells, respectively. For any node $n \in \mathcal{N}$, we use $\mathcal{E}(n)$ as the set of its input edges, and e_n^k is the k-th element of $\mathcal{E}(n)$. Note that we can use other zero-cost proxies as described in section B to replace MeCo in Algorithm 1.

Algorithm 1 MeCo-based NAS

Require: A_0: An untrained supernet; $\mathcal{E}$: The set of edges in search cells; $\mathcal{N}$: The set of nodes in search cells; $\mathcal{O}$: The set of candidate operations; N: the number of candidate networks.
Ensure: The best network A_{best}.

// Stage 1: Architecture Proposal
$C = \emptyset$;
for $i = 1; i \leq N; i++$ **do**
  for $j = 1; j \leq |\mathcal{E}|; j++$ **do**
    Randomly choose an un-discretized edge e_t
    Choose the best edge from the supernet, s.t.

$$e_{t,best} = \arg\min_{1 \leq k \leq |\mathcal{O}|} \mathsf{MeCo}(A_0/e_{t,k})$$

    Use operation $e_{t,best}$ to substitute e_t
  end for
  $A_{|\mathcal{E}|}$ consists of $\{e_{t,best} | 1 \leq t \leq |\mathcal{E}|\}$
  for $j = 1; j < |\mathcal{N}|, j++$ **do** ▷ prune the edges of the obtained architecture $A_{|\mathcal{E}|}$
    Randomly select an unselected node $n \in \mathcal{N}$
    for $k = 1; k < |\mathcal{E}(n)|; k++$ **do**
      Calculate MeCo of the architecture $A_{|\mathcal{E}|}/e_n^k$
    end for
    Retain edges e_n^1, e_n^2 with the 1st and 2nd best MeCo value, and remove the other edges
  end for
  Get the candidate networks A_i that consist of $\{e_{t,best} | 1 \leq t \leq \mathcal{E}\}$, and append it to the set C
end for
// Stage 2: Architecture Validation
Get the best network:

$$A_{best} = \arg\max_{1 \leq i \leq N} \mathsf{MeCo}(A_i),\ s.t.\ A_i \in C$$

D Experimental configurations

We use the same settings for the experiments as in [18]. We summarized the configurations of the searching phase and training phase on CIFAR-10 and CIFAR-100 in Table 1 and Table 2, respectively. For ZiCo, we use two mini-batch of data, which has a size of 64. For the other baseline proxies, we use one mini-batch of data. On the other hand, our MeCo only uses one random data $\mathbf{x} \in \mathbb{R}^{1\times3\times32\times32}$.

E More experimental results of ρ

E.1 MeCo on NASBench-201 and NASBench-301 with three extra datasets

We evaluate MeCo and $\mathsf{MeCo}_{\mathsf{opt}}$ on NASBench-201 and NASBench-301 with Spherical-CIFAR-100, NinaPro, and SVHN, respectively. We only use one random data $x \in \mathbb{R}^{3\times32\times32}$ for MeCo and $\mathsf{MeCo}_{\mathsf{opt}}$. The results are summarized in Table 3. Our MeCo achieves competitive results on most benchmarks and achieves the best results on NASBench-201-SVHN, where the Spearman correlation coefficient is $\rho = 0.88$.

Table 1: The settings of Zero-Cost-PT with all proxies in DARTS-CNN for CIFAR-10

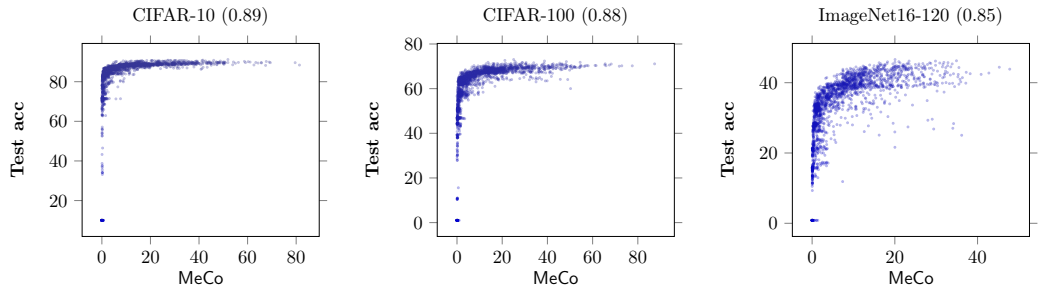
Settings	Searching phase		Training phase	
	Baselines	MeCo (Ours)	Baselines	MeCo (Ours)
batch size	64	1	96	96
cutout	True	False	True	True
cutout length	16	-	16	16
learning rate	0.025	0.025	0.025	0.025
learning rate min	0.001	0.001	-	-
momentum	0.9	0.9	0.9	0.9
weight decay	3e-4	3e-4	3e-4	3e-4
grad clip	5	5	5	5
init channels	16	16	36	36
layers	8	8	20	20
drop path prob	-	-	0.2	0.2

Table 2: The settings of Zero-Cost-PT with all proxies in DARTS-CNN for CIFAR-100

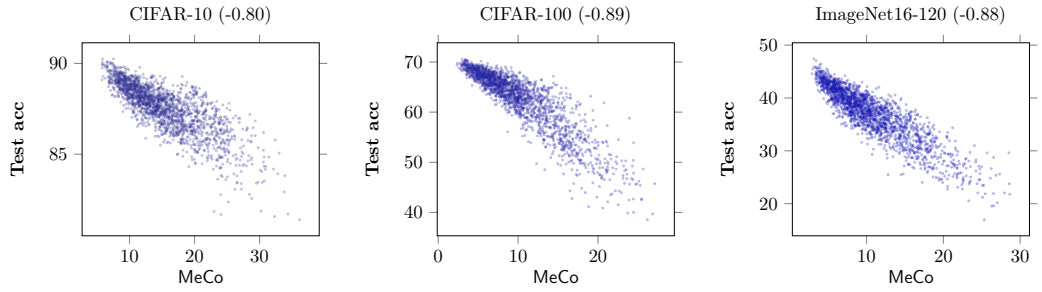
Settings	Searching phase		Training phase	
	Baselines	MeCo (Ours)	Baselines	MeCo (Ours)
batch size	64	1	96	96
cutout	True	False	True	True
cutout length	16	-	16	16
learning rate	0.025	0.025	0.025	0.025
learning rate min	0.001	0.001	-	-
momentum	0.9	0.9	0.9	0.9
weight decay	3e-4	3e-4	3e-4	3e-4
grad clip	5	5	5	5
init channels	16	16	16	16
layers	8	8	20	20
drop path prob	-	-	0.2	0.2

Table 3: Comparisons of ρ with baselines using NASBench-301 and NASBench-201 on three extra datasets

Method	**NASBench-301**			**NASBench-201**		
	Sph-Cifar100	**NinaPro**	**SVHN**	**Sph-Cifar100**	**NinaPro**	**SVHN**
grasp	0.13	0.04	0.18	-0.01	-0.01	0.62
fisher	0.00	-0.11	0.05	0.07	**-0.38**	0.71
grad_norm	-0.00	**-0.20**	0.42	-0.08	-0.23	0.77
snip	-0.01	-0.10	0.38	-0.09	-0.28	0.76
synflow	0.05	-0.07	0.50	0.13	0.02	0.71
l2_norm	**0.12**	-0.07	**0.70**	-0.00	0.02	0.67
#params	0.07	-0.07	**0.70**	-0.14	-0.11	0.72
zen	0.07	-0.09	0.68	0.23	0.15	0.18
jacov	0.08	0.13	-0.36	**-0.41**	0.29	0.67
nwot	0.05	0.02	0.64	-0.02	0.06	0.76
MeCo **(Ours)**	**-0.05**	**-0.11**	**0.68**	**-0.23**	**0.02**	**0.88**
MeCo_{opt} **(Ours)**	**0.03**	**0.12**	**0.68**	-	-	-



(a) Performance on NATS-Bench-TSS



(b) Performance on NATS-Bench-SSS

Figure 1: Relationship between our zero-cost proxies and test accuracy for NATS-Bench-TSS and NATS-Bench-SSS on three datasets.

E.2 MeCo on NAS-Bench-101

We present the Spearman correlation coefficient between all zero-cost proxies and the test accuracy of the networks in NAS-Bench-101. Our MeCo uses one data of CIFAR-10, i.e., $\mathbf{x} \in \mathbb{R}^{1\times3\times32\times32}$ while the baselines adopt a batch of samples as input. The results are summarized in Table 4.

Table 4: Spearman correlation coefficients ρ between zero-cost proxies and test accuracy on NAS-Bench-101

		Baselines							
Dataset	MeCo	grasp [8]	fisher[19]	grad_norm[10]	snip[7]	synflow [9]	jacov[11]	zen[13]	ZiCo [17]
CIFAR-10	0.44	-0.33	-0.27	-0.32	-0.25	0.36	-0.35	**0.63**	**0.63**

MeCo achieves competitive results on NAS-Bench-101 with CIFAR-10, i.e., the Spearman correlation coefficient obtains 0.44, which is higher than the majority of the baselines. However, zen and ZiCo perform better than MeCo.

E.3 MeCo on TransBench-101

We further evaluate our MeCo on diverse tasks. Specifically, we compare MeCo and the other proxies on Transbench-101-Micro and Transbench-101-Macro [20], respectively.

TransBench-101-Micro. We evaluate MeCo and MeCo_{opt} on TransBench-101-Micro with ten tasks. We calculate the Spearman correlation between our proxies and the accuracy of the networks. The results are summarized in Table 5. MeCo achieves the best performance on the three tasks, i.e., Class Objection, Spherical-Cifar100, and NinaPro, which are 0.58, 0.85, and 0.47, respectively. It can be seen that MeCo_{opt} effectively improves the performance of MeCo. For example, MeCo_{opt} improves MeCo from 0.62 to 0.77 on segmentation tasks.

TransBench-101-Macro. We compare MeCo, MeCo_{opt} and the baselines on TransBench-101-Macro with seven tasks. The results are summarized in Table 6. Our proxies achieve the best performance on Autoencoding, Jigsaw, and Surface Normal, which are 0.74, 0.48, and 0.80, respectively. Our proxies also achieve competitive results on the remaining tasks. Experimental results show that our proxies

can be used for diverse tasks. It can be demonstrated that MeCo and MeCo_{opt} become ineffective on a few tasks, such as Room Layout. We would like to note that the existing proxies do not achieve a high correlation on all tasks consistently.

Table 5: Comparisons of ρ with baselines using Transbench-101-Micro on Ten Tasks

Approach	Autoencoding	Class Objection	Scene Classification	Jigsaw	Surface Normal	Segmantation	Room Layout	Spherical -Cifar100	NinaPro	SVHN
grasp	-0.12	-0.22	-0.43	-0.12	0.01	0.00	-0.29	-0.03	-0.20	-0.24
fisher	**-0.58**	0.44	-0.13	0.30	0.16	0.12	0.30	0.72	0.42	0.81
grad_norm	-0.32	0.39	-0.33	0.36	0.36	0.60	0.25	0.72	0.40	0.78
snip	-0.27	0.45	-0.14	0.41	0.49	0.68	0.32	0.76	0.42	0.83
synflow	0.00	0.48	0.27	0.47	0.00	0.00	0.30	0.79	0.45	**0.92**
l2_norm	0.04	0.32	0.28	0.35	0.50	0.48	0.18	0.53	0.36	0.52
#params	-0.01	0.45	0.32	0.44	0.62	0.68	0.30	0.79	0.36	0.76
zen	0.14	0.54	0.27	0.51	0.71	0.67	0.38	0.67	0.42	0.74
jacov	0.18	0.51	0.19	**0.56**	**0.75**	**0.80**	**0.40**	0.71	0.40	0.77
nwot	0.03	0.39	**0.89**	0.42	0.57	0.53	0.25	0.64	0.38	0.63
zico	0.35	-	0.71	0.52	0.68	-	-	-	-	-
MeCo **(Ours)**	**0.03**	**0.58**	**0.62**	**0.45**	**0.65**	**0.62**	**-0.25**	**0.85**	**0.47**	**0.88**
MeCo_{opt} **(Ours)**	**0.03**	**0.59**	**0.64**	**0.47**	**0.67**	**0.77**	**0.26**	**0.85**	**0.47**	**0.88**

Table 6: Comparisons of ρ with baselines using Transbench-101-Macro on Seven Tasks

Approach	Autoencoding	Class Objection	Scene Classification	Jigsaw	Surface Normal	Segmantation	Room Layout
grasp	-0.02	-0.64	-0.43	-0.26	-0.05	-0.02	-0.26
fisher	-0.19	-0.30	-0.13	-0.26	0.15	0.03	-0.26
grad_norm	0.31	-0.56	-0.33	-0.27	0.35	0.21	-0.27
snip	0.20	-0.38	-0.14	-0.19	0.45	0.27	-0.19
synflow	0.00	0.12	0.27	0.34	0.00	0.00	0.34
l2_norm	-0.20	0.08	0.28	0.15	0.30	0.18	0.15
#params	-0.18	0.16	0.32	0.15	0.30	0.06	0.15
zen	-0.01	0.10	0.27	0.24	0.38	0.27	0.24
jacov	0.45	0.07	0.19	0.19	0.50	0.57	0.19
nwot	**0.67**	**0.83**	**0.89**	**0.48**	0.78	**0.80**	**0.76**
MeCo **(Ours)**	**0.51**	**0.59**	**0.81**	**0.17**	**0.80**	**0.62**	**0.23**
MeCo_{opt} **(Ours)**	**0.74**	**0.73**	**0.76**	**0.48**	**0.76**	**0.63**	**0.33**

E.4 MeCo on AutoFormer and MobileNet OFA

AutoFormer. Chen et al. [21] proposed a novel one-shot architecture search framework for transformer-based models. We load the trained supernets and regenerate the candidate subnets. We then re-evaluate the subnets to obtain the accuracy on ImageNet and compute MeCo. The correlation of MeCo and model accuracy on AutoFormer is 0.45.

OFA. To solve the problem of efficient inference across devices and resource constraints, Cai et al. [22] proposed to train a once-for-all (OFA) network, which supports diverse architectural settings. We randomly sample 1,000 subnets from the OFA network and use the accuracy of the predictor predictions as the test accuracy. The Spearman correlation between MeCo and test accuracy is 0.86.

F More experimental results of NAS with MeCo

In this section, we provide more results and comparisons of our zero-shot NAS on NATS-Bench-TSS and DARTS-CNN. In the following descriptions, we denote multi-shot, one-shot, and zero-shot as MS, OS, and ZS, respectively.

F.1 Results on NATS-Bench-TSS

In the NATS-Bench-TSS search space, we evaluate all networks using zero-cost proxies and choose the Top-10 networks with the highest scores. Then we calculate the average test accuracy and standard deviation. The results are summarized in Table 7. As shown in the table, the networks searched by MeCo have the highest average precision on CIFAR-10, which is 0.08 higher than the best baseline proxy ZiCo. For CIFAR-100 and ImageNet16-120, MeCo achieves competitive results,

e.g., $70.86\% \pm 0.96\%$ with CIFAR-100 and $42.59\% \pm 1.77\%$ with ImageNet16-120. In all, the results demonstrate that our MeCo has a great advantage in evaluating network performance, considering MeCo only requires one data sample as input.

We further combine MeCo with Zero-Cost-PT [18] and search for the best architecture three times with different seeds. The accuracy of the selected architectures as well as the comparisons with the baseline methods are presented in Table 8. It can be shown from the results that our MeCo-based NAS achieves competitive results with the SOTA baselines, e.g., synflow, zen, and ZiCo. However, MeCo-based NAS invokes one data for a single forward pass, thus being more resource-saving.

Table 7: The average test accuracy of Top-10 architectures obtained by various zero-cost proxies on NATS-Bench-TSS using CIAFR-10, CIAFR-100, and ImageNet16-120, respectively

		Baselines								
Dataset	MeCo	grasp[8]	fisher [19]	grad_norm[10]	snip [7]	synflow[9]	jacov[11]	NTK[12]	zen[13]	ZiCo[17]
CIFAR-10	**93.64 ±0.31**	89.34 ±2.16	89.27 ±2.10	89.27 ±2.10	89.27 ±2.10	93.27 ±0.74	91.23 ±1.01	92.67 ±0.46	58.99 ±3.44	93.56 ±0.23
CIFAR-100	70.86 ±0.96	60.89 ±3.88	61.06 ±4.02	60.89 ±3.88	60.89 ±3.88	**71.12 ±1.59**	68.50 ±1.21	69.31 ±1.17	12.73 ±1.33	70.64 ±0.28
ImageNet16-120	42.59 ±1.77	22.99 ±11.01	24.10 ±11.36	23.35 ±11.44	23.35 ±11.44	42.65 ±3.59	40.59 ±1.86	39.98 ±1.73	15.10 ±0.51	**42.74 ±1.78**

Table 8: The test accuracy of optimal architectures obtained by Zero-Cost-PT with various zero-cost proxies on NATS-Bench-TSS using CIAFR-10, CIFAR-100, and ImageNet16-120, respectively

		Baselines								
Dataset	MeCo	grasp[8]	fisher [19]	grad_norm[10]	snip [7]	synflow[9]	jacov[11]	NTK[12]	zen[13]	ZiCo[17]
CIFAR-10	**93.76 ±0**	92.59 ±1.12	88.39 ±2.55	91.64 ±0.68	90.11 ±2.85	**93.76 ±0**	91.92 ±2.40	92.61 ±0.65	**93.76 ±0**	**93.76 ±0**
CIFAR-100	**71.11 ±0**	68.98 ±2.69	65.77 ±0.93	65.20 ±0.56	65.29 ±0.96	**71.11 ±0**	69.67 ±2.39	68.27 ±2.34	**71.11 ±0**	**71.11 ±0**
ImageNet16-120	**41.44 ±0**	35.29 ±8.03	28.91 ±5.18	35.82 ±3.99	37.38 ±4.41	**41.44 ±0**	40.35 ±6.56	41.25 ±2.37	**41.44 ±0**	**41.44 ±0**

F.2 Results on DARTS-CNN

For DARTS-CNN search space, we search the architectures by Zero-Cost-PT with different zero-cost proxies on CIFAR-100. Each searched network is trained five times, and the results are summarized in Table 9. There are two settings in our experiments on DARTS-CNN: networks initialized with 16 channels and trained as in Table 2, and networks initialized with 36 channels and trained as in [23]. It can be shown from Table 9 that MeCo achieves competitive results compared with MS, OS, and ZS baselines. More concretely, MeCo-based NAS obtains 19.33% test error with 0.08 GPU days, which outperforms all the ZS methods under the same settings. On the other hand, compared with the manual, MS, and OS methods, MeCo is also competitive. For example, MeCo-based NAS achieves 83.14% accuracy, which is only 0.34% lower than the baseline method β-DARTS [24], but five times more efficient in computation.

We further visualize the networks searched by Zero-Cost-PT with different proxies on DARTS-CNN. The results on CIFAR-10 and CIFAR-100 are presented in Figure 3 and Figure 4, respectively.

F.3 Results on MobileNet V3

We search the architectures on MobileNet space using Algorithm 1. We retrain the searched network using ImageNet-1K for 480 epochs with a batch size of 256 and input resolution 224×224. The results are summarized in Table 10. MeCo achieves 77.8% Top-1 accuracy, which is 0.7% higher than the SOTA baseline method ZiCo. Moreover, our method only takes 0.04 GPU days, which outperforms the SOTA methods.

G Limitations and future works

Table 9: Comparison of our method with SOTA NAS methods using DARTS-CNN and CIFAR-100.

Approach	Test Error (%)	Search Cost (GPU Days)	Params (MB)	Method
DenseNet-BC[25]	17.18	-	25.6	Manual
NASNet-A[26]	16.82	2000	3.3	MS
DARTS(1st)[27]	17.76	1.5	3.3	OS
SNAS [28]	17.55	1.5	2.8	OS
P-DARTS[24]	15.92 ± 0.18	0.4	3.7	OS
R-DARTS[29]	18.01 ± 0.26	-	-	OS
PC-DARTS[23]	16.9	0.1	3.6	OS
β-DARTS[30]	16.52 ± 0.03	0.4	3.83 ± 0.08	OS
Zero-Cost-PT$^{\dagger}_{\mathsf{synflow}}$ [9]	19.82 ± 0.35	0.04	1.2	ZS
Zero-Cost-PT$^{\dagger}_{\mathsf{fisher}}$[19]	21.14 ± 0.24	0.06	0.7	ZS
Zero-Cost-PT$^{\dagger}_{\mathsf{grasp}}$ [8]	22.65 ± 0.30	0.13	0.7	ZS
Zero-Cost-PT$^{\dagger}_{\mathsf{jacov}}$[11]	22.90 ± 0.35	0.04	0.6	ZS
Zero-Cost-PT$^{\dagger}_{\mathsf{snip}}$[7]	19.95 ± 0.28	0.04	0.8	ZS
Zero-Cost-PT$^{\dagger}_{\mathsf{NTK}}$[12]	20.30 ± 0.33	0.19	0.9	ZS
Zero-Cost-PT$_{\mathsf{ZiCo}^{\dagger}}$[17]	19.54 ± 0.28	0.06	1.1	ZS
Zero-Cost-PT$^{\dagger}_{\mathsf{MeCo}}$	$\mathbf{19.33 \pm 0.25}$	0.08	0.8	ZS
Zero-Cost-PT$^{\ddagger}_{\mathsf{MeCo}}$	$\mathbf{16.86 \pm 0.30}$	0.08	3.7	ZS

†: networks initialized with 16 channels and trained as settings in 2.
‡: networks initialized with 36 channels and trained as settings in [23].

Table 10: Comparison of our method with SOTA NAS methods using MobileNet and ImageNet-1K.

Approach	**Top-1 (%)**	**Search Cost (GPU Days)**	**Params (M)**	**Method**
MobileNet-V3(1.0)	75.2	288	5.3	MS
PNAS	74.2	224	5.1	MS
DARTS	73.3	4	4.7	OS
PC-DARTS	75.8	3.8	-	OS
SPOS	74.7	8.3	-	OS
GreedyNAS	74.9	7.6	3.8	OS
TE-NAS	75.5	0.17	5.4	ZS
ZiCo	77.1	0.4	-	ZS
Zero-Cost-PT	76.4	0.04	8.0	ZS
Zen-score	76.1	0.5	-	ZS
MeCo (Ours)	**77.8**	**0.08**	7.9	ZS

Finally, we demonstrate the limitations of this work and the possible directions of our future work. In Section 5, We propose an optimization method to alleviate the channel-sensitive issue of MeCo. This weight-sampling approach can be further improved in future work. Moreover, we would like to point out that the evaluation of a zero-cost proxy is often tied to the availability of benchmarks. Hench though the proxies are "zero cost", the evaluation of the proxies is strongly dependent on a benchmark, which in the first place is very expensive to create. Finally, although MeCo achieves the highest correlation with the test accuracy on multiple benchmarks, it shows near zero correlations on a few tasks (e.g., TransBench-101-Micro with Autoencoding). We will leave these issues as our future work.

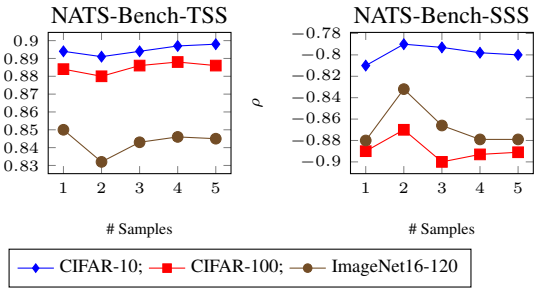


Figure 2: Spearman correlation coefficients of MeCo with different number of samples on NATS-Bench-TSS and NATS-Bench-SSS, respectively.

References

[1] S. S. Du, X. Zhai, B. Poczos, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks," in *International Conference on Learning Representations*, 2019.

[2] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge university press, 2012.

[3] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *The collected works of Wassily Hoeffding*, pp. 409–426, 1994.

[4] Y. Cao and Q. Gu, "Generalization bounds of stochastic gradient descent for wide and deep neural networks," *Advances in neural information processing systems*, vol. 32, 2019.

[5] G. H. Golub and C. F. Van Loan, *Matrix computations*. JHU press, 2013.

[6] Z. Zhu, F. Liu, G. Chrysos, and V. Cevher, "Generalization properties of nas under activation and skip connection search," *Advances in Neural Information Processing Systems*, vol. 35, pp. 23 551–23 565, 2022.

[7] N. Lee, T. Ajanthan, and P. Torr, "Snip: Single-shot network pruning based on connection sensitivity," in *International Conference on Learning Representations*, 2019.

[8] C. Wang, G. Zhang, and R. Grosse, "Picking winning tickets before training by preserving gradient flow," in *International Conference on Learning Representations*, 2020.

[9] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6377–6389, 2020.

[10] M. S. Abdelfattah, A. Mehrotra, Ł. Dudziak, and N. D. Lane, "Zero-cost proxies for lightweight nas," in *International Conference on Learning Representations*, 2021.

[11] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *International Conference on Machine Learning*. PMLR, 2021, pp. 7588–7598.

[12] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on imagenet in four gpu hours: A theoretically inspired perspective," in *International Conference on Learning Representations (ICLR)*, 2021.

[13] M. Lin, P. Wang, Z. Sun, H. Chen, X. Sun, Q. Qian, H. Li, and R. Jin, "Zen-nas: A zero-shot nas for high-performance image recognition," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 347–356.

[14] Y. Shu, S. Cai, Z. Dai, B. C. Ooi, and B. K. H. Low, "NASI: Label- and data-agnostic neural architecture search at initialization," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=v-v1cpNNK_v

[15] J. Xu, L. Zhao, J. Lin, R. Gao, X. Sun, and H. Yang, "Knas: green neural architecture search," in *International Conference on Machine Learning*. PMLR, 2021, pp. 11 613–11 625.

[16] Z. Zhang and Z. Jia, "Gradsign: Model performance inference with theoretical insights," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=HObMhrCeAAF

[17] G. Li, Y. Yang, K. Bhardwaj, and R. Marculescu, "Zico: Zero-shot NAS via inverse coefficient of variation on gradients," in *The Eleventh International Conference on Learning Representations*, 2023.

[18] L. Xiang, Ł. Dudziak, M. S. Abdelfattah, T. Chau, N. D. Lane, and H. Wen, "Zero-cost proxies meet differentiable architecture search," *arXiv preprint arXiv:2106.06799*, 2021.

[19] J. Turner, E. J. Crowley, M. O'Boyle, A. Storkey, and G. Gray, "Blockswap: Fisher-guided block substitution for network compression on a budget," in *International Conference on Learning Representations*, 2020.

[20] Y. Duan, X. Chen, H. Xu, Z. Chen, X. Liang, T. Zhang, and Z. Li, "Transnas-bench-101: Improving transferability and generalizability of cross-task neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5251–5260.

[21] M. Chen, H. Peng, J. Fu, and H. Ling, "Autoformer: Searching transformers for visual recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 12 270–12 280.

[22] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HylxE1HKwS

[23] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient architecture search," in *International Conference on Learning Representations*, 2020.

[24] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 1294–1303.

[25] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

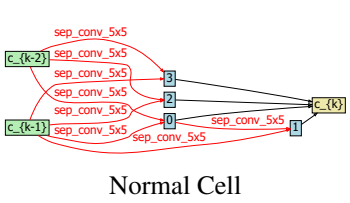
[26] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[27] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations*, 2019.

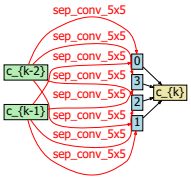
[28] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: stochastic neural architecture search," in *International Conference on Learning Representations*, 2019.

[29] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in *International Conference on Learning Representations*, 2020.

[30] P. Ye, B. Li, Y. Li, T. Chen, J. Fan, and W. Ouyang, "β-darts: Beta-decay regularization for differentiable architecture search," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022, pp. 10 864–10 873.

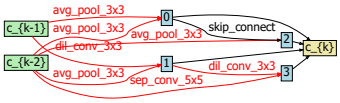


Normal Cell

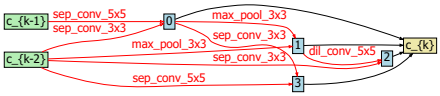


Reduction Cell

(a) synflow

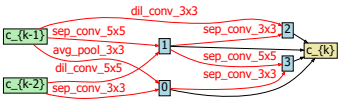


Normal Cell

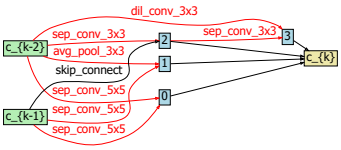


Reduction Cell

(b) fisher

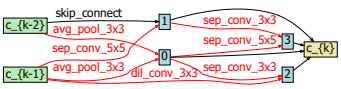


Normal Cell

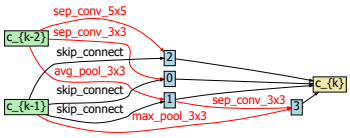


Reduction Cell

(c) snip

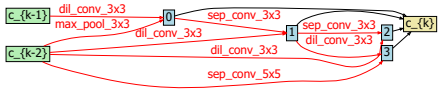


Normal Cell

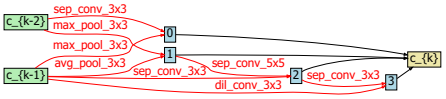


Reduction Cell

(d) grasp

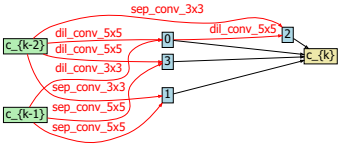


Normal Cell

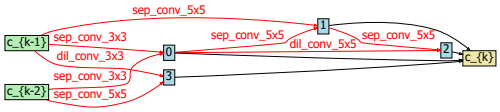


Reduction Cell

(e) jacov

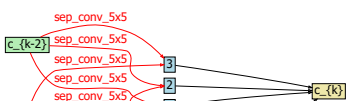


Normal Cell

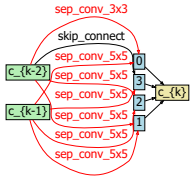


Reduction Cell

(f) NTK



Normal Cell

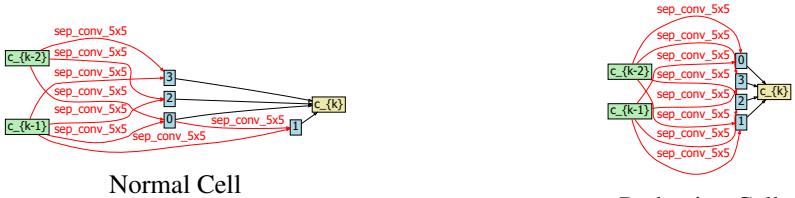


Reduction Cell

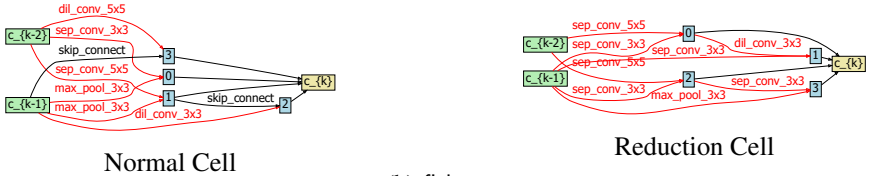
(g) ZiCo

(h) MeCo

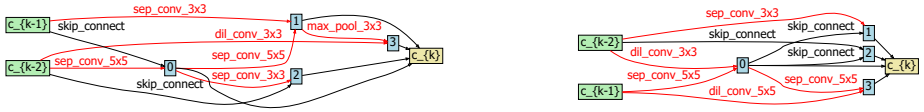
Figure 3: Cells found by Zero-Cost-PT with all zero-cost proxies on the DARTS-CNN search space using CIFAR-10



(a) synflow

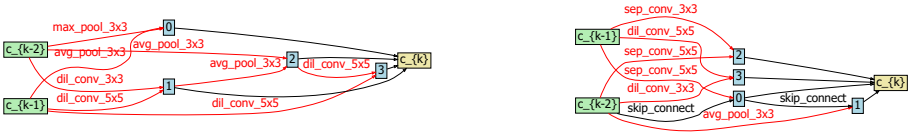


(b) fisher



(c) snip

(d) grasp



(e) jacov

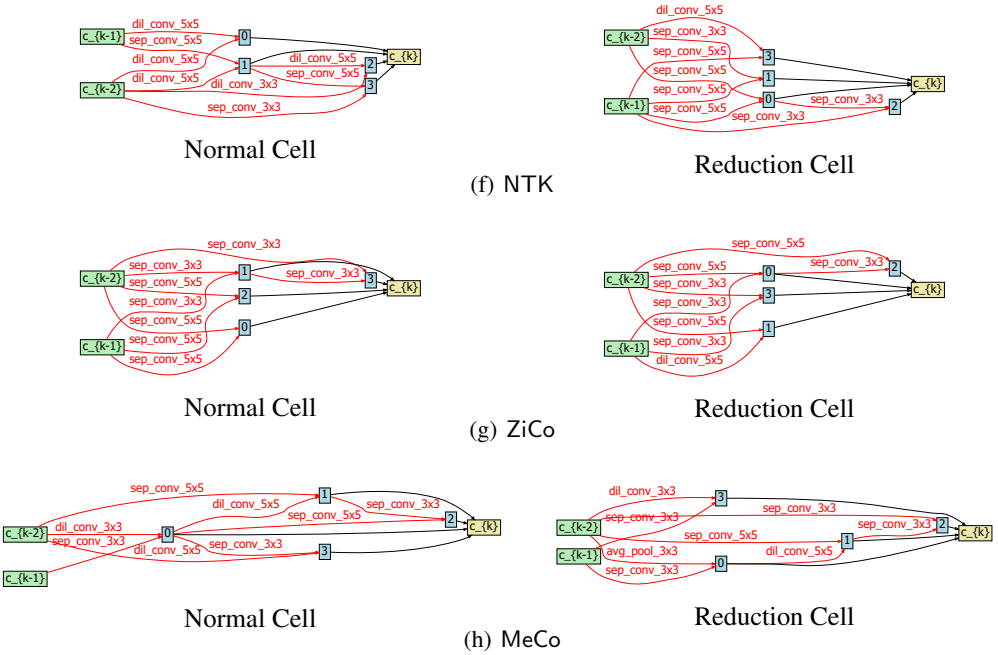


Figure 4: Cells found by Zero-Cost-PT with all zero-cost proxies on the DARTS-CNN search space using CIFAR-100