Posterior-Guided Neural Architecture Search: Supplementary Material

Paper ID 1536

The supplementary materials give the following information, which is not included in the submitted paper due to page limitation.

- Detailed proof
- More details of experimental settings
- A complete version of Table 1
- More results and analysis.

1 Detailed Proof

By introducing the hybrid network representation $\boldsymbol{\varphi}$, Eq. (9) in the paper defines our new objective function which minimize the KL distance between the variational distribution $q_\theta(\boldsymbol{\varphi})$ and the true posterior distribution $p(\boldsymbol{\varphi} \mid \mathcal{D}_t)$

$$\mathcal{D}_{KL}(q_\theta(\boldsymbol{\varphi}), p(\boldsymbol{\varphi} \mid \mathcal{D}_t)), \tag{A1}$$

where $\mathcal{D}_{KL}$ is the KL divergence and $\mathcal{D}_t = \{x_i, y_i\}$ is training dataset. We rewrite the above KL term as

$$\begin{aligned} D_{KL}(q_\theta(\boldsymbol{\varphi}), p(\boldsymbol{\varphi} \mid \mathcal{D}_t)) &= \log(p(\mathcal{D}_t)) \\ &+ \mathbb{E}_{q_\theta(\boldsymbol{\varphi})}[\log(\frac{q_\theta(\boldsymbol{\varphi})}{p(\boldsymbol{\varphi}, \mathcal{D}_t)})]. \end{aligned} \tag{A2}$$

Since $\log(p(\mathcal{D}_t))$ is the constant evidence , minimizing $\mathcal{D}_{KL}(q_\theta(\boldsymbol{\varphi}), p(\boldsymbol{\varphi} \mid \mathcal{D}_t))$ is equivalent to maximizing the evidence lower bound $\mathbb{E}_{q_\theta(\boldsymbol{\varphi})}[\log(\frac{q_\theta(\boldsymbol{\varphi})}{p(\boldsymbol{\varphi}, \mathcal{D}_t)})]$, which leads to

$$\begin{aligned} Minimize\ \mathcal{L}_{VI}(\theta) &:= \mathcal{D}_{KL}(q_\theta(\boldsymbol{\varphi}) \mid\mid p(\boldsymbol{\varphi})) \\ &- \sum_{i=1}^{N} \int q_\theta(\boldsymbol{\varphi}) \log p(y_i \mid f^{\boldsymbol{\varphi}}(x_i)) d\boldsymbol{\varphi}. \end{aligned} \tag{A3}$$

Similar to the section 2.2.2 in (Gal 2016), we assume $q_\theta(\boldsymbol{\varphi})$ to factorize over layer, kernel size and channels as

$$q_\theta(\boldsymbol{\varphi}) = \prod_{s,l,k} q_{\theta^s_{l,k}}(\boldsymbol{\varphi}^s_{l,k}), \tag{A4}$$

where s,l and k denotes l^{th} layer, s^{th} kernel size and l^{th} channel, respectively. As presented by Eq. (11), (12) and (13) in the paper, we re-parametrise the each random variable $\boldsymbol{\varphi}^s_{l,k}$ with

$$\begin{aligned} \boldsymbol{\varphi}^s_{l,k} &= m^s_{l,k} \cdot \boldsymbol{z}^s_{l,k} \cdot \boldsymbol{\alpha}^s_{l,k} = m^s_{l,k} \cdot \boldsymbol{\epsilon}^s_{l,k}, \\ &where\ \boldsymbol{\epsilon}^s_{l,k} \sim Bernoulli(p^s_l). \end{aligned} \tag{A5}$$

If we replace the $\boldsymbol{\varphi}$ with its re-parametrisation form, the function draw over the $q_\theta(\boldsymbol{\varphi})$ in Eq. A2 is equivalent to the function draw over $p(\boldsymbol{\epsilon})$, which leads to

$$\begin{aligned} &\sum_{i=1}^{N} \int q_\theta(\boldsymbol{\varphi}) \log p(y_i \mid f^{\boldsymbol{\varphi}}(x_i)) d\boldsymbol{\varphi} \\ =&\sum_{i=1}^{N} \int p(\boldsymbol{\epsilon}_i) \log p(y_i \mid f^{\theta, \boldsymbol{\epsilon}_i}(x_i)) d\boldsymbol{\epsilon} \\ \approx&\sum_{i=1}^{N} \log p(y_i \mid f^{\theta, \boldsymbol{\epsilon}_i}(x_i)). \end{aligned} \tag{A6}$$

We use Monte Carlo estimation to approximate the integral term in the above equation. Now, our objective function is to minimize

$$\mathcal{D}_{KL}(q_\theta(\boldsymbol{\varphi}) \mid\mid p(\boldsymbol{\varphi})) - \sum_{i=1}^{N} \log p(y_i \mid f^{\theta, \boldsymbol{\epsilon}_i}(x_i)). \tag{A7}$$

Since $\boldsymbol{\varphi}$ is reparameterised with kernel weight $m^s_{l,k}$ multiplying random variable $\boldsymbol{\epsilon}^s_{l,k}$ that subjects to a Bernoulli distribution, the sampling over $\boldsymbol{\varphi}$ is equivalent to sampling over input channel (because of our channel-level factorization), which corresponds to the native Dropout operation. Therefore, the second term in Eq. A7 is equivalent to the inference of a dropout neural network with dataset $\{x_i, y_i\}$. However, the derivative w.r.t. the Eq. A7 is difficult to compute because of the KL term.

Here we leverage the Proposition 4 in (Gal 2016) which proves that given fixed $M, N \in \mathbb{N}$, a probability vector $\mathbf{p} = (p_1, p_2, ..., p_N)$, and $\Sigma_i \in \mathbb{R}^{M \times M}$ diagonal postitive-definite for $i = 1, 2, ..., N$, with the elements of each σ_i not dependent on M, and let $q(x) = \sum_{i=1}^{N} p_i \mathcal{N}(x; \mu_i, \Sigma_i)$ be a mixture of Gaussians with N components, where $\mu_i \in \mathbb{R}^M$,

if assuming that $\mu_i - \mu_j \sim \mathcal{N}(0, I)$, the KL divergence between $q(x)$ and $p(x) = \mathcal{N}(0, I_k)$ can be approximated as

$$\begin{aligned}\mathcal{D}_{KL}(q(x), p(x)) \approx \sum_{i=1}^{N} [\frac{p_i}{2}(\mu_i^t \mu_i + tr(\Sigma_i) \\ - K(1 + \log 2\pi) - \log|\Sigma_i|) + p_i \log p_i].\end{aligned} \tag{A8}$$

According to Eq. A5, $q_{\theta^s_{l,k}}(\boldsymbol{\varphi}^s_{l,k} \mid \boldsymbol{\epsilon}^s_{l,k}) = \delta(\boldsymbol{\varphi}^s_{l,k} - m^s_{l,k} \cdot \boldsymbol{\epsilon}^s_{l,k})$. We can approximate each $q_{\theta^s_{l,k}}(\boldsymbol{\varphi}^s_{l,k} \mid \boldsymbol{\epsilon}^s_{l,k})$ as a narrow Gaussian with a small standard deviation $\Sigma = \sigma^2 I$. Given that $q_\theta(\boldsymbol{\varphi}) = \int q_\theta(\boldsymbol{\varphi} \mid \boldsymbol{\epsilon}) p(\boldsymbol{\epsilon}) d\boldsymbol{\epsilon}$, $q_\theta(\boldsymbol{\varphi}^s_{l,k})$ is actually a mixture of two Gaussians with small standard deviations (similar with the one in the above proposition), where one component fixed at zero and another one fixed at $m^s_{l,k}$.

Since we set the prior u to be zero in the paper, which leads the priori distribution $p(\boldsymbol{\varphi})$ to be exactly $\mathcal{N}(\boldsymbol{w}^s_{l,k}; 0, I/(d^s_{l,k})^2)$. Given the Eq. A4 and the above proposition, it can be easily derived that

$$\begin{aligned}&\frac{\partial}{\partial m^s_{l,k}} \mathcal{D}_{KL}(q_\theta(\boldsymbol{\varphi}) \mid\mid p(\boldsymbol{\varphi})) \\ =&\frac{\partial}{\partial m^s_{l,k}} \sum_{s,l,k} \mathcal{D}_{KL}(q_\theta(\boldsymbol{\varphi}^s_{l.k}) \mid\mid p(\boldsymbol{\varphi}^s_{l,k})) \\ \approx&\frac{\partial}{\partial m^s_{l,k}} \sum_{s,l,k} \frac{(1 - p^s_l) d^s_{l,k}}{2} \|m^s_{l,k}\|^2.\end{aligned} \tag{A9}$$

Besides the variational parameters $m^s_{l,k}$, the optimal distribution of random variable $\boldsymbol{\epsilon}$ which encodes the network architecture information also needs to be found. In order to facilitate a gradient based solution, we employ Gumbel-softmax to relax the discrete Bernoulli distribution to continuous space. More specifically, instead of drawing $\boldsymbol{\epsilon}^s_{l,k}$ w.r.t. the $Bernoulli(p^s_l)$, we deterministically draw the $\boldsymbol{\epsilon}^s_{l,k}$ with

$$\begin{aligned}\boldsymbol{\epsilon}^s_{l,k} =&\sigma(\frac{1}{\tau}[\log p^s_l - \log(1 - p^s_l) \\ &+ \log(\log r_2) - \log(\log r_1)]) \\ &s.t.\ r_1, r_2 \sim Uniform(0, 1).\end{aligned} \tag{A10}$$

Under this re-parametrisation, the distribution of $boldsymbol\epsilon^s_{l,k}$ is smooth for $\tau > 0$ and $p(\boldsymbol{\epsilon}^s_{l,k}) \rightarrow Bernoulli(p^s_l)$ as τ approaches 0 and. Therefore, we have well-defined gradients w.r.t. the probability p^s_l by using a small τ. Combining Eq. A9 and A10, we can obtain the gradients presented by the Eq. (15) and the Algorithm 1 in the paper.

1.1 Cifar-10 and Cifar-100

One-shot Model We test our NASAS with four super networks, SupNet-M/MI/E/EI, respectively, on Cifar-10 and Cifar-100.

The SupNet-M/MI are build on the 3-branch ResNet (Gastaldi 2017). We first choose the multi-branch ResNet as a compact super network and denote it as SupNet-M. The SupNet-M contains three stages. Each stage has four residual blocks. Each residual block consists of two parallel residual paths with a structure unit ReLU-Conv3x3-BN-ReLU-Conv3x3-BN and one skip connection. When downsampling is required, the skip connection has two concatenated branches. Each branch has the structure unit Pool1x1-Conv1x1. The SupNet-M has 25 layers in total and the first layer is a 3x3 convolutional layer. The SupNet-MI is conducted by inflating the SupNet-M via simply doubling the number of channels in each layer.

The SupNet-E/EI are bulid on the architecture designed by ENAS (Pham et al. 2018). The SupNet-E follows the design in ENAS and SupNet-EI is the inflated version by double the channels of SupNet-E. Both of them involve 1×1, 3×3, 5×5, and 7×7 convolutions/separable convolutions.

Hyper-parameters For fair comparison, we follow the setting of parameters in (DeVries and Taylor 2017)(Pham et al. 2018) to train our SupNet-M/MI and SupNet-E/EI.

More specifically, the SupNet-M/MI are trained with the initial learning rate 0.01, 1800 epochs and a Consine decay without restart (Loshchilov and Hutter 2016). The batch size is set to be 128. We train the SupNet-E/EI for 630 epochs using an Cosine learning rate decay with warm restarts. The max learning rate and min learning rate are set to be 0.05 and 0.0001, respectively. Initial period T_0 and period multiplier T_{mul} are set to be 10 and 2, respectively. We also adopt the cutout (DeVries and Taylor 2017) for data augmentation. The cutout size is set to be 16.

There is no weight decay ratio in our scheme as it is replaced by the l^2 term in the Eq. 16 in the paper. On Cifar-10, we use $l^2 = 0.1$ for SupNet-E/EI, $l^2 = 50$ for SupNet-M and $l^2 = 150$ for SupNet-MI, respectively. On Cifar-100, we use $l^2 = 2500$ for SupNet-MI. Temperature τ is set to be 0.2 and initial drop out rate is set to be 0.1. We set N in Eq.16 to be 5e5, which is around 10 times as the real number of training samples, due to data augmentation (e.g. cutout).

1.2 ImageNet

One-shot Model We test our NASAS with two super networks,SupNet-R-50 and SupNet-D-121, on ImageNet.

The SupNet-R-50 is built upon the ResNet50 (He et al. 2016). It is generated by inflating the ResNet50 by replacing every 3x3 convolution in each layer with four convolutions of size 1x1, 3x3, 5x5 and 9x9, respectively. The output of the four convolutions are then concatenated to form a multi-branch and multi-kernel one-shot model. For each convolution, we divide the number of feature maps by four to have the same number of feature maps as that in the ResNet50. Also, since the 5x5 and 9x9 convolutions consume too many parameters, we use dilated convolution with kernel size 3x3 and and 5x5 with dilation rate 1 to mimic the 5x5 and 9x9 convolutions, respectively. Other components, such as number of residual block, layers and connection patterns, are identical to those in the ResNet50.

We choose the DenseNet121 (Huang et al. 2017) as a compact super network. We denote the super network as SupNet-D-121. It consists of multi-branched and densely connected convolutions with filter sizes 1×1 and 3×3. SupNet-D-121 is composed of one 7x7 convolutional layer

Method	Error(%)	GPUs Days	Params(M)	Search Method
DenseNet-BC (Huang et al. 2017)	3.46	-	25.6	-
PyramidNet (Han, Kim, and Kim 2017)	3.31	-	26.0	-
shake-shake (Gastaldi 2017)	2.86	-	26.2	-
shake-shake + cutout (DeVries and Taylor 2017)	2.56	-	26.2	-
NAS (Zoph and Le 2016)	4.47	22400	7.1	RL
NAS + more filters (Zoph and Le 2016)	3.65	22400	37.4	RL
NASNET-A (Zoph et al. 2018)	3.41	1800	3.3	RL
NASNET-A + cutout (Zoph et al. 2018)	2.65	1800	3.3	RL
Macro NAS + Q-Learning (Baker et al. 2016)	6.92	100	11.2	RL
Micro NAS + Q-Learning (Zhong et al. 2018)	3.60	96	-	RL
PathLevel EAS + PyramidNet + cutout (Cai et al. 2018b)	2.30	8.3	13.0	RL
Proxyless-R + cutout (Cai, Zhu, and Han 2018)	2.30	4	5.8	RL
ENAS + cutout (Pham et al. 2018)	2.89	0.5	4.6	RL
EAS (DenseNet) (Cai et al. 2018a)	3.44	10	10.7	RL
AmoebaNet-A + cutout (Real et al. 2018)	3.34	3150	3.2	evolution
AmoebaNet-B + cutout (Real et al. 2018)	2.55	3150	2.8	evolution
Hierachical Evo (Liu et al. 2017)	3.63	300	61.3	evolution
PNAS (Liu et al. 2018)	3.63	225	3.2	SMBO
Proxyless-G + cutout (Cai, Zhu, and Han 2018)	2.08	4	5.7	gradient-based
BayesNAS (Zhou et al. 2019) + cutout	2.81	0.2	3.4	gradient-based
BayesNAS + PyramidNet(Zhou et al. 2019) + cutout	2.40	0.1	3.4	gradient-based
DARTS + cutout (Liu, Simonyan, and Yang 2018)	2.83	4	3.4	gradient-based
SNAS + cutout (Xie et al. 2018)	2.85	1.5	2.8	gradient-based
NAONet + cutout (Luo et al. 2018)	2.07	200	128	gradient-based
One-Shot Top (Bender et al. 2018)	3.70	-	45.3	sampling-based
SMASH (Brock et al. 2017)	4.03	1.5	16.0	sampling-based
PGNAS-MI(ours)	2.06	6.5	33.4	guided sampling
PGNAS-MI*(ours)	**1.98**	11.1	32.8	guided sampling

Table 1: Performance comparison with other state-of-the-art results (full table). Please note that we do not fine-tune the network searched by our method. * indicates the architecture searched by sampling 10000 candidates.

Method	Pre-training	Extra Module	frames	Top1 Accuracy (%)
ECO	Kinetics	None	16	41.4
ECO$_{light}$	Kinetics	None	32	41.3
I3D	ImageNet+Kinetics	None	32x2	41.6
Non-local I3D	ImageNet+Kinetics	Non-local block	32x2	44.4
I3D + Joint GCN	ImageNet+Kinetics	GCN	32x2	43.3
NASAS	ImageNet	None	32	44.2

Table 2: Performance comparison with other state-of-art results on action SomethingSomething. The 3D architecture searched by NASAS performs on par with the most advanced 3D network-based methods that employ extra modules and use more frames. Please note that we still do not fine-tune the network searched by our method.

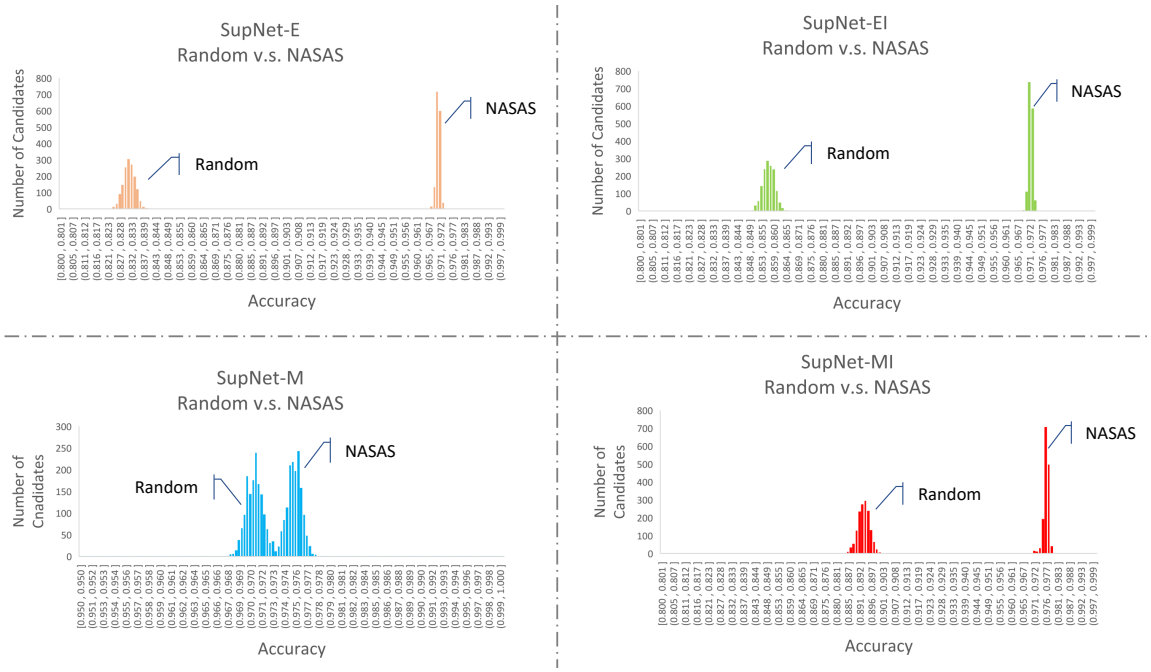

Figure 1: Histograms of performance of sampled candidates. In each sub-figure, we give two histograms of random sampling and our NASAS, respectively. Here "Random" sampling is implemented by employing predefined dropout strategy (Bender et al. 2018) as discussed in the subsection 3.1: ablation study in the submitted paper.

followed by four dense blocks and three transition layers. Each dense block has N serialized convolution blocks and the structure of a single convolution block follows BN-ReLU-Conv1x1-BN-ReLU-Conv3x3. The outputs of each convolution block are fed to all following convolution blocks, where different inputs are concatenated along channel dimension. The N for four dense blocks are 6, 12 ,24 and 16, respectively. Each transition layer has BN-ReLU-Conv1x1-Pool2x2 structure which are inserted between two dense blocks to facilitate the down-sampling operation.

Hyper-parameters For fair evaluation, we follow most of the parameter settings in (He et al. 2016)(Huang et al. 2017) to train the SupNet-D-121 and SupNet-R-50, respectively.

We train the SupNet-R-50 for 90 epochs with an initial learning rate 0.1 and batch size 256. We decay the learning rate 10 times every 30 epochs. We train the SupNet-D-121 for 180 epochs since we find the dropout ratio is not converged when only trained for 90 epochs. We decay the learning rate 10 times at epoch 90, 140 and 160 respectively. The learning rate and batch size are set to be 0.1 and 300. Both networks are trained with Stochastic Gradient Descent (SGD) with a Nesteriv momentum of 0.9.

There is also no weight decay ratio. We use $l^2 = 80$ for SupNet-R-50 and $l^2 = 100$ SupNet-D-121, respectively. Temperature τ is set to be 0.2 and initial drop out rate is set to be 0.1. N in Eq.16 is 1.2e6, which is the number of training samples with standard data augmentation.

2 More Results

2.1 Cifar-10

We provide the full version of Table. 1 here. More related works are enlisted. It further demonstrates that our approach can achieve the most advanced performance with a few GPU days among all the previous methods.

2.2 Performance of the Sampled Candidates

We show the histograms of the performance of sampled candidates in Fig. 1. In each sub-figure, we give two histograms of random sampling and our NASAS, respectively. Here baseline "Random" sampling is implemented by employing predefined dropout strategy (Bender et al. 2018) as discussed in the subsection 3.1: ablation study (Table 3 (a)) in the submitted paper. We observe that the deviation/mean of histograms of our NASAS is much smaller/higher than those of random search for all the one-shot models. It indicates that a posteriori distribution estimated in the NASAS puts more mass on those advanced architectures compared to the predefined distribution, which satisfies our optimization objectives and leading to advanced results.

One might observe that the gap between NASAS and baseline method under SupNet-M is small. It is because the best architecture found by NASAS has a similar parameter size with the one-shot model, and we have to use a very small dropout rate 0.1 in the baseline method to ensure the comparable parameter size. Therefore, the candidate sub-networks sampled by the baseline method are very similar to the one-shot model itself, which leads to the performance of

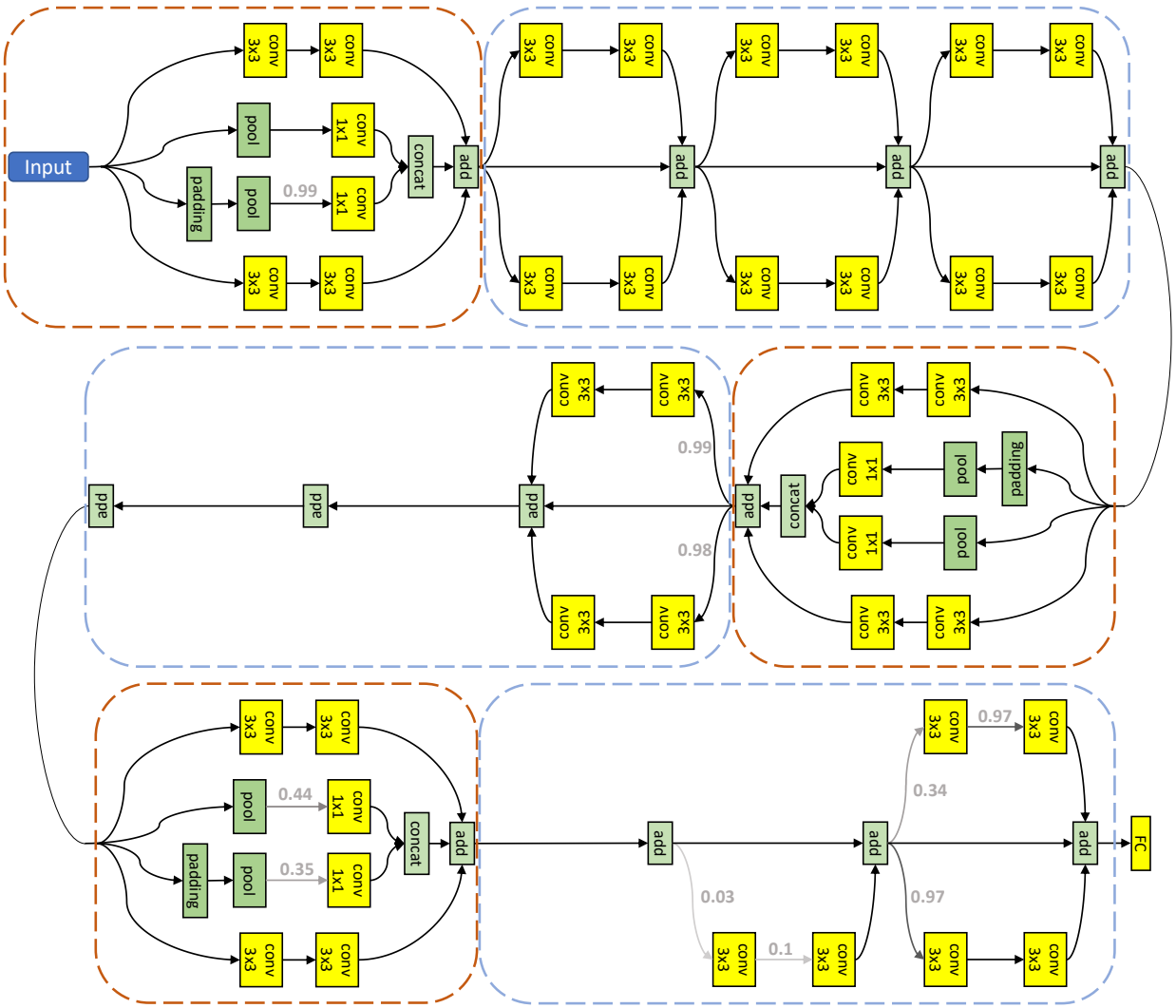

Figure 2: Visualization of the searched architecture for NASAS-MI. Grey number before an operation denotes the proportion of kernels that are kept. An operation without a proportion number indicates all channels are selected. We do not draw the operation whose kernels are not selected at all.

different sub-networks is closed to the performance of one-shot model.

2.3 Visualization

We provide the visualization for the searched architecture of the best-performed NASAS-MI. As can be viewed in Figure. 2, although the initial one-shot model is a multi-branched ResNet whose basic blocks are originally identical, NASAS can still find very diverse architecture for each basic block. We also find that NASAS tends to choose more operations for those layers which increase the number of channels (as denoted by the red dashed bounding box), but to drop more operations for those layers with identical input/output channel numbers (as denoted by the blue dashed bounding box). We believe it is because that increasing channel number is to embed signals into higher dimensional space, which is crucial for classifying data that are not separable in low dimensional space. However, there is only a small part of layers in the original one-shot model increasing the channel number, so the best architecture searched by NASAS try to incorporate all of them.

Actually, this behavior is consistent with the previous analysis in (Veit, Wilber, and Belongie 2016). They find that units where the feature map dimension is doubled (usually with downsampling) have higher impacts on network performance compared with other units. Similar hint has been leveraged by human experts to improve the existing network performance. For example, in (Han, Kim, and Kim 2017), they propose to gradually increases the number of channels as a function of the depth at which the layer occurs rather than increasing the channels only on a couple of pre-selected layers.

2.4 Action Recognition

We apply the proposed NASAS to a more complex task that involves 3D signals, i.e. spatio-temporal feature learning for human action recognition. More concretely, we follow (Carreira and Zisserman 2017) to inflate the 2D version DenseNet121 to a 3D-DensNet121. Instead of employing true 3D convolution, we use separable 3D convolution which consists of a 3x1x1 temporal convolution followed by a 1x3x3 spatial convolution to reduce computation cost. We use NASAS to search an efficient spatio-temporal architecture based on the inflated 3D-DenseNet121 on Something-Something v1 dataset (Goyal et al. 2017) . Result in Table. 2 demonstrates that the performance of the searched architecture is even comparable to the state-of-the-art 3D CNN based approaches that incorporate additional specifically designed modules and use a very large video datasets i.e., Kinetics (Kay et al. 2017), for pre-training.

References

Baker, B.; Gupta, O.; Naik, N.; and Raskar, R. 2016. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*.

Bender, G.; Kindermans, P.-J.; Zoph, B.; Vasudevan, V.; and Le, Q. 2018. Understanding and simplifying one-shot architecture search. In *International Conference on Machine Learning*, 549–558.

Brock, A.; Lim, T.; Ritchie, J. M.; and Weston, N. 2017. Smash: one-shot model architecture search through hypernetworks. *arXiv preprint arXiv:1708.05344*.

Cai, H.; Chen, T.; Zhang, W.; Yu, Y.; and Wang, J. 2018a. Efficient architecture search by network transformation. AAAI.

Cai, H.; Yang, J.; Zhang, W.; Han, S.; and Yu, Y. 2018b. Path-level network transformation for efficient architecture search. *arXiv preprint arXiv:1806.02639*.

Cai, H.; Zhu, L.; and Han, S. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*.

Carreira, J., and Zisserman, A. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6299–6308.

DeVries, T., and Taylor, G. W. 2017. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*.

Gal, Y. 2016. *Uncertainty in deep learning*. Ph.D. Dissertation, PhD thesis, University of Cambridge.

Gastaldi, X. 2017. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*.

Goyal, R.; Kahou, S. E.; Michalski, V.; Materzynska, J.; Westphal, S.; Kim, H.; Haenel, V.; Fruend, I.; Yianilos, P.; Mueller-Freitag, M.; et al. 2017. The" something something" video database for learning and evaluating visual common sense. In *ICCV*, volume 1, 3.

Han, D.; Kim, J.; and Kim, J. 2017. Deep pyramidal residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5927–5935.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

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

Kay, W.; Carreira, J.; Simonyan, K.; Zhang, B.; Hillier, C.; Vijayanarasimhan, S.; Viola, F.; Green, T.; Back, T.; Natsev, P.; et al. 2017. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*.

Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; and Kavukcuoglu, K. 2017. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436*.

Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.-J.; Fei-Fei, L.; Yuille, A.; Huang, J.; and Murphy, K. 2018. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 19–34.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*.

Loshchilov, I., and Hutter, F. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Luo, R.; Tian, F.; Qin, T.; Chen, E.; and Liu, T.-Y. 2018. Neural architecture optimization. In *Advances in Neural Information Processing Systems*, 7826–7837.

Pham, H.; Guan, M. Y.; Zoph, B.; Le, Q. V.; and Dean, J. 2018. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2018. Regularized evolution for image classifier architecture search. *arXiv preprint arXiv:1802.01548*.

Veit, A.; Wilber, M. J.; and Belongie, S. 2016. Residual networks behave like ensembles of relatively shallow networks. In *Advances in neural information processing systems*, 550–558.

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2018. Snas: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926*.

Zhong, Z.; Yan, J.; Wu, W.; Shao, J.; and Liu, C.-L. 2018. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2423–2432.

Zhou, H.; Yang, M.; Wang, J.; and Pan, W. 2019. Bayesnas: A bayesian approach for neural architecture search. *arXiv preprint arXiv:1905.04919*.

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

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