Supplementary Materials

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

- Detailed proofs on Section 3.2 and 3.3 in the paper
- More details on experimental setup.
- More results on the Section 4.5 (Generalization).
- Discussions on Neural Architecture Search (NAS).

1. Detailed Proof

1.1. Equation 5 in the paper

By incorporating the template network, the probability measure function in Eq. (1) in the paper can be converted to

$$\mathcal{P}(\mathcal{M}, W_{\mathcal{M}} \mid \mathbb{D}) \rightarrow \mathcal{P}(\widehat{\mathcal{M}} \circ W_T \mid \mathbb{D}), \tag{S1}$$

where $\circ$ is the Hadamard product[1], $\widehat{\mathcal{M}} \in (0,1)^{L\times L\times 3}$ is a binary random matrix and $\widehat{\mathcal{M}}(l,i,u) = 1/0$ denotes that the feature from the layer i and the fusion unit u is enabled/disabled at layer l in the template network, respectively. $W_T \in \mathbb{R}^{L\times L\times 3\times V}$ denotes the random weight matrix of the template network, where we use V to denote kernel shape for simplicity. This conversion actually integrates the kernel weights into fusion strategies. Since we can fully recover the $\mathcal{M}$ from the embedded version $\widehat{\mathcal{M}} \circ W_T$ (it is because the kernel is defined in real number field, the probability of being zero for every element can be ignored), the first requirement is still satisfied.

We then approximate the posteriori distribution by minimizing the KL divergence

$$KL(\mathcal{Q}(\widehat{\mathcal{M}} \circ W_T) \mid\mid \mathcal{P}(\widehat{\mathcal{M}} \circ W_T \mid \mathbb{D}), \tag{S2}$$

where $\mathcal{Q}(\cdot)$ denotes a variational distribution. We can rewrite the above KL term as

$$\log(p(\mathbb{D})) + \mathbb{E}_{\mathcal{Q}(\widehat{\mathcal{M}}\circ W_T)}[\log(\frac{\mathcal{Q}(\widehat{\mathcal{M}} \circ W_T)}{\mathcal{P}(\widehat{\mathcal{M}} \circ W_T, \mathbb{D})})]. \tag{S3}$$

[1]We actually broadcast (repeat) the elements in the last dimension of $\widehat{\mathcal{M}}$ to match the shape $L \times L \times 3 \times V$

Since $\log(p(\mathbb{D}))$ is the constant evidence , minimizing Eq. S2 is equivalent to minimizing the negative evidence lower bound $\mathbb{E}_{\mathcal{Q}(\widehat{\mathcal{M}}\circ W_T)}[\log(\frac{\mathcal{Q}(\widehat{\mathcal{M}}\circ W_T)}{\mathcal{P}(\widehat{\mathcal{M}}\circ W_T,\mathbb{D})})]$. This lower bound can be further rewritten as

$$\begin{aligned} &KL(\mathcal{Q}(\widehat{\mathcal{M}}\circ W_T) \mid\mid \mathcal{P}(\widehat{\mathcal{M}}\circ W_T)) \\ &-\sum_{t=1}^{N}\int \mathcal{Q}(\widehat{\mathcal{M}}\circ W_T)\log p(y_t \mid x_t, \widehat{\mathcal{M}}\circ W_T)\mathrm{d}\widehat{\mathcal{M}}\circ W_T. \end{aligned} \tag{S4}$$

We assume the $\mathcal{Q}(\widehat{\mathcal{M}} \circ W_T)$ to have a factorized form and we factorize it over fusion units at each layer as

$$\prod_{l,i,u} q(\widehat{\mathcal{M}}(l,i,u) \cdot W_T(l,i,u,:)). \tag{S5}$$

We further re-parameterize the random kernel weight W_T with a deterministic kernel weight multiplying a random variable that subjects to some distribution. Take a univariate Gaussian distribution $x \sim q_\theta(x) = \mathcal{N}(\mu,\sigma)$ as an example, its re-parametrization can be $x = g(\theta,\boldsymbol{\epsilon}) = \mu + \sigma\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(0,1)$ a parameter-free random variable, where μ and σ are the variational parameters θ. Following [3], we choose the Bernoulli distribution for the re-parametrization, which leads to

$$\begin{aligned} \widehat{\mathcal{M}}(l,i,u) \cdot W_T(l,i,u,:) = \widehat{\mathcal{M}}(l,i,u) \cdot (w_{l,i,u} \cdot \boldsymbol{z}_{l,i,u}), \\ where\ \boldsymbol{z}_{l,i,u} \sim Bernoulli(\widetilde{p}_{l,i,u}). \end{aligned} \tag{S6}$$

Here $w_{l,i,u}$ is the deterministic weight matrix associated with the random weight matrix $W_T(l,i,u,:)$. Since each $\widehat{\mathcal{M}}(l,i,u)$ controls the utilization of the fusion units u with binary values 1 and 0, it also subjects to Bernoulli distribution, denoted as $Bernoulli(\widehat{p}_{l,i,u})$. Therefore, we use a new Bernoulli distribution to replace the original two as

$$\begin{aligned} \widehat{\mathcal{M}}(l,i,u) \cdot W_T(l,i,u,:) = w_{l,i,u} \cdot \boldsymbol{\epsilon}_{l,i,u}, \\ where\ \ \boldsymbol{\epsilon}_{l,i,u} \sim Bernoulli(p_{l,i,u}), \\ p_{l,i,u} = \widehat{p}_{l,i,u} \cdot \widetilde{p}_{l,i,u}. \end{aligned} \tag{S7}$$

Now, we replace the $\widehat{\mathcal{M}}(l,i,u) \cdot W_T(l,i,u,:)$ with its re-parameterized form in the second term in Eq. S4, which

leads to

$$\begin{aligned}&\sum_{t=1}^{N}\int \mathcal{Q}(\widehat{\mathcal{M}}\circ W_T)\log p(y_t \mid x_t,\widehat{\mathcal{M}}\circ W_T)\mathrm{d}\widehat{\mathcal{M}}\circ W_T\\ =&\sum_{t=1}^{N}\int p(\epsilon)\log p(y_t \mid x_t, w\cdot\epsilon)\mathrm{d}\epsilon\\ \approx&\sum_{t=1}^{N} p(\epsilon^t)\log p(y_t \mid x_t, w\cdot\epsilon^t).\end{aligned} \tag{S8}$$

We use Monte Carlo estimation to approximate the integral term in the above equation, where ϵ^t indicates t-th sampling. Combining Eq. S4 with Eq. S8, our objective function is converted to minimizing

$$\begin{aligned}&KL(\mathcal{Q}(\widehat{\mathcal{M}}\circ W_T)\;||\;\mathcal{P}(\widehat{\mathcal{M}}\circ W_T))\\ -&\sum_{t=1}^{N} p(\epsilon^t)\log p(y_t \mid x_t, w\cdot\epsilon^t).\end{aligned} \tag{S9}$$

The second term in Eq. S9 is equivalent to the inference of a neural network with DropPath on dataset $\{x_i, y_i\}$. However, the derivatives w.r.t. the Eq. S9 is still difficult to compute because of the KL term.

Here we leverage the Proposition 4 in [2] which proves that given fixed $M, C \in \mathbb{N}$, a probability vector $\mathbf{p} = (p_1, p_2, ..., p_C)$, and $\Sigma_h \in \mathbb{R}^{M\times M}$ diagonal positive-definite for $h = 1, 2, ..., C$, with the elements of each Σ_h not dependent on M, and let $q(x) = \sum_{h=1}^{C} p_h\mathcal{N}(x; \mu_h, \Sigma_h)$ be a mixture of Gaussians with N components, where $\mu_h \in \mathbb{R}^M$, if assuming that $\mu_h - \mu_j \sim \mathcal{N}(0, I)$, the KL divergence between $q(x)$ and $p(x) = \mathcal{N}(0, I_k)$ can be approximated as

$$\begin{aligned}KL(q(x), p(x)) \approx \sum_{h=1}^{C}[&\frac{p_h}{2}(\mu_h^t\mu_h + tr(\Sigma_h)\\ &- K(1+\log 2\pi) - \log|\Sigma_h|) + p_h\log p_h].\end{aligned} \tag{S10}$$

Actually, Eq. S7 suggests that

$$\begin{aligned}&q(\widehat{\mathcal{M}}(l,i,u)\cdot W_T(l,i,u,:))\\ =&\delta(\widehat{\mathcal{M}}(l,i,u)\cdot W_T(l,i,u,:) - w_{l,i,u}\cdot\boldsymbol{\epsilon}_{l,i,u}),\end{aligned} \tag{S11}$$

and we can approximate each $q(\widehat{\mathcal{M}}_{(l,i,u)}\cdot W_T(l,i,u,:)|\epsilon_{l,i,u})$ as a narrow Gaussian with a small standard deviation $\Sigma = \sigma^2 I$. Therefore, $q(\widehat{\mathcal{M}}_{(l,i,u)}\cdot W_T(l,i,u,:)) = \int q(\widehat{\mathcal{M}}_{(l,i,u)}\cdot W_T(l,i,u,:)|\epsilon_{l,i,u})p(\boldsymbol{\epsilon})d\boldsymbol{\epsilon}$ is also 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 $w_{l,i,u}$. If we assume the prior of u to be 'S+ST' at all layers in the template network and the prior of kernel weight to be Guassian distribution $\mathcal{N}(w_{l,i,u}; 0, I/(k_{l,i,u})^2)$, where $k_{l,i,u}$ is a prior length scale, the prior distribution of each $\widehat{M}(l,i,u)\cdot W_T(l,i,u,:)$ is still Guassian. Given the Eq. S5 and the proposition Eq. S10, it can be easily derived that

$$\begin{aligned}&\frac{\partial}{\partial w\partial p}KL(\mathcal{Q}(\widehat{\mathcal{M}}\circ W_T)\;||\;\mathcal{P}(\widehat{\mathcal{M}}\circ W_T))\\ =&\frac{\partial}{\partial w\partial p}\sum_{l,i,u}KL(q(\widehat{\mathcal{M}}_{(l,i,u)}\cdot W_T(l,i,u,:))\;||\;p(\widehat{\mathcal{M}}_{(l,i,u)}\cdot W_T(l,i,u,:)))\\ \approx&\frac{\partial}{\partial w\partial p}\sum_{l,i,u}\frac{(1-p_{l,i,u})k_{l,i,u}^2}{2}\|w_{l,i,u}\|^2 + p_{l,i,u}\log p_{l,i,u}.\end{aligned} \tag{S12}$$

In addition to the variational parameters $w_{l,i,u}$, the optimal distribution of random variable ϵ 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 $\epsilon_{l,i,u}$ w.r.t. the $Bernoulli(p_{l,i,u})$, we deterministically draw the $\epsilon_{l,i,u}$ with

$$\begin{aligned}\epsilon_{l,i,u} = Sigmoid(\frac{1}{\tau}[&\log p_{l,i,u} - \log(1-p_{l,i,u})\\ &+\log(\log r_2) - \log(\log r_1)])\\ s.t.\; r_1, r_2 &\sim Uniform(0,1).\end{aligned} \tag{S13}$$

Under this re-parametrisation, the distribution of $\epsilon_{l,i,u}$ is smooth for $\tau > 0$ and $p(\epsilon_{l,i,u}) \to Bernoulli(p_{l,i,u})$ as τ approaches 0 and. Therefore, we have well-defined gradients w.r.t. the probability $p_{l,i,u}$ by using a small τ. Combining Eq. S12 and S13, we can obtain the gradients presented by the Eq. (5) in the paper.

1.2. Equation 7 in the paper

We use v_0 to denote index (l_0, i_0, u_0) for simplicity. It is straightforward to derive that

$$\begin{aligned}&\mathcal{P}(\widehat{\mathcal{M}}(v_0) = 0 \mid \mathbb{D})\\ =&\int_{W_T(v_0)}\mathcal{P}(\widehat{\mathcal{M}}(v_0) = 0, W_T(v_0) \mid \mathbb{D})\\ =&\int_{W_T(v_0)}\mathcal{P}(\widehat{\mathcal{M}}(v_0)\cdot W_T(v_0) = 0 \mid \mathbb{D}).\end{aligned} \tag{S14}$$

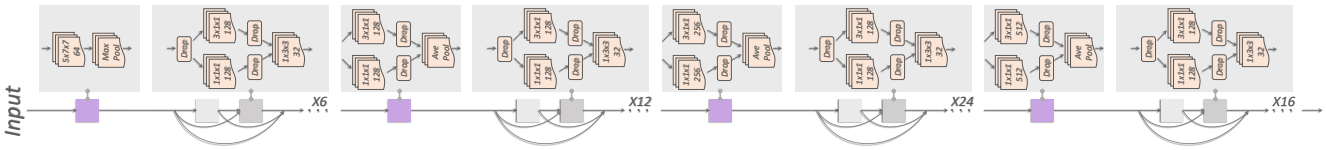


Figure 1: Architecture of the template network used in our method. The notion 'x6' indicates that the basic module (a single grey box) is repeated by six times. The spatial stride size used for all the convolution and pooling operations is one and two, respectively. The temporal stride size is always set to be one.

Once the variational distribution Q is available, it is easy to derive that

$$\begin{aligned}
&\mathcal{P}(\widehat{\mathcal{M}}(v_0)\cdot W_T(v_0)\mid\mathbb{D})\\
=&\int_{\widehat{\mathcal{M}}(v)\cdot W_T(v),v\neq v_0}\mathcal{P}(\widehat{\mathcal{M}}(v)\cdot W_T(v)\mid\mathbb{D})\\
\approx&\int_{\widehat{\mathcal{M}}(v)\cdot W_T(v),v\neq v_0}\prod_v q(\widehat{\mathcal{M}}(v)\cdot W_T(v))\\
=&\int_{\widehat{\mathcal{M}}(v)\cdot W_T(v),v\neq v_0} q(\widehat{\mathcal{M}}(v_0)\cdot W_T(v_0))\prod_{v\neq v_0} q(\widehat{\mathcal{M}}(v)\cdot W_T(v))\\
=&q(\widehat{\mathcal{M}}(v_0)\cdot W_T(v_0))\int_{\widehat{\mathcal{M}}(v)\cdot W_T(v),v\neq v_0}\prod_{v\neq v_0} q(\widehat{\mathcal{M}}(v)\cdot W_T(v))\\
=&q(\widehat{\mathcal{M}}(v_0)\cdot W_T(v_0)).
\end{aligned}\tag{S15}$$

According to Eq. S7, S14 and S15, we have

$$\begin{aligned}
&\mathcal{P}(\widehat{\mathcal{M}}(v_0)=1\mid\mathbb{D})\\
=&1-\mathcal{P}(\widehat{\mathcal{M}}(v_0)=0\mid\mathbb{D})\\
=&1-\int_{W_T(v_0)}\mathcal{P}(\widehat{\mathcal{M}}(v_0)\cdot W_T(v_0)=0\mid\mathbb{D})\\
\approx&1-\int_{W_T(v_0)} q(\widehat{\mathcal{M}}(v_0)\cdot W_T(v_0)=0)\\
=&1-\widehat{p_{v_0}}.
\end{aligned}\tag{S16}$$

In practice we optimize $p_{l,i,u}=\widehat{p}_{l,i,u}\cdot\widetilde{p}_{l,i,u}$ as a whole, and the $\widehat{p}_{l,i,u}$ and $\widetilde{p}_{l,i,u}$ are initialized with the same value, therefore, we have

$$\widehat{p_{v_0}}=\sqrt{p_{v_0}}.\tag{S17}$$

Hereby, we have the posterior probability of fusion unit at each layer, which can be used as numerical measurements for the layer-level importance of the fusion units as described in the paper.

2. More Details on Experimental Setups

2.1. Template Network

We visualize the complete structure of the template network in Fig. 1. As can be viewed in the figure, each layer in the template network contains three convolution operations which is used to derive the three basic fusion units as illustrated in the Fig. 3 in the paper. The feature map output from each layer will be used as input for all the succeeding layers, which forms a densely-connected 3D network. Similar to [6], we also insert reduction block for memory efficiency.

2.2. Figure 3

Fig. 3 in the paper is a part of the whole template network (Fig. 1 in this supplementary material). It illustrates how the basic units are derived from the template network. Each grey block refers to a module that contains three convolutions as well as three drop-path operations (D1, D2, and D3). By activating each drop-path operation w.r.t. its probability, we can derive three basic fusion units, i.e., S (activate D3 only), ST (activate D2 only) and S+ST (no activation), plus one residual operation (activate D1). This design helps us to construct the probability space via network training with DropPath, as described in the Section 3.2 in the paper.

2.3. Basic Fusion Units

There two reasons for $U=\{S,ST,S+ST\}$ chosen instead of the full set, *i.e.*, $U=\{S,T,ST,S{+}T,S{+}ST,T{+}ST,S+T+ST\}$. 1) Hardware constraint. Something and Kinetics dataset both have more than 200K videos and the training time for a single trial is two weeks. If we consider all possible combinations in template network, the parameter size and training time would increase around 25% and 40%, respectively, which results in one more week for training. Besides, we have to reduce batch size from 32 to 16, which also affects the training speed and convergence badly. We can not afford such computational cost given the limited hardware resources. Therefore, we choose to only investigate the three widely investigated fusion units [25,35] in this paper. 2) Fair comparison. As discussed in Section 4.1 in the paper, we prefer that the fusion units explored in our approach are conceptually included in most of other fusion methods for fair comparison with state-of-the-arts in terms of efficiency and effectiveness.

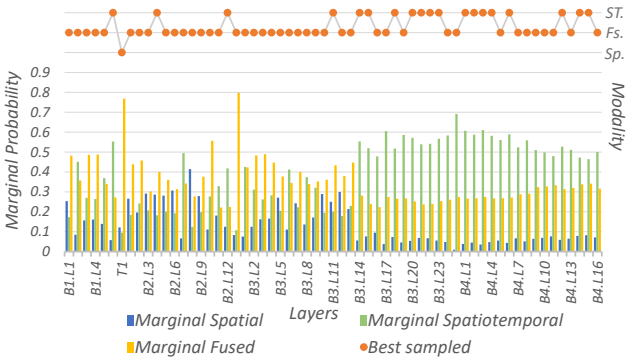
3. More Results on Generalization

In order to justify that the observations obtained from the probability space can generalize, we construct new spatiotemporal strategies based on the observations discussed in Section. 4.4 in the paper, but with five very different backbone networks. They are DenseNet121[6], ResNet50[5], MobileNetV2[10], ResNeXt50[11], and ResNeXt101[11], respectively. They differ from each other in terms of topology, parameter size and FLOPs. We inflate them into 3D CNNs with the same module visualized in our template network. We compare four different fusion strategies on each backbone, i.e., optimized(Opt), fused(S+ST), spatial(S) and spatiotemporal(ST). 'Opt' means we follow the observations to design the fusion strategies (except for Densenet we directly use the best one sampled from the probability space). Please note that we can only roughly follow the observations because the network topology varies from backbone to backbone. 'S+ST' indicates that we employ S+ST convolution all the way. 'Spatial' and 'spatiotemporal' indicate using spatial convolutions and spatiotemporal convolutions all the way, respectively. Please also note that there are 3D poolings existing in the 'spatial' mode, so it is not pure 2D network. We report clip-level performance in this section for quick comparison.

On Something-Something V1 and Something-Something V2, we follow the Observation I and II to construct the strategy 'Opt' where fused convolutions are used for the first half and the last three layers of network, and spatiotemporal convolutions are applied on the remaining layers. As can be viewed from the Fig. 2(b) and 3(b), although backbones are quite different, all the networks perform better with the optimized fusion strategy. One exception is MobileNetV2, where the 'Opt' strategy is slightly worse than the fused mode. We think the reason is that MobileNetV2 is too small to fit this large dataset and any kind of bonus on the parameter size would help improve the performance greatly. The 'S+ST' mode contains 7 more 2D convolutions than the 'Opt' mode in the MobileNetV2 backbone.

Similar results can be observed on UCF101, where the 'Opt' strategy makes all the four backbone networks outperform their counterparts, which is consistent with the Observation III. Please note that the optimized strategy is equivalent to the fused strategy on UCF101 according to the Fig. 4.

On Kinetics400, we roughly follow the Observation I II and III as well as the patterns in Fig. 5(a) to use the S+ST convolutions and S convolutions periodically in the first two third of network and ST convolutions in the remaining layers for the strategy 'Opt'. We can see from the Fig. 5(b) that 'Opt' strategy still performs the best on different backbones. Due to the limited, we can not evaluate the S+ST strategy on Kinetics400 with the 3D ResNeXt101 backbone.



(a) Figure: Marginal probability of layer-level fusion units.

Net. \ Mod.	Opt	S+ST	S	ST
3D Dense.121	50.2	46.5	41.8	47.5
3D ResNet50	41.2	38.9	33.8	40.1
3D ResNeXt50	43.6	40.7	35.2	42.1
3D ResNeXt101	44.0	42.3	36.6	42.7

(b) Table: The performance of different backbones constructed with the spatiotemporal hints and their counter-parts.

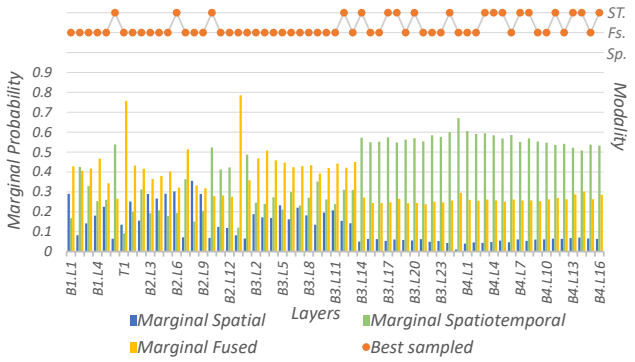
Figure 2: Spatiotemporal fusion hints obtained from the probability space on Something-Something V1.

We also implement a small experiment to illustrate the learned probability space can help the spatiotemporal fusion strategy capture the character of the data. More specifically, we reduce the temporal resolution of input video clips by employing temporal pooling with a stride of 2 and window size of 3. We draw the corresponding marginal probability and best-sampled fusion strategy on Something V1 in Fig. 6. It can be easily observed that the layer-level preference changes accordingly. There are more spatial convolutions and less spatiotemporal convolutions utilized in the best-sampled strategy when compared with Fig. 2 in which no additional temporal pooling is used.

4. More Discussions on NAS

We believe that our proposed algorithm can be extended for NAS. But in this paper, it is specially designed for analyzing spatiotemporal fusion from a probabilistic view. To ensure the whole scheme is theoretically sound and correct, we have to construct a valid probability space to facilitate efficient and effective analysis with numerical measurements. To our best knowledge, no published NAS papers manage to formulate and construct such probability space.

Additionally, existing NAS methods have shortcoming in spatiotemporal analysis. For example, differentiable NAS methods such as DARTS[7] assign a scalar unit (regularized by softmax function) to each operation and jointly optimize weight and the unit in an alternative fashion. Only

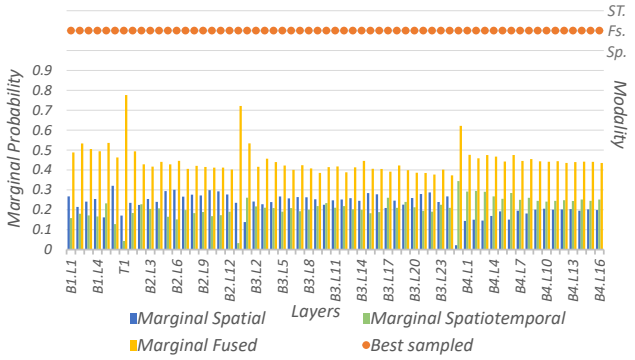


(a) Figure: Marginal probability of layer-level fusion units.

Net. \ Mod.	Opt	S+ST	S	ST
3D Dense.121	62.4	59.5	55.1	60.5
3D Mobile.v2	59.5	59.7	52.9	59.3

(b) Table: The performance of different backbones constructed with the spatiotemporal hints and their counter-parts.

Figure 3: Spatiotemporal fusion hints obtained from the probability space on Something-Something V2.



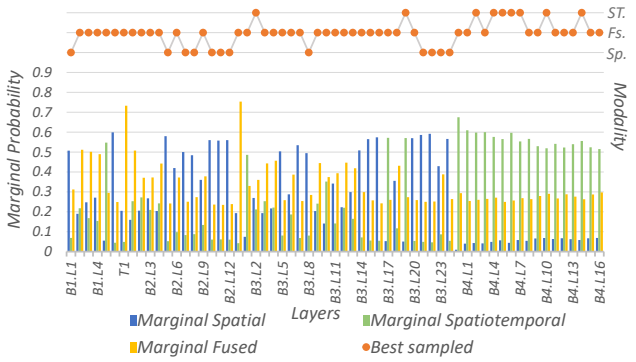
(a) Figure: Marginal probability of layer-level fusion units.

Net. \ Mod.	Opt	S+ST	S	ST
3D Dense.121	84.2	84.2	83.6	83.1
3D ResNet50	82.4	82.4	81.2	81.6
3D Mobile.v2	81.8	81.8	81.3	80.8
3D ResNeXt50	85.1	85.1	83.9	82.9

(b) Table: The performance of different backbones constructed with the spatiotemporal hints and their counter-parts.

Figure 4: Spatiotemporal fusion hints obtained from the probability space on UCF101.

one single architecture is derived by selecting the operation that has the largest activation in each layer and the weight needs to be trained from scratch to produce final performance. It can not properly sample a group of diverse architectures to be directly evaluated for further fusion analysis. Sampling-based NAS methods such as [1, 4] randomly sample different sub-networks and naively inherit the kernel weights from the template network. A group of candidates can be obtained in this way. However, the inherited weight may not match the sampled architectures and thus the per-

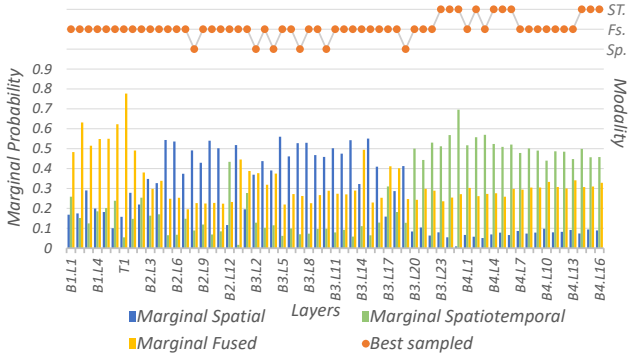


(a) Figure: Marginal probability of layer-level fusion units.

Net. \ Mod.	Opt	S+ST	S	ST
3D Dense.121	71.7	69.7	67.8	68.3
3D ResNeXt101	72.0	-	70.6	70.9

(b) Table: The performance of different backbones constructed with the spatiotemporal hints and their counter-parts.

Figure 5: Spatiotemporal fusion hints obtained from the probability space on Kinetics400.



(a) Figure: Marginal probability of layer-level fusion units.

Net. \ Mod.	Opt	S+ST	S	ST
3D Dense.121	44.2	40.1	38.8	41.1

(b) Table: The performance of Densenet121 constructed with the spatiotemporal hints and their counter-parts

Figure 6: Spatiotemporal fusion hints obtained from the probability space on Something-Something V1 with less temporal resolution.

formance of each sub-network does not match the ideal performance as well, which suggests it can not be used to facilitate an effective analysis on spatiotemporal fusion in 3D CNNs. As another example, Evolution-based NAS methods iteratively derive new child architectures from population. Due to the high complexity of evolution algorithm, it cannot facilitate fine-grained layer-level spatiotemporal analysis. For instance, evolved module is repeated several times to form a homogeneous network which has few variations in terms of network-level structure diversity [8], and only connectivity among several predefined modules can be evolved [9].

References

[1] Gabriel Bender. Understanding and simplifying one-shot architecture search. 2019. 5

[2] Yarin Gal. *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge, 2016. 2

[3] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016. 1

[4] 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. *arXiv preprint arXiv:1904.00420*, 2019. 5

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[6] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 3, 4

[7] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 4

[8] AJ Piergiovanni, Anelia Angelova, Alexander Toshev, and Michael S Ryoo. Evolving space-time neural architectures for videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1793–1802, 2019. 6

[9] Michael S Ryoo, AJ Piergiovanni, Mingxing Tan, and Anelia Angelova. Assemblenet: Searching for multi-stream neural connectivity in video architectures. *arXiv preprint arXiv:1905.13209*, 2019. 6

[10] 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*, pages 4510–4520, 2018. 4

[11] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 4