## A. Multi-View Classification Dataset

We also evaluate SplitNets on a multi-view system with ModelNet 3D Warehouse classification dataset that consists of 12,311 CAD models from 40 categories. CAD models in this benchmark come from 3D Warehouse [1]. For each object, 12 views are obtained by rotating the object every 30 degrees along the gravity direction. Using multiple views is necessary to obtain high classification accuracy according to prior work [16]. We use the same training and testing split as in [57]. For our experiments, the reported metric is per-class accuracy.

## B. Interlacing Concatenation for View Fusion

In Section 3.1.3, we introduce a splitting module for multi-view systems, consisting of two layers *Conv-Reduce* and *View-Fuse*. Since we mainly focus on searching the position of *View-Fuse* (*i.e.*, the position of splitting module), we introduce a simple fusion operation - concatenation. More specifically, we concatenate features from $V$ views in an interlacing way along the channel dimension as illustrated in Figure 10. In the interlacing concatenation, we first concatenate the first channel from $V$ views, then repeat the same operation for the second channels, and so on. We adopt interlacing concatenation because the number of channels of each view is dynamically sampled during training and interlacing concatenation guarantees that the $i$-th channel of the fused features is always from the $(i \bmod V)$-th view.

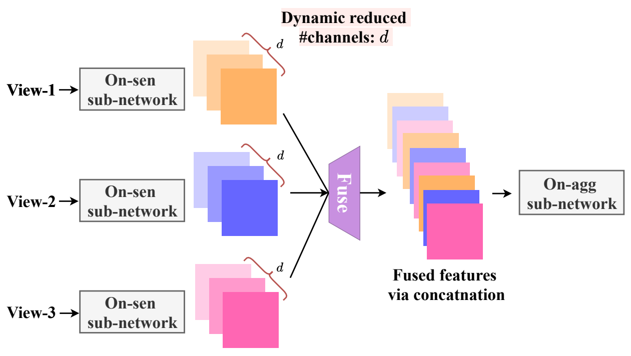

Figure 10. Illustration of interlacing concatenation. Interlacing concatenation guarantees that the $i$-th channel of the fused features is always from the $(i \bmod V)$-th view. In the example of this figure, $V = 3$.

## C. Supernet Configuration

As we mentioned in Section 3, We build the supernet's search spaces mainly based on [9, 54] with some extra search spaces from our SA-NAS as shown in Figure 3. Search spaces from standard NAS [9, 54] include number of layers, output channel size and expand ratio of each inverted residual block (MB). Following [9, 54], we use Swish as activation function. The supernet architecture configuration and search space for single-view SplitNets on ImageNet are summarized in Table 7.

## D. Sampling Strategy of Splitting Modules for Supernet Training

### D.1. Single-View SplitNets

In supernet training, we jointly train multiple sub-networks sampled from the supernet at each iteration. Various sampling strategies can be used [5, 9, 54, 60]. For example, BigNAS [60] samples three kinds of networks: 1) the largest sub-network ("max"); 2) the smallest sub-network ("min"); 3) several randomly sub-sampled networks. Since the largest sub-network usually has better accuracy, it can be used as teacher to supervise other sub-networks via knowledge distillation losses [23]. Some variants of this "sandwich" sampling strategy are also proposed like sampling sub-networks according to their predicted accuracy [54].

Compared with previous work, we have different interests on sub-networks. Eventually, we want to find the optimal architecture with exact one splitting module. However, as an information bottleneck, splitting module will inevitably introduce accuracy degradation. During training, we aim to minimize this degradation by improving the accuacy lower bound. We insert multiple splitting modules to help sub-networks increase tolerance of splitting modules. Inspired by BigNAS [60] and AttentiveNAS [54], we sample five sub-networks per iteration to train jointly.

1. "Max with zero hot": The largest sub-network without splitting module can be treated as the best Pareto architecture. Training this sub-network helps improve the accuracy upper bound of all sub-networks.

2. "Max with all hot": The largest sub-network with all $N$ splitting modules contains all weights in the supernet. Training this sub-network ensures all weights are updated at least once per iteration.

3. "Min with zero hot": The smallest sub-network without splitting module contains weights which are shared across all sub-networks. Training this sub-network helps the optimization of the most frequently shared weights.

4. "Min with all hot": The smallest sub-network with all $N$ splitting modules can be treated as the worst Pareto architecture. Training this sub-network helps improve the accuracy lower bound of the all sub-networks.

5. 'Random with one hot": A randomly sampled sub-network with exact one splitting module is the sub-network we are interested in during the second stage and final deployment.

| Input Resolution | | {192, 224, 256, 288} | | | | |
|---|---|---|---|---|---|---|
| Model Phase | Block | NAS Search Space | | SA-NAS Search Space | | |
| Model Phase | Block | Channel | Expansion Ratio | Depth Before Split Module | Depth After Split Module | Reduced channel $d$ Module |
| - | Conv | {16,24} | {1} | - | - | - |
| Phase 1 | MBConv-1 | {16,24} | {1} | {2,3,4,5} | {1,2,3} | {4,6,8} |
| | MBConv-2 | {24,32} | {4,5,6} | | | |
| Phase 2 | MBConv-3 | {32,40} | {4,5,6} | {1,2,3} | {1,2,3} | {6,8,10} |
| Phase 3 | MBConv-4 | {64,72} | {4,5,6} | {1,2,3} | {4,5,6,7,8,9} | {10,14,18} |
| | MBConv-5 | {112,120,128} | {4,5,6} | | | |
| Phase 4 | MBConv-6 | {192,200,208,216} | {6} | {1,2,3,4} | {2,3,4,5,6} | {16,24,32} |
| | MBConv-7 | {216,224} | {6} | | | |
| - | MBPool | {1792,1984} | {6} | - | - | - |

Table 7. The supernet architecture configuration and search spaces for single-view SplitNets on ImageNet.

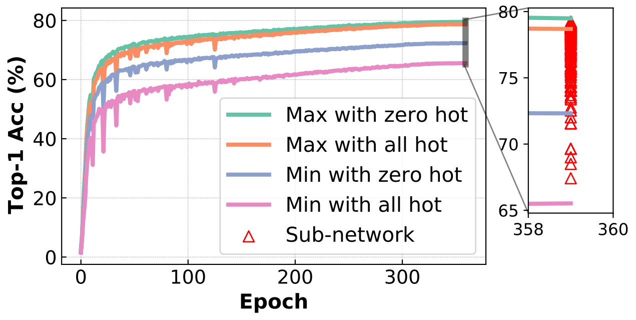

Figure 11. Training curves of four different sub-networks sampled from the supernet for single-view SplitNets. "max" (or "min") indicts the maximum (or "minimum") sub-network sampled from the supernet. "all hot" (or "zero hot") means that the sampled sub-network contains all possible (or none of) splitting modules. The drop of performance introduced (green line against orange line and blue line against magenta line) by inserting splitting module is mitigated. The red triangles are sampled sub-networks with one splitting module (*i.e.*, "Random with one hot"). The green and pink lines can be treated as the upper and lower bounds of sub-networks.

We visualize training curves of aforementioned sub-networks in Figure 11. The red triangles are some sampled sub-networks with one splitting module. Their accuracy is bounded by the "Max with zero hot" and "Min with all hot" sub-networks.

In addition, we compare our new sampling strategy against the baseline "sandwich" sampling strategy, which replaces "Max with all hot" (and "Min with all hot") with "Max with one hot" (and "Min with one hot") during supernet training. From Figure 12, we find that sampling more than one splitting modules during training helps reduce accuracy drop introduced by splitting modules in SplitNets. For example, the discrepancy between solid blue and pink lines is significantly smaller than discrepancy between

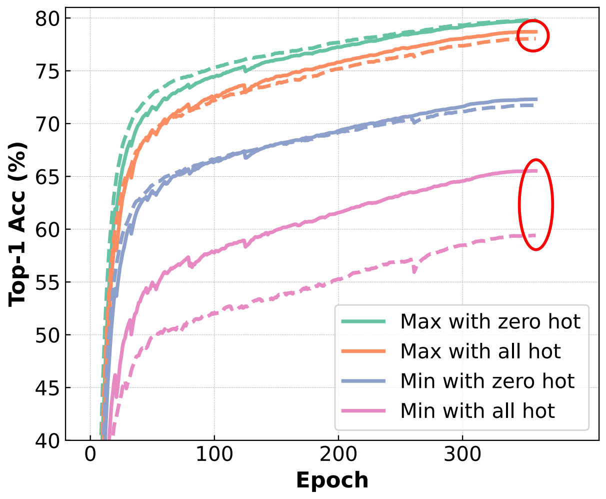

Figure 12. Comparison between our new sampling strategy (solid lines) and baseline "sandwich" sampling strategy (dashed lines) for single-view SplitNets as elaborated in Appendix D.1. Our new sampling strategy helps reduce accuracy drop in SplitNets due to splitting modules. For example, the discrepancy between solid blue and pink lines is significantly smaller than discrepancy between dashed blue and pink lines.

dashed blue and pink lines.

## D.2. Multi-View SplitNets

For multi-view SplitNets, each splitting module contains a fusion operation which can only be performed once for one network. So we are not able to sample more than one splitting modules during neither supernet training nor architecture searching. We use the baseline "sandwich" sampling strategy for multi-view SplitNets.

## E. Analysis of Different Initializations

As we discussed in Section 3.1.3, both forward and backward passes of a convolution layer can be expressed by convolution operations. The goal of initialization is to ensure the magnitude of output (and gradient) does not explode during forward (and backward) pass as shown in the following two equations,

$$\texttt{CONV}_{\text{forward}}\left(\mathbf{W}, \mathbf{x}_l\right) = \mathbf{y}_l \sim \mathcal{N}(0,1) \tag{2}$$

$$\texttt{CONV}_{\text{backward}}\left(\mathbf{W}^T, \frac{\partial \mathcal{L}}{\partial \mathbf{y}_l}\right) = \frac{\partial \mathcal{L}}{\partial \mathbf{x}_l} \sim \mathcal{N}(0,1), \tag{3}$$

where we assume the activation function as ReLU [22], $\mathbf{x}_{l+1} = \max(\mathbf{y}_l, 0)$. Solving Equation (2) or Equation (3) leads to Kaiming Fan-In or Kaiming Fan-Out [21],

$$\mathbf{W} \sim \mathcal{N}\left(0, \frac{2}{k^2 \cdot c_{\text{in}}}\right) \qquad (\textsc{Fan-In}) \tag{4}$$

$$\mathbf{W} \sim \mathcal{N}\left(0, \frac{2}{k^2 \cdot c_{\text{out}}}\right) \qquad (\textsc{Fan-Out}), \tag{5}$$

where $k$ is kernel size and $c_{\text{in}}$ (and $c_{\text{out}}$) is the input (and output) channel size.

In a splitting module, the difference between input and output channel sizes is usually enormous. For example, in a certain *Conv-Reduce*, we have $c_{\text{in}} = 256$ and $c_{\text{out}} = 8$. In this case, Fan-In mode's weights variance is $\frac{256}{8} = 32$ times larger than Fan-Out's. Choosing either Fan-In or Fan-Out will cause the other one's variance too large or too small. Although Xavier [17]'s arithmetic average of $c_{\text{in}}$ and $c_{\text{out}}$ may mitigate this issue, it is far from enough because arithmetic average between two numbers is dominated by the larger one, $0.5 \cdot (256 + 8) \gg 8$.

Our split-aware initialization adopts geometric average instead of arithmetic average to make a better balance between forward and backward, $\sqrt{c_{\text{in}} \cdot c_{\text{out}}}$. In the next section, we empirically show that supernet training can benefit from our split-aware initialization.

## F. Empirical Comparison of Different Initializations

We compare our split-aware initialization against Kaiming initialization (Fan-In) on "Max with all hot" sub-network's accuracy for ImageNet in Figure 13. Our split-aware initialization improves training stability as well as final accuracy (by 0.3%). In addition, if the base learning rate (0.05 in Figure 13) is slightly increased, conventional initialization will lead to divergence in training.

## G. Hardware Modeling

As discussed in Section 3.2, we use a hardware simulator customized for a realistic head mounted device. The on-sen. processor is equipped with a 16nm neural processing unit (NPU) with peak performance of $\text{Comp}_{\texttt{sen}} = 125$ GOP/s. So, the latency of on-sensor part (in Equation (1)) can thus be approximately computed via $T_{\texttt{sen}}(f_{\texttt{sen}}, \mathbf{x}) = \text{OP}(f_{\texttt{sen}}, \mathbf{x}) / \text{Comp}_{\texttt{sen}}$, where $\text{OP}(\cdot,\cdot)$ is the profiling function through cycle-accurate simulation for measuring the number of operations given model and input. In addition, the on-sen. processor's peak memory is $\text{Mem}_{\texttt{sen}} = 2$ MB. Thus, the peak memory consumption of the on-sen part cannot exceed this peak memory constraint $M(f_{\texttt{sen}}, \mathbf{x}) \leq \text{Mem}_{\texttt{sen}}$. When computing the peak memory consumption ($M(\cdot,\cdot)$), we consider the memory of both weights and activations: $M(f_{\texttt{sen}}, \mathbf{x}) = M_{\texttt{W}}(f_{\texttt{sen}}) + M_{\texttt{A}}(f_{\texttt{sen}}, \mathbf{x})$, where $M_{\texttt{W}}(f_{\texttt{sen}})$ is the memory consumption for storing all weights of $f_{\texttt{sen}}$ and $M_{\texttt{A}}(f_{\texttt{sen}}, \mathbf{x})$ measures the peak memory consumption of activation taking residual connections into consideration. In this work, we consider homogeneous sensors which can represent most of AR/VR devices like Quest2 [2]. We leave the extension of SplitNets as a future work when heterogeneous sensors occur.

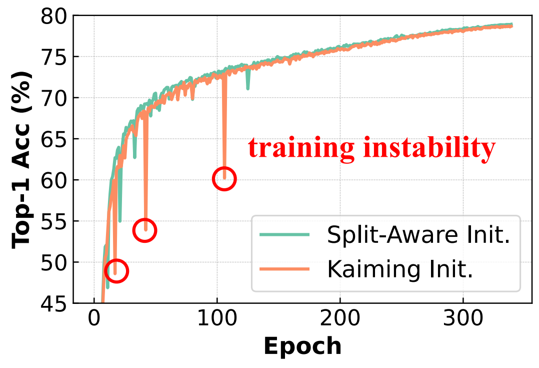

Figure 13. Accuracy comparison between Kaiming and our split-aware initalizations on the "Max with all hot" sub-network.

## H. Adaptability of SplitNets

As we discussed in Section 3, two-stage NAS decouples the supernet training (stage 1) and architecture searching (stage 2). As a result, when the hardware configuration changes, one just needs to rerun the stage 2 without retraining the supernet. In this section, we change the hardware configuration and observe how the searched architectures evolve to fit the change of hardware configuration. Specifically, we increase the computation capability of on-sen. processors by four times and show the change of architectures with the best accuracy for the multi-view task in Figure 14. The left architecture is the best network for the default hardware configuration ($\text{Comp}_{\texttt{sen}} = 125$ GOP/s) from Table 6 (SplitNets-C). If on-sen. processors' peak computation performance is increased by $4\times$, SA-NAS can

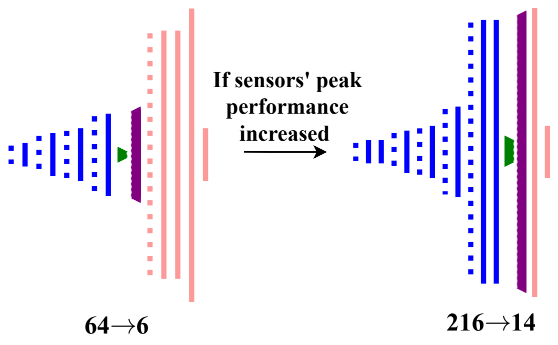

Figure 14. Evolution of the best architectures when on-sen. processor's peak computation performance increases. SA-NAS can automatically put more and wider layers on sensors.

automatically put more and wider layers on-sen. and reduce the number of on-agg. layers to make a better trade-off. The network on the right achieves a better performance (94.0% top-1 accuracy) and lower latency (0.62 ms) compared with the left one.