A. Experiment details

We describe more experiment details in this appendix to facilitate other researchers to reproduce our work. Our architecture search is divided into two stages. In the first stage, we train the stochastic super net to find an optimal architecture distribution. In the second stage, we sample architectures from the distribution and train them from scratch.

To train the stochastic super net, we randomly sample 100 classes from the original 1,000 classes of ImageNet. Training the super net on this smaller proxy dataset is much faster. We train the stochastic super net for 90 epochs with a batch size of 192. In each epoch, we first train the operator parameters w_a on 80% of the training set using stochastic gradient descent with momentum. The initial learning rate is 0.1, and decay following a cosine decaying schedule. The momentum is 0.9, and weight decay is 10^{-4}. Next, we train the architecture distribution parameter $\boldsymbol{\theta}$ on the rest 20% of the training set with Adam optimizer [10] with a learning rate of 10^{-2} and weight decay of 5×10^{-4}. The split of weight and architecture parameter training ensure the architecture generalize to the validation dataset. To control the Gumbel Softmax in (8), we use an initial temperature of 5.0 and exponentially anneal it by $exp(-0.045) \approx 0.956$ every epoch. For the loss function in (2), we set α to 0.2 and β to 0.6. We use the standard ResNet data augmentation [4] to process the input images. We found that at the beginning of the training, operators are usually not sufficiently trained, so their contributions to the accuracy are not clear. However, their costs are always significantly different from each other. As a consequence, the super net may always pick low-cost operators at the beginning of the training. To prevent this, we postpone the training of the architecture parameter $\boldsymbol{\theta}$ by 10 epochs to allow operator weights to be sufficiently trained first. At the end of the super net training, we sample 6 architectures from the final distribution to be trained from scratch.

To train the sampled architectures, the training protocols are different for different models. Here we describe the training protocol for FBNet-$\{$A, B, C$\}$. These models have an input resolution of 224, channel size scaling of 1.0. We train the models with a batch size of 256 on 8 GPUs for 360 epochs. We set the initial learning rate to be 0.1, and decay 10x at 90, 180, and 270 epochs. The momentum is 0.9, weight decay is 4×10^{-5}. We use dropout at the last convolution layer of the network, and the dropout ratio is 0.2. We use the standard GoogleNet data augmentation [19] to randomly resize the image during training.