A. APPENDIX

We broke model leaf node layers into seven categories across the six examined architectures: Conv2d, BatchNorm2d, ReLu, MaxPool2d, Linear, Dropout, and AveragePool2d. For each trial, a model's leaf node layers were first sorted into these categories regardless of parameters. The aggregate runtime and run count were calculated for each category by summing the individual runtimes of leaf node layers in the same category, and by counting every time a leaf node layer of a specific category was run. Finally, aggregate FLOPS per category were calculated using the results from the THOP library, summing the FLOPS per leaf node layer for leaf node layers in the same category. After finding aggregate runtime, run count, and FLOPS per category for each trial, these values were averaged across all trials, as summarized by the following equations, where t is trials, n_t is the nth (and last) leaf node layer to run in trial t, c is a layer category, r_c is average runtime for category c, k_c is average run count for category c, and m_c is average FLOPS for category c:

$$r_c = \frac{1}{5}\sum_{t=1}^{5}\sum_{l=1}^{n_t}\text{runtime}(l)\{l \in C\}$$

$$k_c = \frac{1}{5}\sum_{t=1}^{5}\sum_{l=1}^{n_t}\{l \in C\}$$

$$m_c = \frac{1}{5}\sum_{t=1}^{5}\sum_{l=1}^{n_t}\text{FLOPS}(l)\{l \in C\}$$

A.1. Additional Model descriptions

Depthwise Seperable Convolutional Neural Network (DSCNN) [**?**] DSCNN consists of MBConvReLU blocks where each MBConvReLU block contains one pointwise MBConvReLU block, one depthwise MBConvReLU block, and one pointwise linear convolution layer. The DSCNN structue as a whole begins with a stride-2 convolution layer, goes into a sequence of inverted residual bottleneck layers, and exits into a stride-1 convolution layer and final linear layer.

DenseNet [**?**] Densenet consists of 1x1 bottleneck convolution block is formed by a 1x1 bottleneck convolution layer with stride 1 sandwiched between two batch normalizations, followed by a 3x3 convolution layer of stride 1 and finally one last batch normalization and a ReLU layer . Transition convolution blocks are formed by a batch normalization - ReLU - 1x1 stride 1 convolution - average pooling sequence. The alternating bottleneck and transition layers are preceded by a 3x3 convolution layer with stride 1, and succeeded by a linear layer.

Convolutional Neural Network (CNN) [**?**] We utilizing a repeating sequence of stride 1 convolution, batch normalization, and ReLU layers followed by three linear layers.

MobileNet (DSCNN) [**?**] MobileNet was designed with efficiency in mind, as an answer to the practical challenge of implementing neural networks on platforms with less

are preceded by a 3x3 convolution layer with stride 1, and succeeded by a linear layer.

mnasNet [**?**] MNasNet uses training-aware neural architecture search (NAS), cnvolutions, depthwise separable convolutions, sequences of inverted residuals (pointwise, depthwise, linear pointwise), and finally a linear layer

EfficientNet [**?**] EfficientNet merges sequences of 'MBConv' blocks consisting of convolution and batchnorm layers with NAS, which is designed to select the best configuration. Ends with a linear layer We use scalings 1 and 7.

EfficientNet-V2 [**?**] These models were searched from the search space enriched with new ops such as Fused-MBConv. A combination of training-aware neural architecture search (NAS) and scaling was used to improve both training speed and parameter efficiency