A. Proof of Theorem 1

According to [13,42,46,64], we have the following lemmas.

Lemma 1. *The differential entropy $H(x)$ of random variable $x \sim \mathcal{N}(0, \sigma)$ is*

$$H(x) \propto log(\sigma^2). \tag{6}$$

Lemma 2. *For any random variable x, its differential entropy $H(x)$ is bounded by its Gaussian entropy upper bound*

$$H(x) \leq c \log[\sigma^2(x)], \tag{7}$$

where $c > 0$ is a universal constant, $\sigma(x)$ is the standard deviation of x.

Lemma 3. *For N random variables $\{x_1, \cdots x_i, \cdots x_N\}$, the laws of expectation and variance of sum of random variables are*

$$\mathbb{E}(\sum_1^N x_i) = \sum_1^N \mathbb{E}(x_i), \tag{8}$$

$$\sigma^2(\sum_1^N x_i) = \sum_1^N \sigma^2(x_i). \tag{9}$$

The laws of expectation and variance of product of random variables are

$$\mathbb{E}(\prod_1^N x_i) = \prod_1^N \mathbb{E}(x_i), \tag{10}$$

and

$$\sigma^2(x_i x_j) = \sigma^2(x_i)\sigma^2(x_j) + \sigma^2(x_i)\mathbb{E}^2(x_j) + \sigma^2(x_j)\mathbb{E}^2(x_i). \tag{11}$$

Suppose that in an L-layer MLP $f(\cdot)$, the i-th layer has w_i input channels and w_{i+1} output channels. The trainable weights in i-th layer is denoted by $\mathbf{M}_i \in \mathbb{R}^{w_{i+1} \times w_i}$. For simplicity, we assume that each element $\mathbf{x}_1^j$ in $\mathbf{x}_1$ and each element $\mathbf{M}_i^{j,k}$ in $\mathbf{M}_i$ follow the standard normal distribution, i.e.

$$\mathbf{x}_1^j \sim \mathcal{N}(0, 1), \tag{12}$$

$$\mathbf{M}_i^{j,k} \sim \mathcal{N}(0, 1). \tag{13}$$

According to Eqs. (8) to (12), in i-th layer, the output $\mathbf{x}_{i+1}$ and the input $\mathbf{x}_i$ are connected by $\mathbf{x}_{i+1} = \mathbf{M}_i \mathbf{x}_i$. The expectation of j-th element in $\mathbf{x}_{i+1}$ is

$$\begin{aligned}
\mathbb{E}(\mathbf{x}_{i+1}^j) &= \mathbb{E}(\sum_{k=1}^{w_i} \mathbf{M}_i^{jk} \mathbf{x}_i^k) \\
&= \sum_{k=1}^{w_i} \mathbb{E}(\mathbf{M}_i^{jk} \mathbf{x}_i^k) \\
&= \sum_{k=1}^{w_i} \mathbb{E}(\mathbf{M}_i^{jk}) \mathbb{E}(\mathbf{x}_i^k) \\
&= 0.
\end{aligned} \tag{14}$$

According to Eqs. (8) to (14), the variance of j-th element in $\mathbf{x}_{i+1}$ is

$$\begin{aligned}
\sigma^2(\mathbf{x}_{i+1}^j) &= \sigma^2(\sum_{k=1}^{w_i} \mathbf{M}_i^{jk} \mathbf{x}_i^k) \\
&= \sum_{k=1}^{w_i} \sigma^2(\mathbf{M}_i^{jk} \mathbf{x}_i^k) \\
&= \sum_{k=1}^{w_i} \{\sigma^2(\mathbf{M}_i^{jk})\sigma^2(\mathbf{x}_i^k) + \sigma^2(\mathbf{M}_i^{jk})\mathbb{E}(\mathbf{x}_i^k) \\
&\qquad + \sigma^2(\mathbf{x}_i^k)\mathbb{E}(\mathbf{M}_i^{jk})\} \\
&= \sum_{k=1}^{w_i} \sigma^2(\mathbf{x}_i^k) \\
&= w_i \sigma^2(\mathbf{x}_i^k).
\end{aligned} \tag{15}$$

With the variances propagating in networks and $\mathbf{x}_1^j \sim \mathcal{N}(0, 1)$ in Eq. 12, the variance of j-th element in L-th layer is

$$\sigma^2(\mathbf{x}_L^j) = \prod_{i=1}^{L} w_i \tag{16}$$

According to Eq. 6, the entropy of each element x_L^j of L-th MLP is

$$\begin{aligned}
H(\mathbf{x}_L^j) &\propto \log(\prod_{i=1}^{L} w_i), \\
&= \sum_{i=1}^{L} \log(w_i).
\end{aligned} \tag{17}$$

After considering the width of the output feature vector, the normalized Gaussian entropy upper bound of the L-th feature map of MLP $f(\cdot)$ is

$$H_f = w_{L+1} \sum_{i=1}^{L} \log(w_i). \tag{18}$$

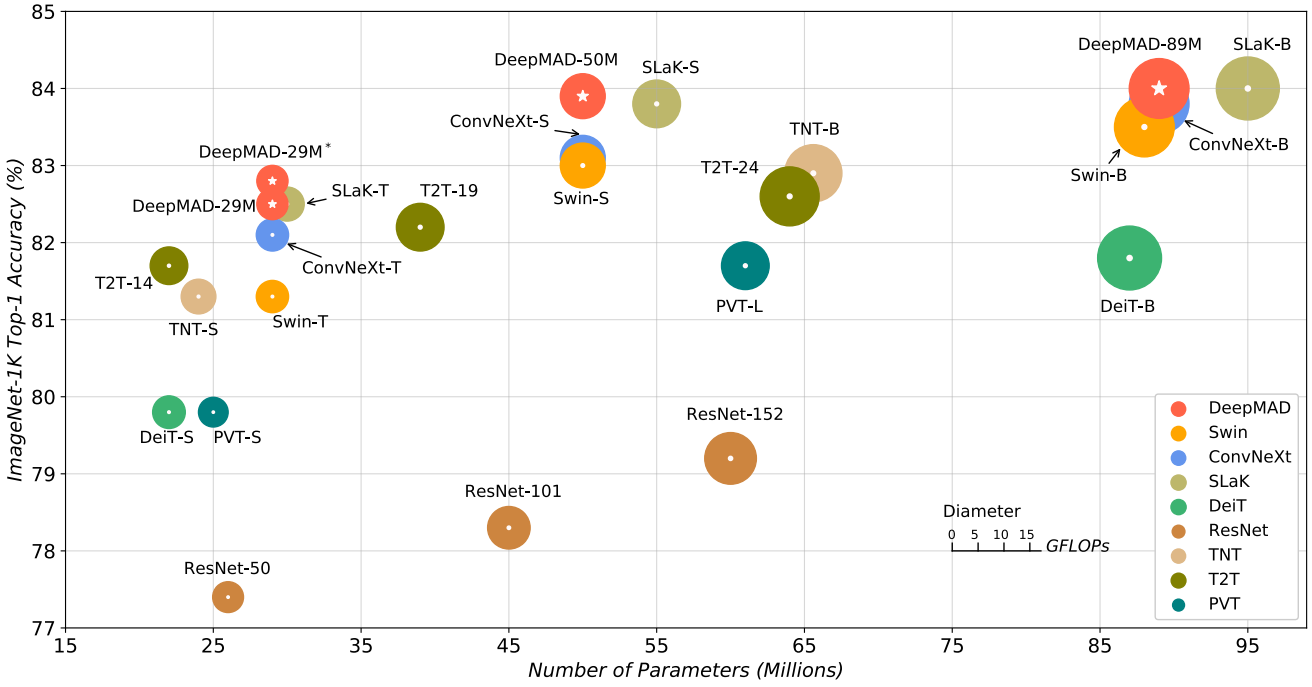

Figure 4. DeepMAD v.s. SOTA ViT and CNN models on ImageNet-1K. $\rho = 0.5$ for all DeepMAD models. All DeepMAD models except DeepMAD-29M* is trained with 224 resolution. x-axis is the Params, the smaller the better. y-axis is the accuracy, the larger the better.

B. Proof of Proposition 1

Assume there is an MLP model $f_A(\cdot)$ that has L-layers with different width w_i, and the entropy of the MLP model is H. To define the "average width" of $f_A(\cdot)$, we compare $f_A(\cdot)$ to a new MLP $f_B(\cdot)$. $f_B(\cdot)$ also has L-layers but with all layers sharing the same width $\bar{w}$. Suppose that the two networks have the same entropy for each output neuron, that is,

$$H_{f_a} = \sum_{i=1}^{L} \log(w_i), \quad H_{f_b} = L \cdot \log(\bar{w}). \tag{19}$$

When the above equality holds true(i.e., $H_{f_a} = H_{f_b}$), we can have the following equation,

$$\sum_{i=1}^{L} \log(w_i) = L \cdot \log(\bar{w}). \tag{20}$$

Therefore, we define $\bar{w}$ as the "average width" of $f_A(\cdot)$. Then we derive the definition of average width of MLP in Proposition 1 as following,

$$\bar{w} = \exp\left(\frac{1}{L}\sum_{i=1}^{L} \log w_i\right). \tag{21}$$

C. SOTA DeepMAD Models

We provide more SOTA DeepMAD models in Figure 4. Especially, Deep-MAD achieves better performance on small and base level. DeepMAD-50M can achieve 83.9% top-1 accuracy, which is even better than ConvNeXt-Base [41] with nearly only half of its scale. DeepMAD-89M achieves 84.0% top-1 accuracy at the "base" scale, outperforming ConvNeXt-Base and Swin-Base [40], and achieves similar accuracy with SLaK-Base [39] with less computation cost and smaller model size. On "tiny" scale, DeepMAD-29M also achieves 82.8% top-1 accuracy under 4.5G FLOPs and 29M Params. It is 1.5% higher than Swin-Tiny with the same scale, and is 2.2x reduction in Params and 3.3x reduction in FLOPs compared to T2T-24 [78] with 0.2% higher accuracy. Therefore, we can find that building networks only with convolutional blocks can achieve better performance than those networks built with vision transformer blocks, which shows the potentiality of the convolutional blocks.

D. DeepMAD Optimized for GPU Inference Throughput

We optimize GPU inference throughput using DeepMAD. To measure the throughput, we use float32 precision (FP32) and increase the batch size for each model until no more images can be loaded in one mini-batch inference. The throughput is tested on NVIDIA V100 GPU with 16 GB Memory. ResNet building block is used as our design space.

To align with the throughput of ResNet-50 and Swin-Tiny on GPU, we first use DeepMAD to design networks of different Params and FLOPs. Then we test throughput for all models. Among these models, we choose two models labeled as DeepMAD-R50-GPU and DeepMAD-ST-GPU such that the two models are aligned with ResNet-50 and Swin-Tiny respectively. The top-1 accuracy on ImageNet-1k are reported in Table 9.

Method	Res.	#Param.	Throughput	FLOPs	Acc.(%)
ResNet-50	224	26 M	1245	25.6 G	77.4
DeepMAD-R50-GPU	224	19 M	1171	3.0 G	80.0
Swin-Tiny	224	29 M	750	4.5 G	81.3
DeepMAD-ST-GPU	224	40 M	767	6.0 G	81.7

Table 9. DeepMAD models optimized for throughput on GPU. 'Res': image resolution.

E. Other Experiments Results

We fine-tune the *effectiveness* ρ in Table 10. Comparing to the results in the main text, we can achieve better accuracy when ρ is fine-tuned.

Method	ρ	Res.	#Param.	FLOPs	Acc.(%)
ResNet-18 [21]	0.01	224	11.7 M	1.8 G	70.9
DeepMAD-R18	0.3	224	11.7 M	1.8 G	77.7
DeepMAD-R18	0.15	224	11.7 M	1.8 G	**78.2**
ResNet-34 [21]	0.02	224	21.8 M	3.6 G	74.4
DeepMAD-R34	0.3	224	21.8 M	3.6 G	79.7
DeepMAD-R34	0.15	224	21.8 M	3.6 G	**80.3**
MobileNet-V2 [23]	0.9	224	3.5 M	320 M	72.0
DeepMAD-MB	0.5	224	3.5 M	320 M	72.3
DeepMAD-MB	1	224	3.5 M	320 M	**72.9**

Table 10. Fine-tuned ρ in DeepMAD. 'Res': image resolution.

F. Complexity Comparison with NAS Methods

DeepMAD is also compared with classical NAS works in complexity as well as accuracy on ImageNet-1K. The classical NAS methods [6, 38, 49, 75, 82] need to train a considerable number of networks and evaluate them the in searching phase, which is time-consuming and computing-consuming. DeepMAD need not train any model in the search phase, and it just needs to solve the MP problem to obtain optimized network architectures in a few minutes. As shown in Table 11, DeepMAD takes only a few minutes to search for a network that can achieve better accuracy (76.1%) than other NAS methods. It should be noted that Table 11 only consider the search time cost and does not consider the training time cost. However, DeepMAD only needs one training process to produce a high-accuracy model with trained weights, while these baseline methods train multiple times.

Method	#Param.	FLOPs	Acc.(%)	Search Cost (GPU hours)
NASNet-A [82]	5.3 M	564 M	74.0	48,000
ProxylessNAS [6]	5.8 M	595 M	76.0	200
PNAS [38]	5.1 M	588 M	74.2	5,400
SNAS [75]	4.3 M	522 M	72.7	36
AmoebaNet-A [49]	5.1 M	555 M	74.5	75,600
DeepMAD	5.3 M	390 M	76.1	< 1 (CPU hour)

Table 11. Complexity and accuracy comparison with NAS Methods on ImageNet-1K.

G. Discussion on Architectures

Figure 5 shows as *effectiveness* ρ increases, the depth of networks increases while the width decreases. As discussed in Section 5.3, the model accuracy does not always increase with ρ. When ρ is small, model accuracy increases as depth increases and width decreases. When ρ is large, the opposite phenomenon occurs. The existence of an optimal *effectiveness* means the existence of optimal depth and width of networks to reach the best accuracy.

The architectures of DeepMAD models are released along with the source codes. Compared to ResNet families, DeepMAD suggests deeper and thinner structures. The final stage of DeepMAD networks is also deeper. The width expansion after each downsampling layer is around 1.2-1.5, which smaller than 2 in ResNets.

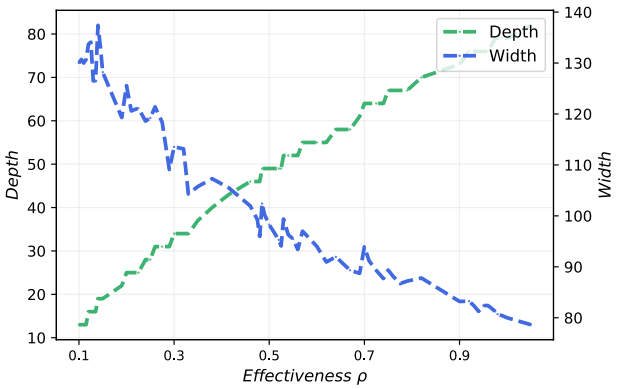

Figure 5. *Effectiveness* ρ v.s. the depth and width of each generated network on CIFAR-100. The architectures shown in this figure are same as those shown in Figure 2. The depth increases with ρ monotonically while the width decreases at the same time.

H. Experiment Settings

The detailed training hyper-parameters for CIFAR-100 and ImageNet-1K datasets are reported in Table 12.

Hyper-parameter	CIFAR-100	ImageNet-1K
warm-up epoch	5	20
cool-down epoch	0	10
epochs	1440	480
optimizer	SGD	SGD
batchnorm momentum	0.01	0.01
weight decay	5e-4	5e-5
nesterov	True	True
lr scheduler	cosine	cosine
label smoothing	0.1	0.1
mix up	0.2	0.8
cut mix	0	1.0
mixup switch prob	0.5	0.5
crop pct	0.875	0.95
reprob	0.5	0.2
auto augmentation	auto [14]	rand-m9-mstd0.5
lr	0.2	0.8
batch size	512	2048
amp	False	True

Table 12. Experiment Settings.