We proposed a new paradigm for network pruning: Train a CNN. We called it **SCAN**.

**Main Idea of TAS**

- Train a large CNN
- Prune filters, get a small CNN
- Fine-tune the CNN
- An efficient CNN

**Objective**

\[
L = - \log \left( \frac{\exp(x_y)}{\sum_{i=1}^{C} \exp(x_i)} \right) + \lambda_{cost} L_{cost}
\]

\[
L_{cost} = \begin{cases} 
\log(E_{cost}(A)) & F_{cost}(A) > (1 + t)R \\
0 & \text{otherwise} \\
-\log(E_{cost}(A)) & F_{cost}(A) < (1 + t)R 
\end{cases}
\]

- \(A\) is the set of parameters modeling the net config
- \(R\) is the target computational cost, e.g., 300M FLOPs
- \(E_{cost}(A)\) is the expectation of costs based on \(A\)
- \(F_{cost}(A)\) is the actual cost of the searched architecture

**Importance of TAS and KD**

- Pre-defined each layer prune 77% channels
- Random Search
- Pick the best from 10 random configurations
- TAS automatically search for the best configuration

- Use knowledge distillation
- Init use pre-trained weights
- KD use knowledge distillation

**Network Pruning via Transformable Architecture Search**

Xuanyi Dong\(^1\), Yi Yang\(^1\)

---

**Network Pruning**

Each convolutional layer is equipped with a learnable distribution for the size of the channels in this layer, indicated by \(p^t\) on the left side. The feature map for every layer is built sequentially by the layers, as shown on the right side. For a specific layer, \(K\) (2 in this example) feature maps of different sizes are sampled according to corresponding distribution and combined by channel-wise interpolation (CWI) and weighted sum. This aggregated feature map is fed as input to the next layer.

---

**Introduction**

Transformable Architecture Search (TAS): search for the best size of a network, i.e., the width and depth.

Traditional Neural Architecture Search (NAS): search for the topology structure of a network.

We proposed a new paradigm for network pruning: Train a CNN. The proposed paradigm is shown on the right side. The feature map for every layer is built sequentially by the layers, as shown on the right side. For a specific layer, \(K\) (2 in this example) feature maps of different sizes are sampled according to corresponding distribution and combined by channel-wise interpolation (CWI) and weighted sum. This aggregated feature map is fed as input to the next layer.

**Objective**

\[
L = - \log \left( \frac{\exp(x_y)}{\sum_{i=1}^{C} \exp(x_i)} \right) + \lambda_{cost} L_{cost}
\]

\[
L_{cost} = \begin{cases} 
\log(E_{cost}(A)) & F_{cost}(A) > (1 + t)R \\
0 & \text{otherwise} \\
-\log(E_{cost}(A)) & F_{cost}(A) < (1 + t)R 
\end{cases}
\]

- \(A\) is the set of parameters modeling the net config
- \(R\) is the target computational cost, e.g., 300M FLOPs
- \(E_{cost}(A)\) is the expectation of costs based on \(A\)
- \(F_{cost}(A)\) is the actual cost of the searched architecture

---

**Main Idea of TAS**

- Train a large CNN
- Prune filters, get a small CNN
- Fine-tune the CNN
- An efficient CNN

**Contributions**

1. A new pruning paradigm with SOTA performance.
2. A differentiable searching method for the network shape.