

NAS는 계산력 시간이 너무 걸리기 때문에 훈련과 search를 별개하여 수행한다.

## ⇒ Introduction.

App 배포시, model은 hardware에 효율적으로 배포되어야 하기 때문에 쉽지 않다.

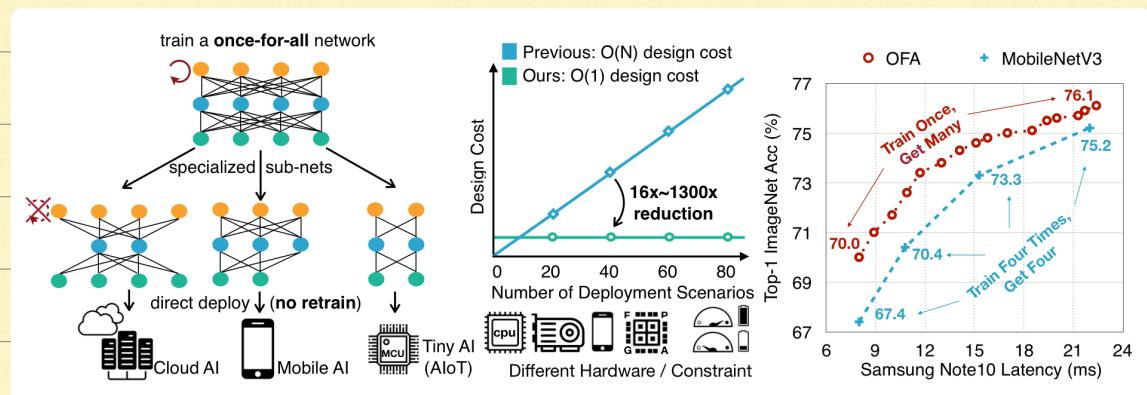
↳ hardware와深深 architecture가 달라짐

다양한 hardware의 다른 효율적인 압축이 NAS는 시간이 너무 오래 걸린다.

↳ 다양한 device의 적용을 수 있다.

따라서 다양한 network의 적용을 수 있는 once-for-all 기법

inference는 subnetwork의 실행모드, depth, resolution 등을 retrain 없이 조정할 수 있다.



## ⇒ model training 과 search를 decouple 한다

↳ training 시에는 모든 subnetwork의 성능도를 동시에 미리 측정

↳ subnet 선택 후마다 update

↳ 각각의 모든 subnet의 성능도를 미리 측정 후에는 update를 멈춰 (각 subnet의 성능은 초기)

↳ progressive shrinking algorithm 사용

↳ 가장 큰 net은 optim한 후 큰 net과 가중치를 공유하는 subnet은 차례대로 OFA를 fine-tune.

↳ 같은 subnet의 중요한 weight를 고려으로서 좋은 initialize를 제공해준다.

↳ 작은 subnet은 distill하는 데서 training efficient를 즐기한다

⇒ ImageNet으로 실험

## \* Related work

### - Efficient Deep Learning

SqueezeNet, MobileNet, ShuffleNet,

### - NAS

↳ NAS는 scalable하지 않다.

### - Dynamic Neural Networks

↳ 흐름적인 신경망을 위해, input image를 기반으로, model의 일부를 skip하는 방법

↳ 여러 조정 가능한 width multiplier로, runtime 시에 몇개의 net을 실행하는 방법도 있다.

논문의 방법은 다양하고 새로운 특수 신경망을 가능하다.

## \* Method

### - Problem formalization

$W_o$ 를 OFA의 weight,  $\text{arch}_i$ 를 architecture 구성이라고 가정한다.

$C(W_o, \text{arch}_i)$ 은 subnet을  $W_o$ 와  $\text{arch}_i$ 로 선택하는 방식이라고 할 때.

$$\min_{W_o} \sum_{\text{arch}_i} L_{\text{val}}(C(W_o, \text{arch}_i))$$

로 문제를 나타낼 수 있다.

전체적으로는 일반적인  $\text{arch}_i$ 의 훈련과 비슷한 정도로 subnet을 유지할 수 있도록  $W_o$ 를 optim 한다.

### - Architecture space

OFA는 several model이지만, depth, width, kernel, resolution 등 여러 많은 subnet을 사용한다.

CNN의 실행에 따라, feature map 각각은 채널 ch는 늘리는 Unit으로 나뉜다.

Unit은 feature map 크기가 같은데는 차례로 쪼개며 convolution stride가 1이다.

각 model은独立로 depth, width, kernel, resolution을 가질 수 있다.

### - Training the OFA network

#### - Naive approach

하나의 OFA를 학습시키는 것은 각 subnet이 다른 multi objective function 문제를 일으킬 수 있다.

naive한 접근은 모든 학습이 많은 subnet의 objective를 합쳐서 OFA를 update하는 것이다.

↳ subnet이 늘어남에 따라 cost가 극단적으로 늘어나기 때문에 안된다.

다른 접근은 update마다 몇개의 subnet을 뽑는 방법이다.

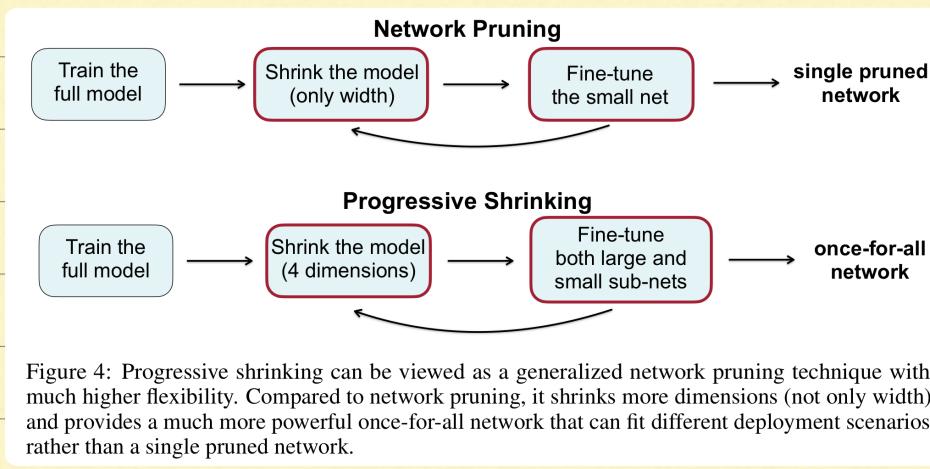
↳ accuracy가 떨어짐

### - Progressive shrinking

OFA는 작은 subnet이 큰 subnet에 합쳐진 형식의 다른 여러 subnet을 일으킨다.

subnetwork 간의 관계를 병기하기 위해, large net with small net으로 절전적 효율을 갖는다.

#### ↳ progressive shrinking



먼저 가장 큰 MV ( $k=7, D=4, W=6$ )을 훈련시킨다.

그 다음, network가 다수의 net을 지원할 수 있도록 sampling space에 추가하여 fine-tuning 한다.

↳ Del w/o 최대값인 대로 우선 elastic k를 지원하고 그 다음 Del W를 지원한다.

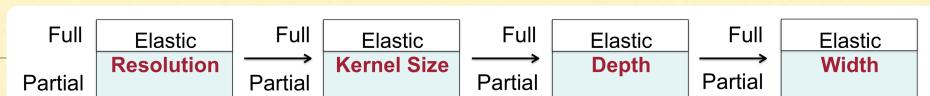


Figure 3: Illustration of the progressive shrinking process to support different depth  $D$ , width  $W$ , kernel size  $K$  and resolution  $R$ . It leads to a large space comprising diverse sub-networks ( $> 10^{19}$ ).

Resolution은 training을 batch에서 다른 해상도의 이미지를 sampling으로 한다.

전부 훈련하기 elastic sha.

또한 가장 큰 MV를 훈련시킨 후 distill을 수행한다.

↳ large MV의 soft label과 GT를 모두 사용하여 loss 계산

PS 방식은 small subnet을 위한 OFA를 fine-tuning 한다.

이미 훈련이 완료된 large subnet의 영향이 안가져온다.

small network은 large subnet과 가중치를 공유하기 때문에, small subnet은 잘 학습된다

large subnet으로 initialize 하면 학습을 계속할 수 있다.

Fig 4의 network pruning과 비교하여, width 뿐만 아니라, w, d, r, k 모두 가능

또한 BE subnet의 fine-tune 가능

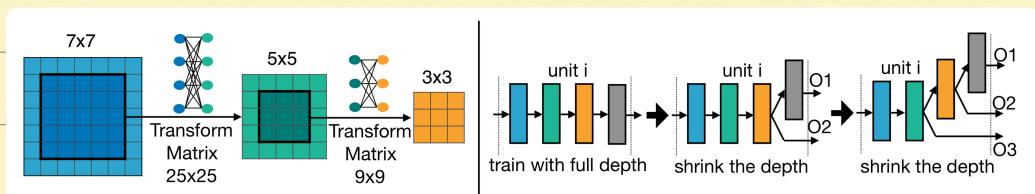


Figure 5: Left: Kernel transformation matrix for elastic kernel size. Right: Progressive shrinking for elastic depth. Instead of skipping each layer independently, we keep the first  $D$  layers and skip the last  $(4 - D)$  layers. The weights of the early layers are shared.

### - Elastic kernel size

7x7 kernel의 중앙 5x5, 3x3을 사용함으로써, elastic kernel을 만들수 있다.

이2인 모든 subnet의 같은 kernel을 사용하면, 성능 차이가 일어난다.

kernel transformation matrix를 weight sharing에 사용한다

↳ layer의 다른 matrix A는

같은 layer의 다른 channel 간 공유되는 경우  $15 \times 15 + 9 \times 9 = 108$ 개의 total parameter 필요하다

### - Elastic depth (layer 수)

N개를 만들 때는 보통 (weight sharing) 대로, 전체 skip하는 형식

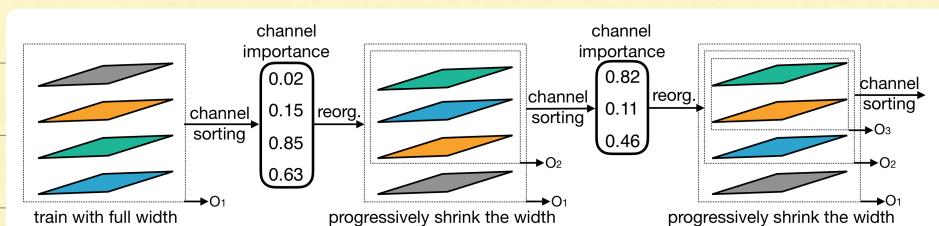


Figure 6: Progressive shrinking for elastic width. In this example, we progressively support 4, 3, and 2 channel settings. We perform channel sorting and pick the most important channels (with large L1 norm) to initialize the smaller channel settings. The important channels' weights are shared.

### - Elastic width. (channel size)

channel을 다양한 확장 크기로 설정하는데 있도록 허용

예) full-size를 학습, 중요한 channel 순으로 sorting 후 학습

↳ small net은 초기화를 잘 되는 것.

### - Specialized model deployment with Once-for-all network.

OFA 훈련 후 다음 단계는 hardware에 맞는 model을 찾는 것

↳ NN twin 이미지.

↳ arch, acc table 만들.

\* Experiments.