Deep Learning – Parte 9

Técnicas de Busca da Melhor Arquitetura em Redes Neurais Profundas

Índice Geral

1 Introdução

- Em vários momentos deste curso, nos concentramos em técnicas de regularização, que visam maximizar a capacidade de generalização do modelo de aprendizado, etapa ainda mais decisiva no caso de redes neurais profundas. Mas, para tanto, sempre foi suposta a disponibilidade de um modelo de aprendizado previamente definido, ou então foi atribuída ao projetista a tarefa de buscar manualmente a estrutura do modelo de aprendizado, para cada frente de aplicação pretendida.
- São muito raros os casos em que uma arquitetura ótima de rede neural profunda está disponível previamente, dadas as demandas de uma aplicação. Logo, até este momento do curso, a solução recaía sobre a busca manual. Buscar manualmente a estrutura do modelo de aprendizado significa definir manualmente seus hiperparâmetros. Não estamos falando aqui dos hiperparâmetros do algoritmo de otimização que vai realizar o ajuste de pesos. O foco está "apenas" nos hiperparâmetros que definem a arquitetura da rede neural profunda.

- O teste manual por tentativa-e-erro é uma maneira tradicional e ainda predominante de ajuste de hiperparâmetros, embora o sucesso desse processo de tentativa-e-erro exija uma compreensão ampla das demandas da aplicação e de como cada configuração de hiperparâmetros poderia contribuir ou não para o seu atendimento.
- Vale salientar que uma busca manual é capaz de explorar poucas possibilidades de estruturas de modelos de aprendizado, quando comparada com técnicas automáticas que recorrem à força bruta da máquina.
- Sendo assim, o ajuste manual é ineficaz para muitos problemas práticos devido a fatores como: (1) presença de um grande número de hiperparâmetros em modelos de aprendizado muito complexos, que é o caso de arquiteturas profundas; (2) alto custo computacional da avaliação de modelos candidatos; e (3) presença de interações não-lineares e multivariadas entre os hiperparâmetros, dificultando a tomada de decisão.

- Esses e outros fatores fomentaram um aumento da pesquisa envolvendo técnicas para otimização automática de hiperparâmetros (LUO, 2016), também denominada NAS (do inglês *Network Architecture Search*), sendo um ramo do AutoML (HUTTER et al., 2019). Recomenda-se a leitura dos seguintes *surveys* sobre NAS:
  - ✓ WISTUBA et al. (2019)
  - ✓ ELSKEN et al. (2019)
  - ✓ REN et al. (2020)
- Fica evidente, assim, que são muito bem-vindas técnicas apropriadas de ajuste automático de hiperparâmetros estruturais do modelo de aprendizado.
- Repare que, no contexto de arquiteturas de redes neurais profundas, há uma explosão de possibilidades de estruturas de modelos de aprendizado, fazendo com que mesmo a busca automática fundamentada na força bruta da máquina se torne inviável ou de alto custo. Sendo assim, o AutoML, que até pode levar a modelos de aprendizado parcimoniosos, se torna BigML quando executa o NAS.

- Logo, determinar automaticamente uma estrutura (também chamada de arquitetura da rede neural) otimamente flexível, e capaz de atender as funcionalidades demandadas por uma aplicação específica, com ótima capacidade de generalização, envolve uma definição apropriada do espaço de hipóteses e sua exploração eficiente (YANG & SHAMI, 2020).
- Para se chegar a bons resultados junto a diferentes problemas de aplicação, o ajuste automático de uma rede neural profunda vai requerer a sintonia adequada de hiperparâmetros tais como número e tipo de camadas, número e tipo de neurônios por camada, presença ou não de conexões diretas entre camadas não-contíguas, presença ou não de conexões recorrentes, etc.
- Restrito ao contexto de redes neurais convolucionais, abordaremos aqui quatro dimensões de hiperparâmetros: resolução, tamanho do *kernel*, profundidade e largura.

2 Ajuste automático de hiperparâmetros na busca de arquiteturas otimizadas

- Técnicas de ajuste automático de hiperparâmetros estruturais, definidores da arquitetura ótima de redes neurais profundas, devem ser buscadas pelas seguintes razões, sendo que parte delas já foi mencionada na seção introdutória:
  - ✓ Liberação do especialista humano, que deixa de ser responsável por especificar a priori esses hiperparâmetros, geralmente por processos manuais de tentativa-e-erro. A máquina passa a ser responsável pela sua definição em tempo de treinamento e de forma automática.
  - ✓ Necessidade do atendimento de requisitos de *hardware*, como tempo de latência e uso de memória, na operação das máquinas de aprendizado implementadas. É possível buscar arquiteturas dedicadas a cada *hardware* específico e que otimizam simultaneamente outros objetivos do projeto, que muitas vezes são conflitantes entre si.

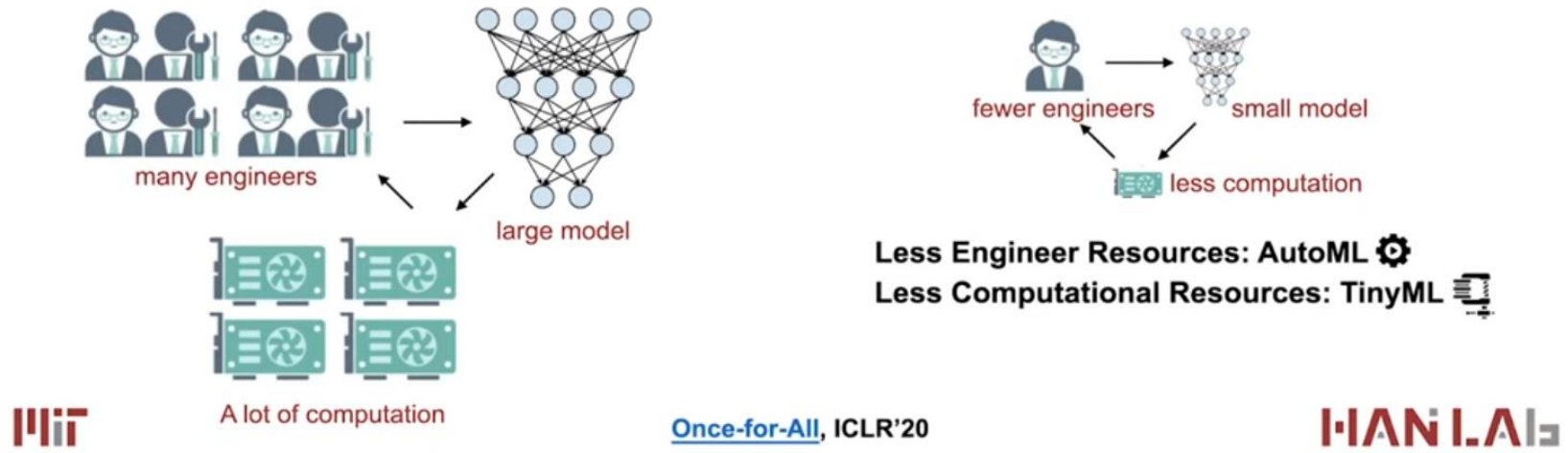
- ✓ Potencial de ganho de desempenho, pois uma arquitetura flexível "na medida certa" pode contribuir com a capacidade de generalização. Além disso, se o resultado final deste ajuste automático de hiperparâmetros levar a um modelo de aprendizado mais parcimonioso, com menos pesos sinápticos, o aprendizado tende a ser menos desafiador para os algoritmos de otimização não-linear, por exemplo, com menos problemas de mínimos locais ruins.

- De fato, selecionar a melhor configuração de hiperparâmetros estruturais para modelos de aprendizado de máquina pode promover ganho de desempenho em tarefas variadas de aprendizado de máquina (HUTTER et al., 2019), inclusive por identificar soluções dedicadas a hardwares distintos, como dispositivos móveis, CPUs, GPUs e FPGAs.
- Não faltam motivações, portanto, para que uma configuração ótima, o mais enxuta possível e de alto desempenho, deva ser buscada para a máquina de aprendizado, no caso de interesse aqui, uma rede neural profunda.

- A competência da busca tende a ser maior quanto menor for a dimensão do espaço de hipóteses, sendo que esta dimensão é atrelada ao número de hiperparâmetros a serem especificados. É comum, como já mencionado, impor um certo alcance máximo para o espaço de hipóteses, visando tornar a busca factível e sem comprometer o desempenho global da máquina de aprendizado.
- Mesmo com essa iniciativa de restringir o alcance do espaço de hipóteses, em aprendizado profundo é sabido que a construção de um modelo de aprendizado de máquina com estrutura ótima é um processo complexo e que pode consumir uma quantidade absurda de recursos computacionais, pois para cada proposta de hiperparâmetros associados à sua estrutura interna, especificando um ponto no espaço de hipóteses, é preciso, em princípio, realizar o ajuste de parâmetros livres do respectivo modelo de aprendizado.
- Repare que cada ponto do espaço de hipóteses corresponde a uma estrutura com parâmetros livres, candidata a exercer o papel de modelo de aprendizado.

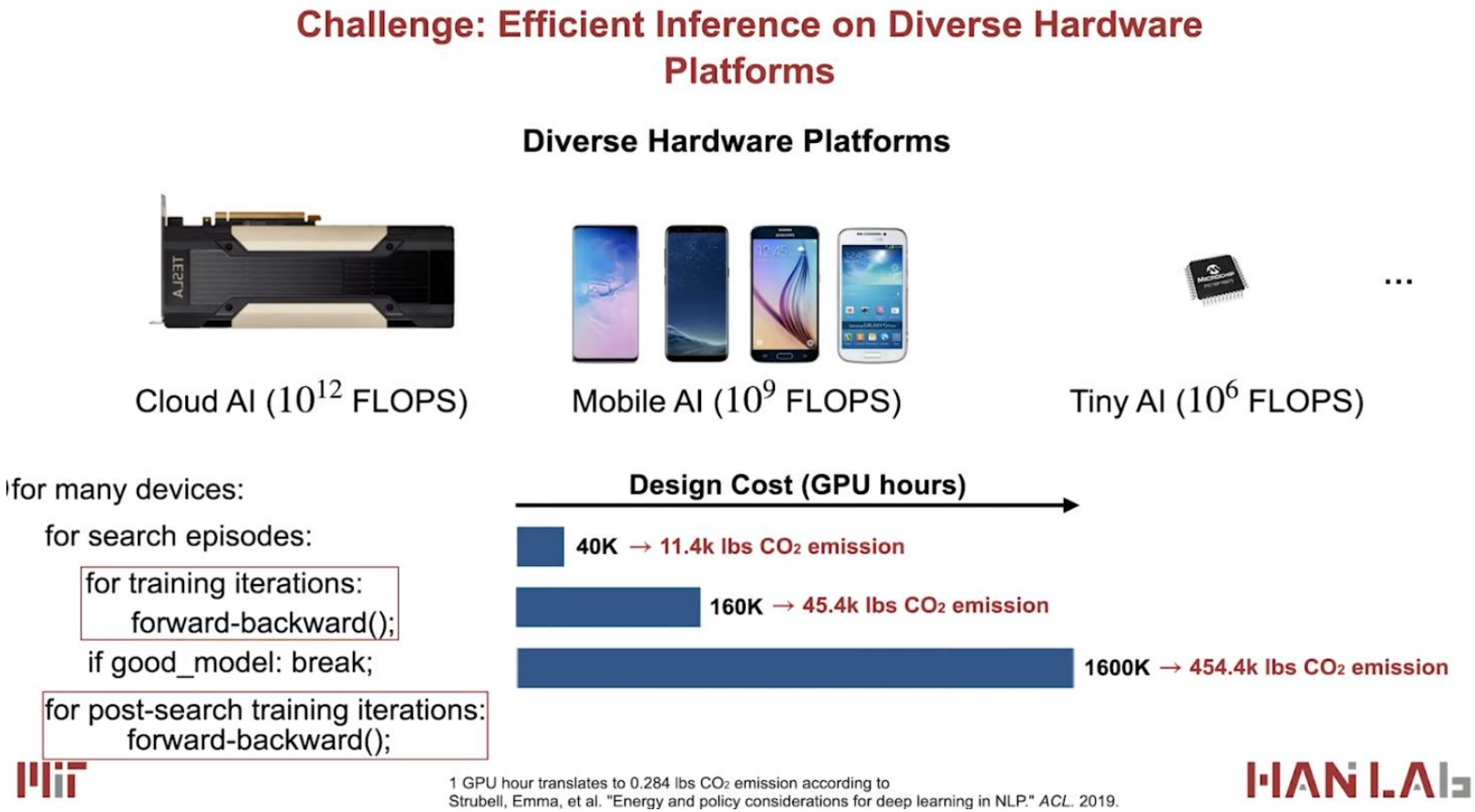
- Já existem algoritmos de otimização bem consolidados para o ajuste de parâmetros livres de modelos de aprendizado, tendo sido abordados em vários momentos deste curso. Mesmo bem consolidados e eficientes, em cenários de arquiteturas profundas, a demanda computacional requerida por esses algoritmos de ajuste de parâmetros livres, para cada proposta individual de modelo de aprendizado, pode ser bem elevada.
- Isso implica que a busca automática por uma estrutura ótima para o modelo de aprendizado, a partir da definição ótima de seus hiperparâmetros, se configura como um enorme desafio computacional, embora progressos expressivos tenham sido verificados na literatura recente (SHAWI et al., 2019).
- As várias propostas recentemente apresentadas na literatura, sendo que parte delas será alvo de discussão neste tópico do curso, têm diferentes vantagens e desvantagens quando aplicadas a diferentes tipos de problemas (YANG & SHAMI, 2020; ZÖLLER & HUBER, 2019), mas é certo que elas contribuem para promover:

1. Uma redução acentuada do esforço humano envolvido, particularmente em cenários que envolvem uma explosão combinatória de soluções candidatas e de decisões de projeto, como é o caso do aprendizado profundo;
2. Uma redução acentuada da demanda por expertise humana, pois a máquina assume grande parte das decisões que eram tomadas pelo projetista humano;
3. Uma maior confiança de que o potencial de um certo tipo de modelo de aprendizado está sendo explorado devidamente;
4. Uma maior reprodutibilidade das pesquisas, pois apenas quando um investimento compatível no nível de ajuste de hiperparâmetros estruturais é implementado é que se pode comparar de maneira consistente diferentes técnicas de aprendizado de máquina. Em outras palavras, o emprego de um mesmo método de sintonia automática de hiperparâmetros em diferentes algoritmos de aprendizado de máquina robustece a etapa de determinação do melhor modelo de aprendizado para uma tarefa específica.



Fonte: https://ofa.mit.edu

- A figura acima, de autoria de Cai et al. (2020), autores da técnica NAS denominada Once-for-All, ilustra essa iniciativa de buscar uma IA verde, algo premente com base nos números que serão apresentados a seguir.
- Cabe evidenciar que a tendência de uso de menos projetistas humanos e obtenção de modelos de aprendizado estruturalmente mais parcimoniosos, possivelmente ordens de magnitude mais enxutos que modelos já tidos como estado-da-arte, pode levar a um consumo de recursos computacionais inviável na fase de treinamento.



Fonte: CAI et al. (2020) (https://ofa.mit.edu)

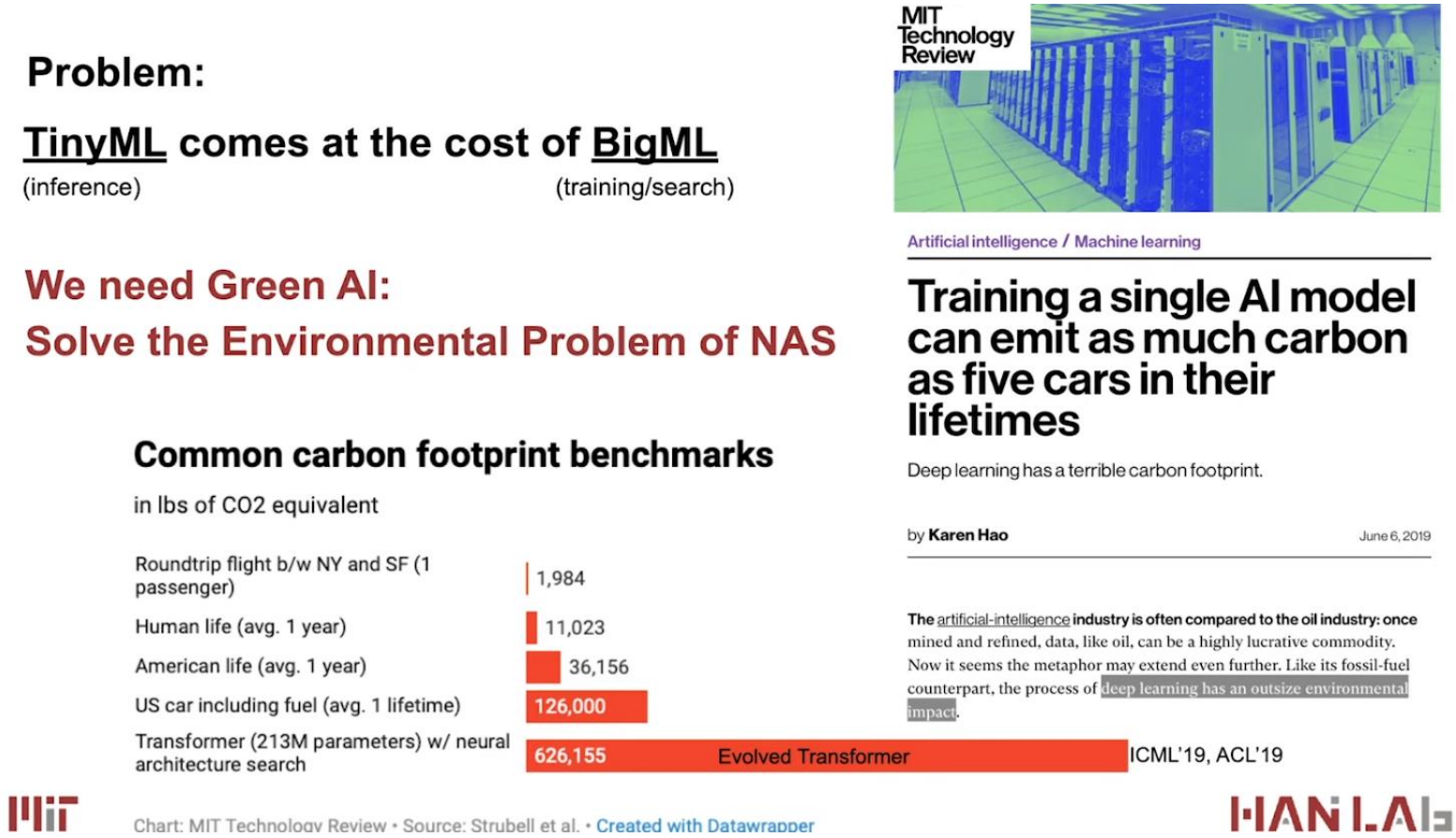
Problem:

TinyML comes at the cost of BigML

(inference) (training/search)

We need Green AI:

Solve the Environmental Problem of NAS



Artificial intelligence / Machine learning

Training a single AI model can emit as much carbon as five cars in their lifetimes

Deep learning has a terrible carbon footprint.

by **Karen Hao** June 6, 2019

The artificial-intelligence industry is often compared to the oil industry: once mined and refined, data, like oil, can be a highly lucrative commodity. Now it seems the metaphor may extend even further. Like its fossil-fuel counterpart, the process of deep learning has an outsize environmental impact.

Fonte: CAI et al. (2020) (https://ofa.mit.edu)

- Ao longo das próximas seções, após apresentar técnicas de busca mais simples (efetivas em cenários com poucos hiperparâmetros, mas que escalam mal para cenários com muitos hiperparâmetros), iremos nos concentrar em três contribuições de destaque na literatura, seja pelo desempenho diferenciado, seja pela escalabilidade, seja pelo caráter inovador da referida proposta.
- Haverá, assim, condições técnicas para desfazermos a maldição "*TinyML comes at the cost of BigML*": Os potenciais danos causados por uma certa tecnologia relevante se combatem, por pessoas de bem, com mais tecnologia."

3 Busca em grade, busca aleatória e BOA

- As técnicas de otimização de hiperparâmetros têm que lidar com a natureza não-convexa e não-diferenciável da busca, sendo sujeitas a mínimos locais. Além de detectar hiperparâmetros contínuos, muitos dos algoritmos para sintonia automática de hiperparâmetros têm a capacidade de identificar com eficácia hiperparâmetros discretos, categóricos e condicionais.

- Visando selecionar uma combinação de hiperparâmetros de melhor desempenho, busca em grade (do inglês *grid search*) (BERGSTRA et al., 2011) é uma abordagem exaustiva que determina a configuração ótima em um domínio discreto e previamente fixado de hiperparâmetros. No entanto, a busca em grade não escala bem com o crescimento do número de hiperparâmetros a determinar.

- Já a busca aleatória (do inglês *random search*) (BERGSTRA & BENGIO, 2012) seleciona aleatoriamente combinações de hiperparâmetros no espaço de busca, adotando como critério de terminação o tempo de execução ou um limiar de recursos utilizados.

- Em ambas as buscas, cada configuração de hiperparâmetros é tratada de forma independente. Ao contrário desta suposta independência, algoritmos de otimização bayesiana (BOA, do inglês *Bayesian Optimization Algorithm*) (EGGENSPERGER et al., 2013) determinam a próxima configuração de hiperparâmetros com base num histórico de desempenho de configurações anteriormente testadas, o que evita muitas avaliações desnecessárias de modelos candidatos.

- Logo, a otimização bayesiana pode detectar a combinação ótima de hiperparâmetros em menos iterações do que é necessário para as buscas em grade e aleatória. Para ser aplicada a diferentes problemas, a otimização bayesiana precisa modelar as relações de dependência entre os hiperparâmetros, usando, por exemplo, grafos probabilísticos (KOLLER & FRIEDMAN, 2009). No entanto, uma vez que a otimização bayesiana tem, ao menos em parte do seu fluxo de informação, uma natureza sequencial na exploração do espaço de busca, é menos efetiva a adoção de iniciativas que visem paralelizar a busca.
- Em anos recentes, surgiu um número elevado de propostas para NAS com melhores resultados no contexto de aprendizado profundo (ELSKEN et al., 2019; WISTUBA et al., 2019; REN et al., 2020). Essas propostas agregam várias contribuições isoladas que passam a operar em sinergia e adaptam técnicas poderosas, como gradiente descendente, visando ganhos de escalabilidade na otimização da arquitetura de uma rede neural profunda.

4 EfficientNet

- A EfficientNet (TAN & LE, 2020) é uma proposta de NAS (ou AutoML) para um certo padrão de redes convolucionais que parte de duas hipóteses:
  - ✓ Ao lidar com problemas de classificação desafiadores e havendo recursos computacionais limitados, é necessário escalar adequadamente as dimensões envolvidas no projeto da arquitetura da rede neural convolucional. Nesses cenários de limitação de recursos, quanto maior a dimensão que a rede neural convolucional puder atingir, maior tende a ser o seu desempenho.
  - ✓ Ao escalar as dimensões envolvidas no projeto da arquitetura da rede neural convolucional, é necessário levar em conta a existência de uma interdependência entre três classes de hiperparâmetros estruturais do modelo de aprendizado: largura, profundidade e resolução.

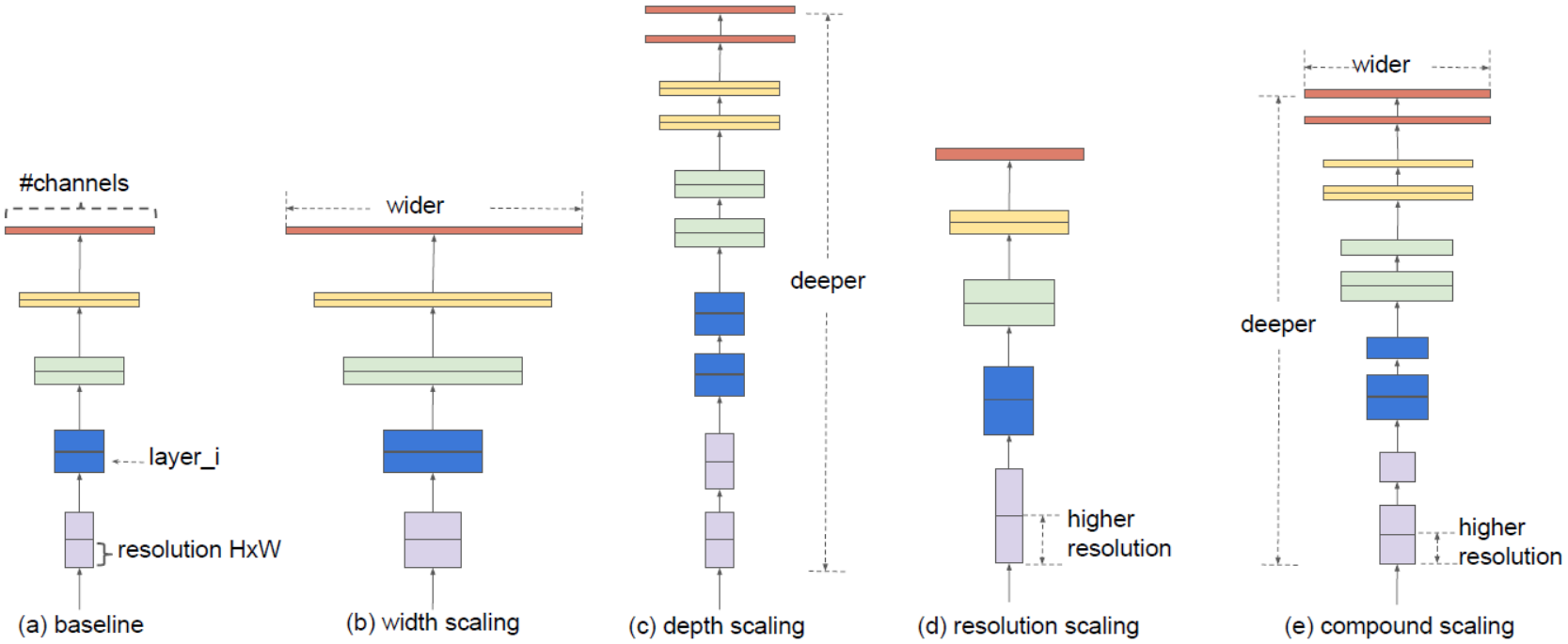


Figura 1 – Três classes de hiperparâmetros em redes convolucionais

Fonte: TAN & LE (2020)

- Partindo de uma arquitetura-base, denominada EfficientNet-B0, a principal contribuição do estudo é a proposição de uma família de modelos fundamentados em um método de escalamento que emprega uma regra empírica de relacionamento entre esses três hiperparâmetros estruturais, produzindo:

$$\begin{aligned} \text{depth: } & d = \alpha^{\phi} \\ \text{width: } & w = \beta^{\phi} \\ \text{resolution: } & r = \gamma^{\phi} \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned}$$

onde ϕ é um coeficiente de escalamento e α, β e γ são constantes que podem ser determinadas por uma busca em grade.

- Intuitivamente, o coeficiente ϕ vai especificar o montante de recursos computacionais a ser alocado ao modelo, sendo definido pelo projetista, e α, β e γ são responsáveis por distribuir esses recursos entre as três dimensões estruturais, que são profundidade, largura e resolução.
- Repare que o número de operações em ponto flutuante (FLOPS) demandadas por uma rede neural convolucional cresce com d, w^2 e r^2, dobrando com o dobro da profundidade e quadruplicando com o dobro da largura ou da resolução.

- Como as operações de convolução dominam o custo computacional em redes convolucionais, então, com base na restrição imposta:

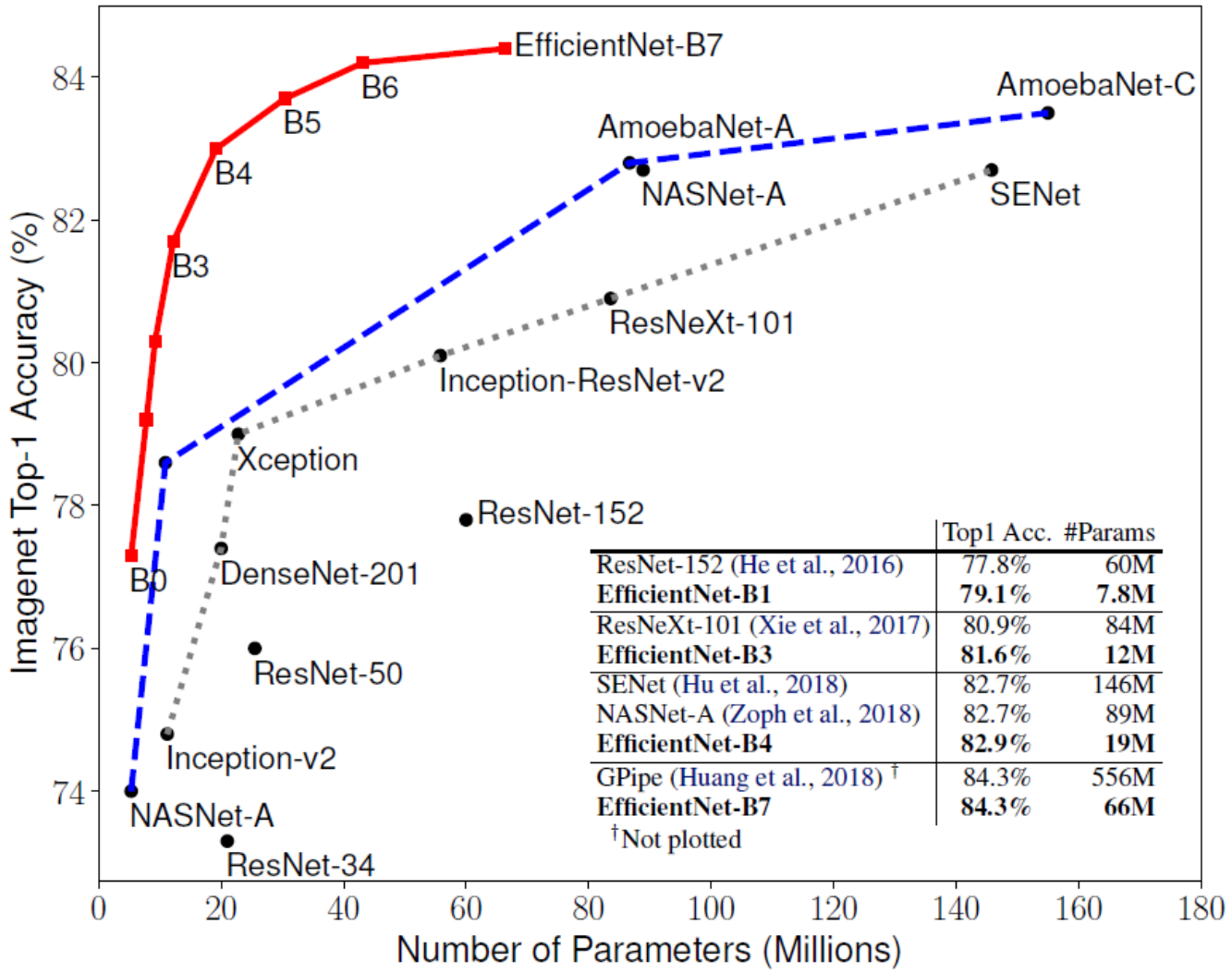
$$\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2,$$

o número de FLOPS tende a variar na forma:

$$\text{FLOPS} = \left(\alpha \cdot \beta^2 \cdot \gamma^2\right)^{\phi},$$

o que implica que os recursos computacionais requeridos pela arquitetura de rede neural convolucional variam aproximadamente com 2^{ϕ}.

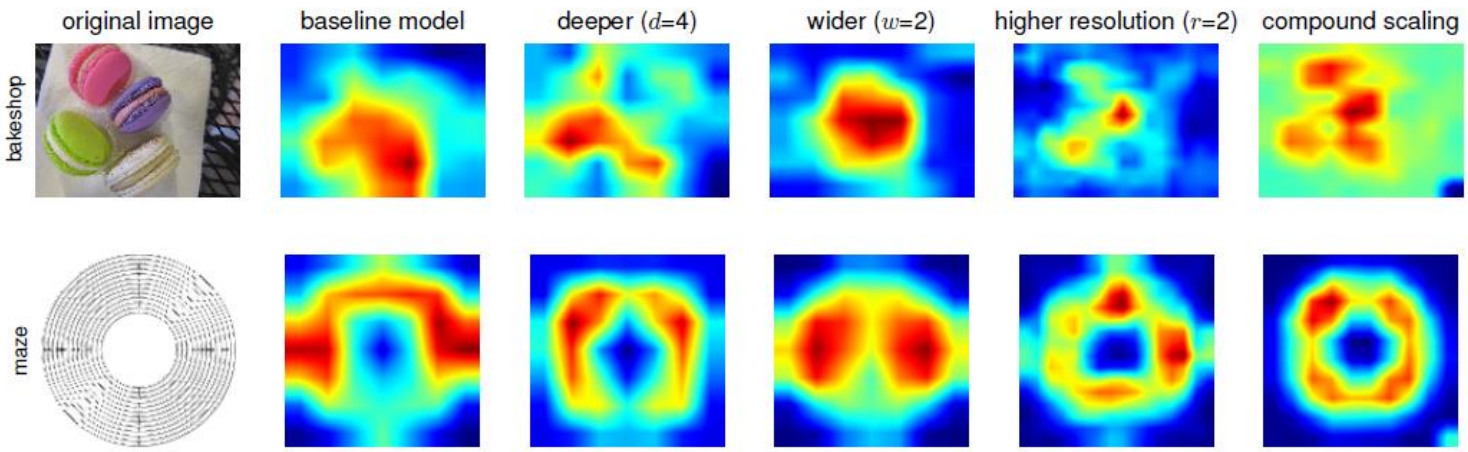
- O custo-benefício desta proposta se mostrou muito positivo, levando a resultados impressionantes junto a vários problemas de classificação desafiadores. Apresentamos a seguir resultados para a base ImageNet, extraídos da proposta original em TAN & LE (2020). Além disso, resultados experimentais adicionais apontam também para a relevância de escalar proporcionalmente, portanto simultaneamente, nas três dimensões de hiperparâmetros estruturais.



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.1%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.6%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.9%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.3%**	**66M**

†Not plotted

Fonte: TAN & LE (2020)



Model	FLOPS	Top-1 Acc.
Baseline model (EfficientNet-B0)	0.4B	77.3%
Scale model by depth (d=4)	1.8B	79.0%
Scale model by width (w=2)	1.8B	78.9%
Scale model by resolution (r=2)	1.9B	79.1%
Compound Scale (d=1.4, w=1.2, r=1.3)	**1.8B**	**81.1%**

Fonte: TAN & LE (2020)

- Favor consultar TAN & LE (2020) para entender como foi concebida a *baseline network*. A partir disso, a estratégia para se chegar às demais arquiteturas da família é a seguinte:

Starting from the baseline EfficientNet-B0, we apply our compound scaling method to scale it up with two steps:

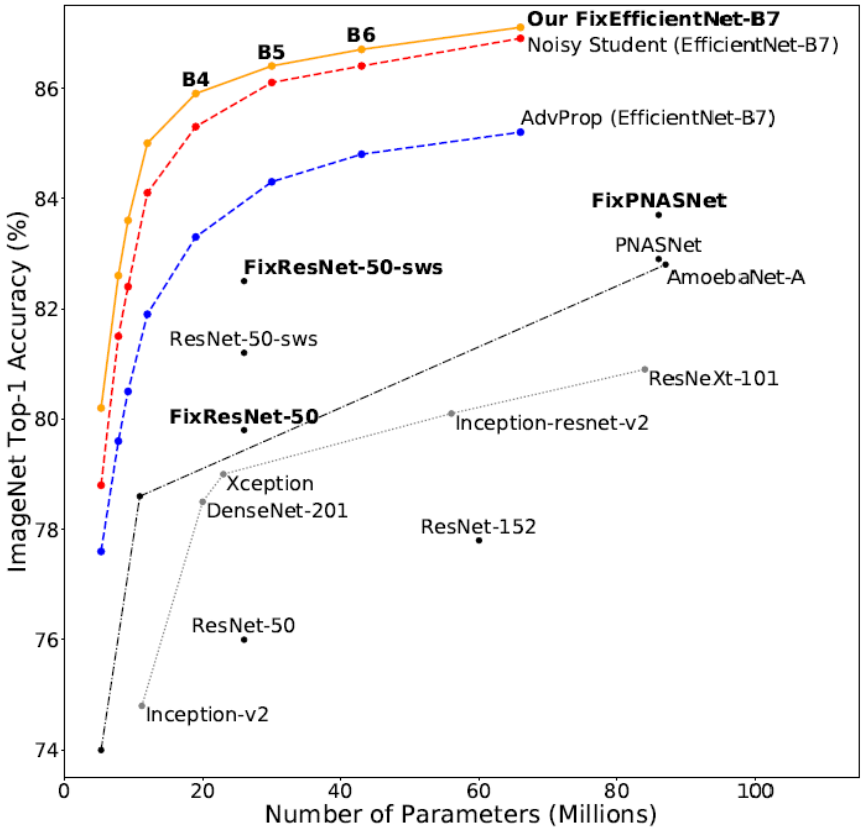
- STEP 1: we first fix $\phi = 1$, assuming twice more resources available, and do a small grid search of α, β, γ based on Equation 2 and 3. In particular, we find the best values for EfficientNet-B0 are $\alpha = 1.2, \beta = 1.1, \gamma = 1.15$, under constraint of $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$.
- STEP 2: we then fix α, β, γ as constants and scale up baseline network with different ϕ using Equation 3, to obtain EfficientNet-B1 to B7 (Details in Table 2).

Table 1. **EfficientNet-B0 baseline network** – Each row describes a stage i with $\hat{L}_i$ layers, with input resolution $\langle \hat{H}_i, \hat{W}_i \rangle$ and output channels $\hat{C}_i$. Notations are adopted from equation 2.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Fonte: TAN & LE (2020)

- Recentemente, houve a incorporação de uma técnica capaz de incrementar o desempenho de muitas redes convolucionais em tarefas de classificação de imagens, conhecida por "*Fixing Resolution*" (TOUVRON et al., 2019). Esta nova proposta de NAS foi denominada FixEfficientNet (TOUVRON et al., 2020).



Fonte: TOUVRON et al. (2020)

5 DARTS

- Em lugar de partir de uma *baseline network*, como feito no caso da EfficientNet e suas variações, boa parte das abordagens convencionais em NAS, que não serão revistas aqui (para tanto, recorra aos *surveys* apresentados na seção introdutória), recorre a técnicas de aprendizado por reforço (maximizando o retorno acumulado após uma sequência de decisões de projeto da estrutura do modelo de aprendizado) ou então a meta-heurísticas de busca, as quais serão abordadas na última seção deste tópico do curso.
- Ambas as abordagens interpretam o espaço de hipóteses (espaço de busca) como sendo um espaço discreto, que não admite o emprego das poderosas técnicas de gradiente descendente, tão eficazes na etapa de ajuste dos parâmetros livres (pesos sinápticos) das redes neurais profundas. De fato, a inclusão ou não de um neurônio, de uma camada e até o ajuste das dimensões de um filtro convolucional promovem variações em degrau na superfície de erro, que perde a continuidade.

- Nesta seção, por outro lado, vamos investigar uma técnica que propõe uma relaxação contínua da representação da arquitetura de uma rede neural profunda, permitindo a busca eficiente da arquitetura usando gradiente descendente. A relaxação contínua na representação dos pontos do espaço de hipóteses torna a superfície de erro contínua, viabilizando o cálculo do vetor gradiente em qualquer ponto.
- Já sabemos que buscas iterativas que não recorrem ao gradiente (como aquelas vinculadas a meta-heurísticas evolutivas) são buscas de ordem zero. Aquelas que recorrem ao gradiente são buscas de ordem um e aquelas que recorrem ao gradiente e à matriz hessiana são buscas de ordem dois. Quanto maior a ordem, mais informação é utilizada para guiar a busca, embora o custo computacional por iteração se amplie muito.
- Se soubermos fazer um bom uso desta informação adicional, o progresso a cada iteração pode ser mais intenso, possivelmente compensando com folga o aumento

do custo computacional por iteração. Exemplo: um algoritmo de busca que custa (em termos de recursos computacionais requeridos, processamento e memória) 10 vezes mais que um outro por iteração, mas que requer 200 vezes menos iterações para convergir, é 20 vezes mais rápido na solução do problema de otimização.

- Portanto, no contexto de NAS, ao substituir as convencionais buscas de ordem zero por uma busca baseada em gradiente descendente, técnica muito difundida e que dispõe de algoritmos competentes para a sua implementação no contexto de aprendizado profundo, espera-se verificar este tipo de ganho de desempenho.
- Os pesos sinápticos continuam a ser ajustados por gradiente descendente, para cada arquitetura proposta.
- Esta é a ideia que está por trás do DARTS (do inglês *Differentiable ARchiTecture Search*) (LIU et al., 2019) e de outras técnicas para NAS que estão baseadas em gradiente. Como agora passam a existir duas buscas baseadas em gradiente, geralmente se adota o procedimento ilustrado na Figura 2.

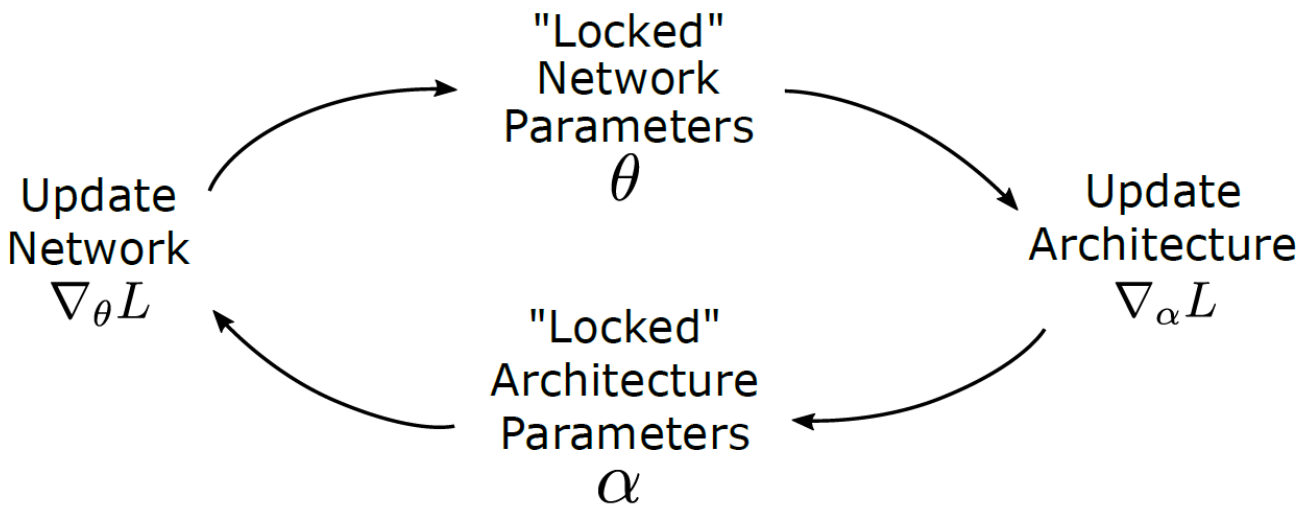


Figura 2 – Estratégia de otimização alternada (binível) para NAS baseada em gradiente

Fonte: GREEN et al. (2019)

- DARTS pode ser aplicado tanto para redes convolucionais como para redes recorrentes e realiza a NAS ordens de magnitude mais rápido que propostas de busca de ordem zero para NAS (LIU et al., 2019).
- Trata-se de uma grande conquista, pois a obtenção de arquiteturas estado-da-arte para os problemas de classificação de imagens CIFAR-10 e ImageNet requeriam 2000 dias de GPU para NAS empregando aprendizado por reforço (ZOPH et al.,

2018) ou 3150 dias de GPU para NAS empregando algoritmo evolutivo (REAL et al., 2018). Restrições que limitam o alcance do espaço de hipóteses podem ser impostas visando uma redução neste custo da busca, mas o problema de escalabilidade não era atenuado o suficiente, pois o número de arquiteturas de redes neurais profundas candidatas que precisavam ser treinadas continuava muito elevado.

- A técnica DARTS é capaz de aprender blocos construtivos de arquiteturas de alto desempenho com base em topologias de grafos que representam um amplo espaço de hipóteses para as arquiteturas candidatas. Além disso, como já antecipado, DARTS não se restringe a nenhuma família de arquitetura específica e é aplicável a redes convolucionais e recorrentes.
- Também foi verificado, em experimentos práticos, que as arquiteturas resultantes da DARTS são transferíveis entre aplicações. Por exemplo, a arquitetura encontrada para a base CIFAR10 apresentou bom desempenho para a ImageNet.

- DARTS visa definir uma célula de computação como um bloco construtivo da arquitetura final. A célula aprendida pode ser empilhada para formar uma rede convolucional, ou conectada recursivamente para formar uma rede recorrente.
- Uma célula é um grafo direcionado acíclico composto por uma sequência ordenada de *N* nós. Cada nó $x^{(i)}$ é uma representação latente (por exemplo, um mapa de ativação em redes convolucionais) e cada aresta (i, j) é associada a alguma operação $o^{(i,j)}$ que transforma $x^{(i)}$. É considerado ainda que cada célula tem dois nós de entrada e um nó de saída. No caso de células convolucionais, os dois nós de entrada correspondem às saídas das células das duas camadas antecessoras. No caso de células recorrentes, as duas entradas são a entrada da célula no instante atual e seu estado interno atualizado no último passo. A saída da célula é obtida pela aplicação de uma operação de redução (por exemplo, uma concatenação) envolvendo todos os nós intermediários.

- Cada nó intermediário *j* é computado somando a contribuição de todos os nós antecessores *i* < *j*, na forma:

$$x^{(j)} = \sum_{i<j} o^{(i,j)}\left(x^{(i)}\right)$$

- Uma operação especial nula é incluída para indicar a ausência de conexão entre dois nós. A tarefa de aprendizado da célula se resume, assim, a aprender as operações realizadas por suas arestas.

5.1 Relaxação contínua e otimização

- Seja *O* um conjunto de operações candidatas (por exemplo, convolução, *max pooling*, nula) a serem aplicadas a $x^{(i)}$. Para deixar o espaço de busca contínuo, a escolha de uma operação em particular é relaxada pelo emprego da função *softmax* a todas as operações possíveis, produzindo:

$$\bar{o}^{(i,j)} = \sum_{o \in O} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in O} \exp\left(\alpha_{o'}^{(i,j)}\right)} o(x)$$

onde cada peso que leva à mistura do resultado das operações é parametrizado por um vetor $\alpha^{(i,j)}$ que tem dimensão igual à cardinalidade de *O*.

- A NAS então leva ao ajuste conjunto das variáveis contínuas $\alpha = \left\{\alpha^{(i,j)}\right\}$ e dos pesos sinápticos de cada célula.
- Ao final da busca, com uma definição do conjunto $\alpha = \left\{\alpha^{(i,j)}\right\}$, uma arquitetura discreta pode ser obtida substituindo o operador de mistura $\bar{o}^{(i,j)}$ pelo operador mais verossímil, na forma:

$$o^{(i,j)} = \arg\max_{o \in O} \alpha_o^{(i,j)},$$

conforme ilustrado na Figura 3.

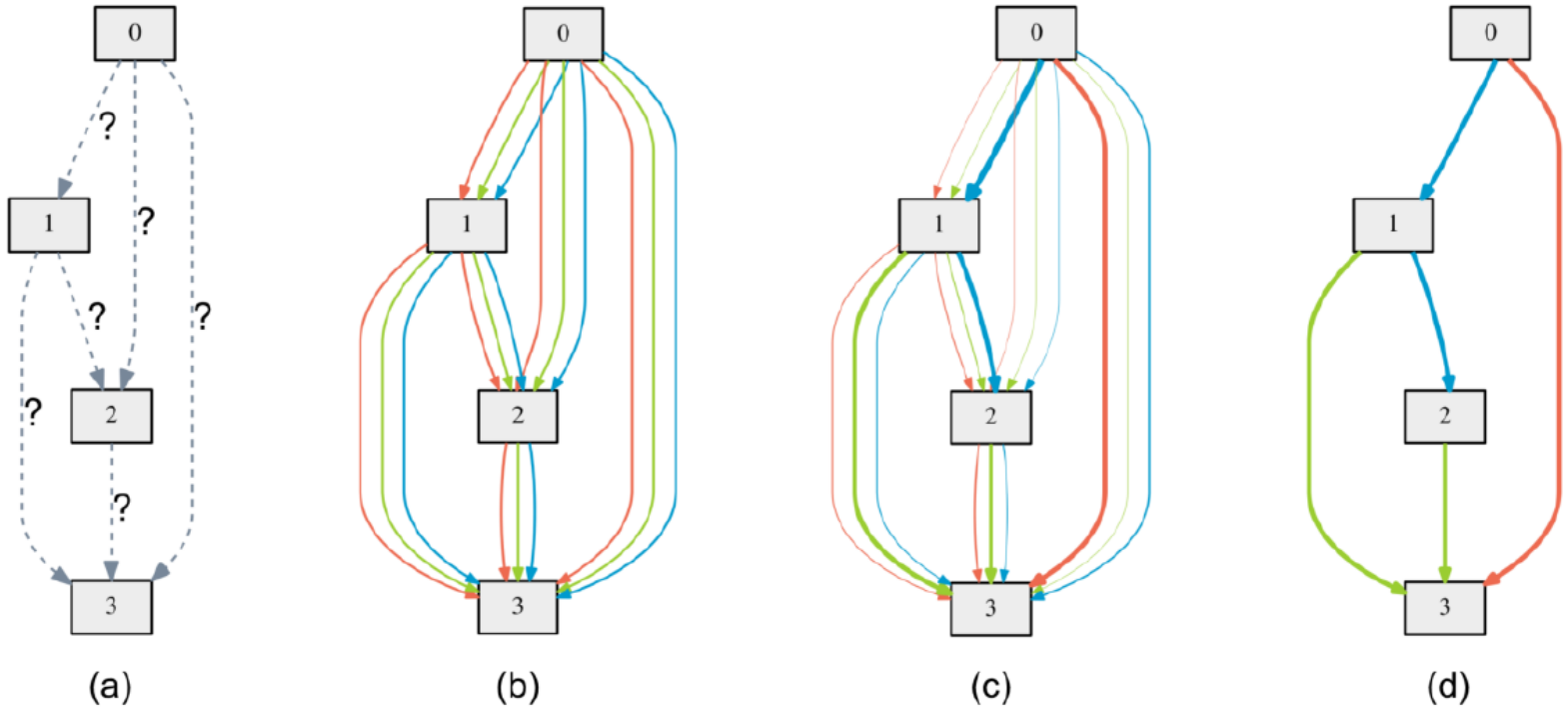


Figura 3 – Processo de aprendizado da célula computacional: (a) Operações nas arestas do grafo são desconhecidas a priori; (b) Relaxação contínua, com a definição de uma mistura de operações candidatas nas arestas do grafo; (c) Aprendizado conjunto das probabilidades da mistura e dos pesos da rede neural; (d) Amostragem da arquitetura final com base nas probabilidades aprendidas.

Fonte: LIU et al. (2019)

- Da mesma forma feita em aprendizado por reforço, em que o reforço está associado ao desempenho junto ao conjunto de validação, e em busca evolutiva, em que o *fitness* está associado ao desempenho junto ao conjunto de validação, DARTS também visa minimizar a perda junto ao conjunto de validação, mas empregando gradiente descendente. O problema de otimização binível resultante é apresentado a seguir:

$$\min_{\alpha} L_{val}\left(w^*(\alpha),\alpha\right)$$
$$\text{s.a.} \quad w^*(\alpha) = \arg\min_{w} L_{train}\left(w,\alpha\right)$$

5.2 P-DARTS

- No DARTS, a busca é realizada em uma rede de 8 células enquanto a arquitetura descoberta é avaliada em uma rede de 20 células. No entanto, pode haver uma grande diferença entre os comportamentos de redes menos e mais profundas (CHEN et al., 2019), quando formadas pelos mesmos módulos constituintes, o que

implica que as estruturas sintetizadas ao longo da busca não são necessariamente as melhores na fase de avaliação. Isso conduz a uma lacuna de profundidade (*depth gap*) entre a pesquisa e a avaliação.

- CHEN et al. (2019) propuseram preencher a lacuna de profundidade com a adoção de uma estratégia que aumenta progressivamente a profundidade da rede durante o processo de busca, de forma que, ao final da busca, a profundidade seja suficientemente próxima ao cenário utilizado na avaliação. Essa proposta foi denominada P-DARTS (*Progressive* DARTS).
- Os resultados de ganho de desempenho e de redução de custo da busca são expressivos, sendo apresentados a seguir.
- Para maiores detalhes acerca das arquiteturas resultantes da busca, sugere-se uma consulta direta a CHEN et al. (2019).

Architecture	Test Err. (%)		Params	Search Cost	Search Method
	C10	C100	(M)	(GPU-days)	
DenseNet-BC [10]	3.46	17.18	25.6	-	manual
NASNet-A + cutout [37]	2.65	-	3.3	1800	RL
AmoebaNet-A + cutout [22]	3.34	-	3.2	3150	evolution
AmoebaNet-B + cutout [22]	2.55	-	2.8	3150	evolution
Hireachical Evolution [17]	3.75	-	15.7	300	evolution
PNAS [16]	3.41	-	3.2	225	SMBO
ENAS + cutout [21]	2.89	-	4.6	0.5	RL
DARTS (first order) + cutout [18]	3.00	17.76†	3.3	1.5‡	gradient-based
DARTS (second order) + cutout [18]	2.76	17.54†	3.3	4.0‡	gradient-based
SNAS + mild constraint + cutout [33]	2.98	-	2.9	1.5	gradient-based
SNAS + moderate constraint + cutout [33]	2.85	-	2.8	1.5	gradient-based
SNAS + aggressive constraint + cutout [33]	3.10	-	2.3	1.5	gradient-based
ProxylessNAS [2] + cutout	2.08	-	5.7	4.0	gradient-based
P-DARTS CIFAR10 + cutout	2.50	16.55	3.4	0.3	gradient-based
P-DARTS CIFAR100 + cutout	2.62	15.92	3.6	0.3	gradient-based
P-DARTS CIFAR10 (large) + cutout	2.25	15.27	10.5	0.3	gradient-based
P-DARTS CIFAR100 (large) + cutout	2.43	14.64	11.0	0.3	gradient-based

Table 1: Comparison with state-of-the-art architectures on CIFAR10 and CIFAR100. † indicates that this result is obtained by training the corresponding architecture on CIFAR100. ‡ We ran the code released by the authors with necessary modifications to fit PyTorch 1.0, and a single run took about 0.5 GPU-days for first order and 2 GPU-days for second order, respectively.

Fonte: CHEN et al. (2019)

Architecture	Test Err. (%)		Params	×+	Search Cost	Search Method
	top-1	top-5	(M)	(M)	(GPU-days)	
Inception-v1 [29]	30.2	10.1	6.6	1448	-	manual
MobileNet [9]	29.4	10.5	4.2	569	-	manual
ShuffleNet 2× (v1) [34]	26.4	10.2	∼5	524	-	manual
ShuffleNet 2× (v2) [19]	25.1	-	∼5	591	-	manual
NASNet-A [37]	26.0	8.4	5.3	564	1800	RL
NASNet-B [37]	27.2	8.7	5.3	488	1800	RL
NASNet-C [37]	27.5	9.0	4.9	558	1800	RL
AmoebaNet-A [22]	25.5	8.0	5.1	555	3150	evolution
AmoebaNet-B [22]	26.0	8.5	5.3	555	3150	evolution
AmoebaNet-C [22]	24.3	7.6	6.4	570	3150	evolution
PNAS [16]	25.8	8.1	5.1	588	225	SMBO
MnasNet-92 [31]	25.2	8.0	4.4	388	-	RL
DARTS (second order) [18]	26.7	8.7	4.7	574	4.0	gradient-based
SNAS (mild constraint) [33]	27.3	9.2	4.3	522	1.5	gradient-based
ProxylessNAS (GPU) [2]	24.9	7.5	7.1	465	8.3	gradient-based
P-DARTS (searched on CIFAR10)	24.4	7.4	4.9	557	0.3	gradient-based
P-DARTS (searched on CIFAR100)	24.7	7.5	5.1	577	0.3	gradient-based

Table 2: Comparison with state-of-the-art architectures on ImageNet (mobile setting).

Fonte: CHEN et al. (2019)

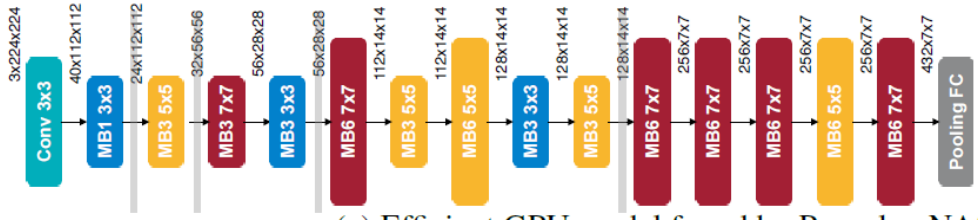
5.3 Extensões de DARTS

- Uma extensão de DARTS e P-DARTS, com o foco na alocação de recursos computacionais de acordo com o *hardware*, conduziu à técnica RAPDARTS (*Resource-Aware* P-DARTS) (GREEN et al., 2019).
- Já a proposta DropNAS tratou do problema de coadaptação existente no DARTS, usando *dropout* (HONG et al., 2020), com ganhos de desempenho.
- ProxylessNAS (CAI et al., 2019) também estendeu DARTS. Esta técnica armazena uma grande rede superparametrizada na memória do sistema, porque a rede é muito grande para caber em uma GPU. Durante a avaliação, uma sub-rede é amostrada e transferida para a GPU para avaliação. Os gradientes são calculados e usados para atualizar os pesos compartilhados da rede superparametrizada.
- Com essas iniciativas, ProxylessNAS eliminou a necessidade de uso de tarefas *proxy*, como treinar em um conjunto de dados menor, aprender com apenas alguns blocos ou treinar apenas por alguns períodos.

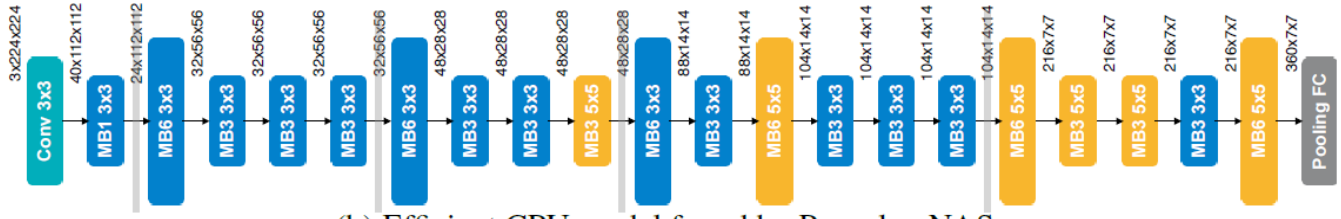
- Essas arquiteturas otimizadas em tarefas *proxy* não têm garantia de serem plenamente transferíveis para a tarefa de destino. ProxylessNAS, por sua vez, aprende diretamente as arquiteturas para tarefas de destino em grande escala, além de aliviar bastante o alto consumo de memória verificado no DARTS, o colocando no mesmo nível de um treinamento regular.

Fonte: CHEN et al. (2019)

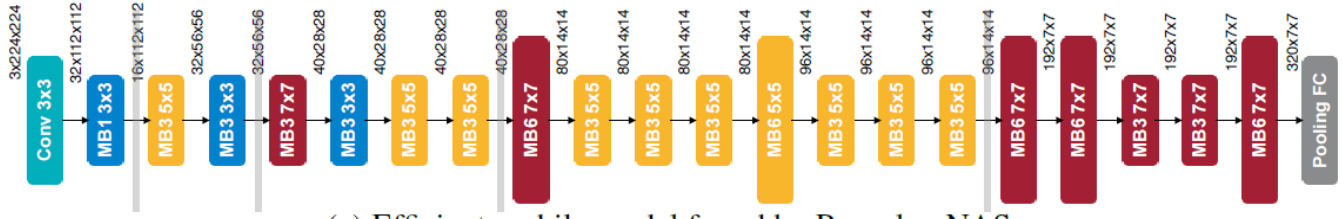
Figura 4 – Esquemas com e sem tarefas *proxy*, sendo que ProxylessNAS usa o esquema (2).



(a) Efficient GPU model found by ProxylessNAS.



(b) Efficient CPU model found by ProxylessNAS.



(c) Efficient mobile model found by ProxylessNAS.

Model	Top-1 (%)	GPU latency	CPU latency	Mobile latency
Proxyless (GPU)	75.1	5.1ms	204.9ms	124ms
Proxyless (CPU)	75.3	7.4ms	138.7ms	116ms
Proxyless (mobile)	74.6	7.2ms	164.1ms	78ms

Fonte: CAI et al. (2019)

Figura 5 – Arquiteturas otimizadas para *hardwares* distintos junto à mesma tarefa de aprendizado.

6 Once-for-All

- Além do conteúdo do paper original, parte do material apresentado nesta seção foi obtida do link: https://ofa.mit.edu/
- A técnica *Once-for-All* (CAI et al., 2020) treina uma única rede neural profunda (abordagem *one-shot*) que contém um número exponencial de versões mais simples e aninhadas nesta estrutura completa. Com isso, o treinamento envolve uma parte dos pesos sinápticos a cada etapa, reduzindo a demanda por memória e processamento ao longo do treinamento, e permite ajustar conexões sinápticas que participam de um número exponencial de modelos de aprendizado aninhados.
- Apenas como um exemplo, filtros convolucionais 7 x 7, 5 x 5 e 3 x 3, podem ter pesos compartilhados e concebidos de forma aninhada (o de menor dimensão centrado internamente no de maior dimensão, além de incluir um mapeamento linear), de modo que o treinamento de qualquer dessas três dimensões vai levar, ao menos em parte, ao ajuste de pesos para as demais dimensões, simultaneamente.

- Além desta ideia de aninhamento, que promove escalabilidade no treinamento e é implementada por uma técnica denominada *progressive shrinking*, a busca pela melhor arquitetura é feita de forma independente da etapa de treinamento, empregando um algoritmo de computação evolutiva (uma meta-heurística de otimização). Essa independência confere ainda mais escalabilidade à técnica.
- Diferente de grande parte das demais propostas de NAS, não é necessário treinar do zero (*from scratch*) as várias propostas de arquitetura da rede neural profunda, bastando amostrá-las como sub-redes da única rede completa.
- É impressionante constatar também que foi verificado, em uma ampla gama de aplicações práticas, que o desempenho da sub-rede amostrada da única rede completa é superior àquele produzido pela mesma sub-rede quando treinada do zero.
- Os resultados experimentais para a técnica *Once-for-All* (CAI et al., 2020) evidenciam uma redução em ordens de magnitude no número de parâmetros

ajustáveis e no uso de recursos computacionais para se chegar na melhor arquitetura de rede neural profunda. Além disso, é mantido um nível competitivo de desempenho junto a problemas desafiadores na literatura, considerando múltiplos objetivos de projeto a serem otimizados, como latência de *hardware* e acurácia do modelo de aprendizado (CAI et al., 2020).

- A conquista de uma separação entre as fases de treinamento e busca da arquitetura ótima é, sem dúvida, o ponto forte desta proposta, pois traz escalabilidade, geralmente medida na forma de MACs (*# of multiplier-accumulator operations*).
- Explorando esta escalabilidade, os autores da técnica Once-for_All disponibilizaram redes neurais pré-treinadas e com arquitetura otimizada para mais de 50 dispositivos de ponta, incluindo dispositivos móveis, CPUs, GPUs e FPGAs. E uma das razões para esta iniciativa foi a constatação de que essas arquiteturas otimizadas, pré-treinadas e de alto desempenho podem ser transferidas (*transfer learning*) com sucesso a outros cenários de aplicação.

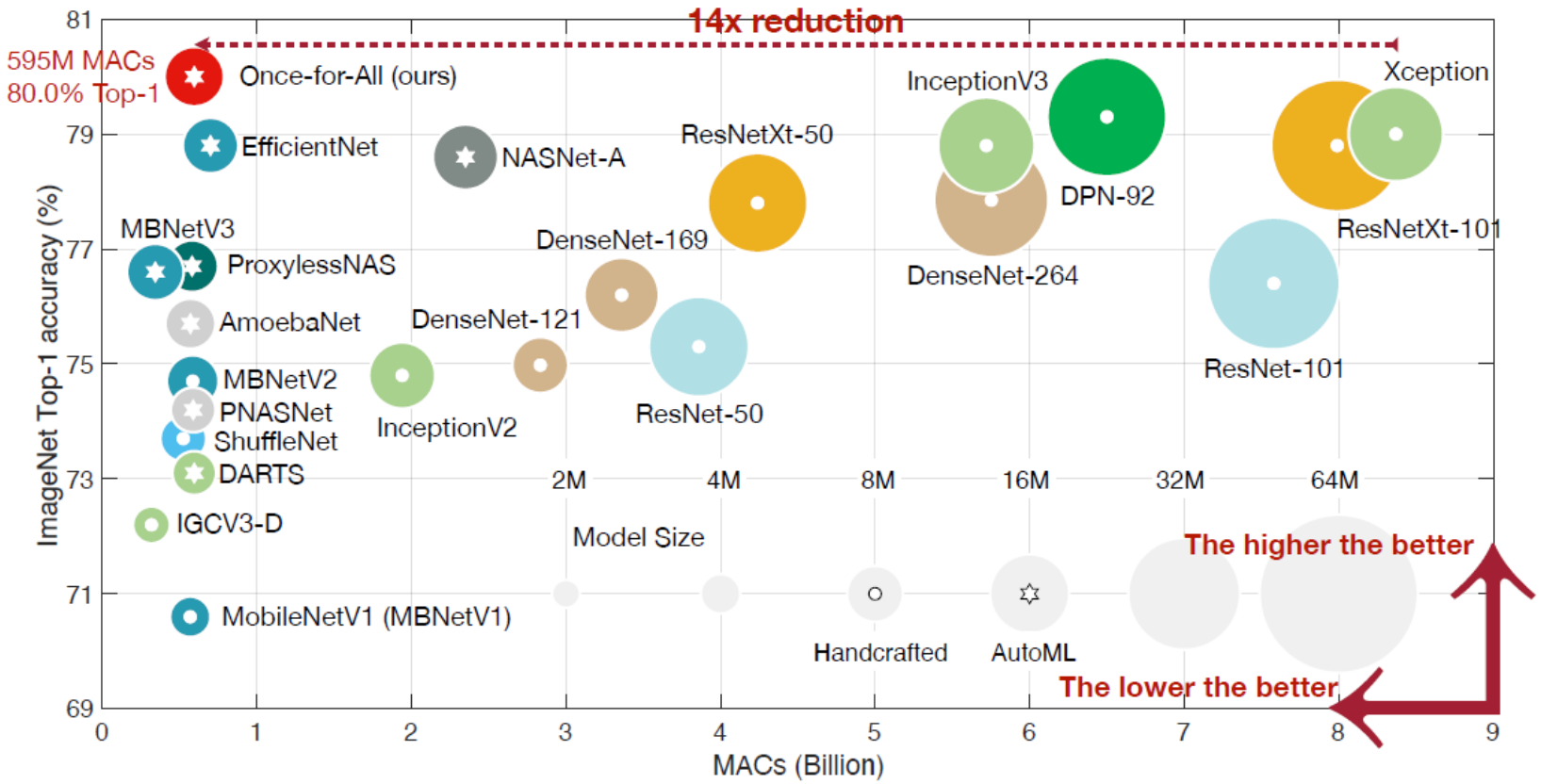
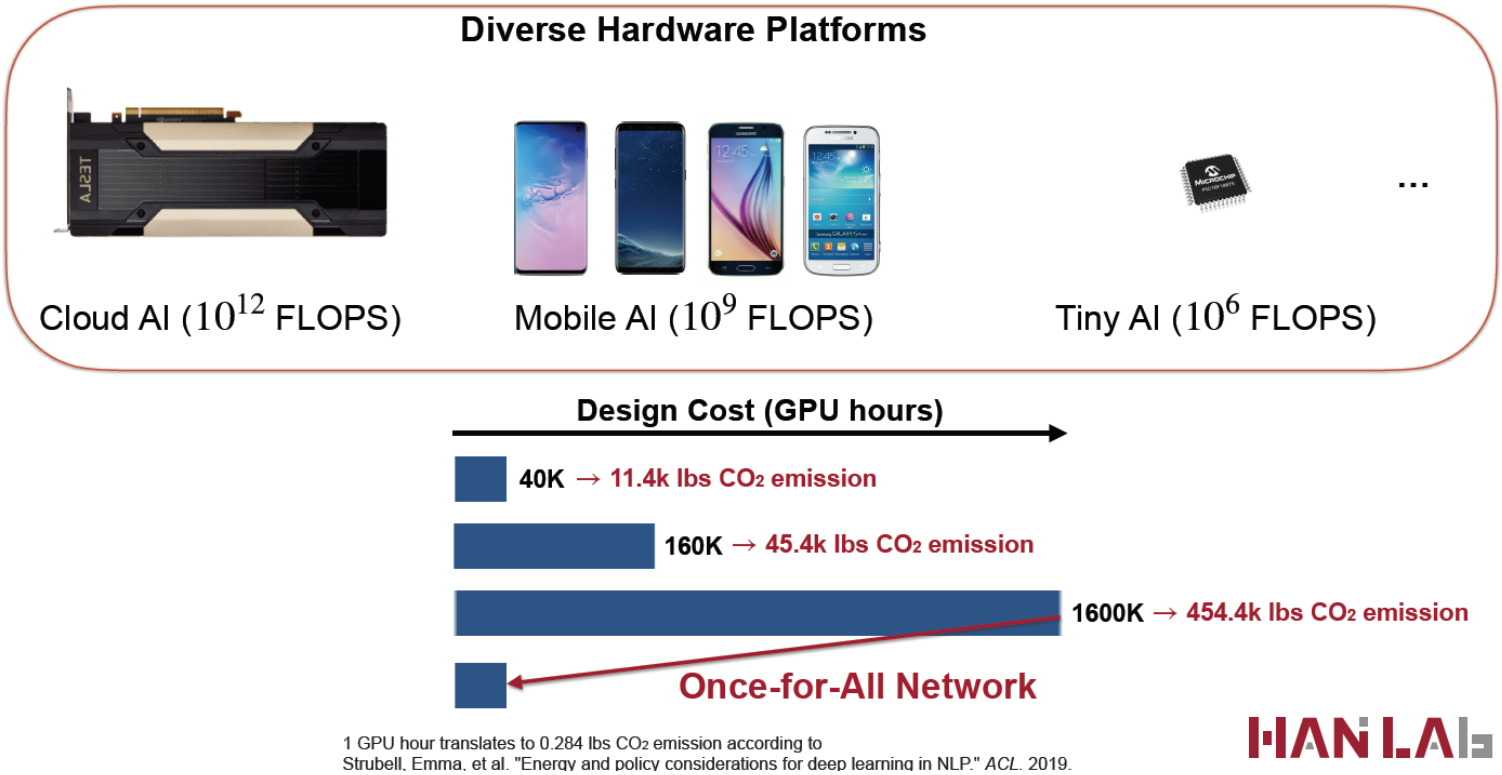


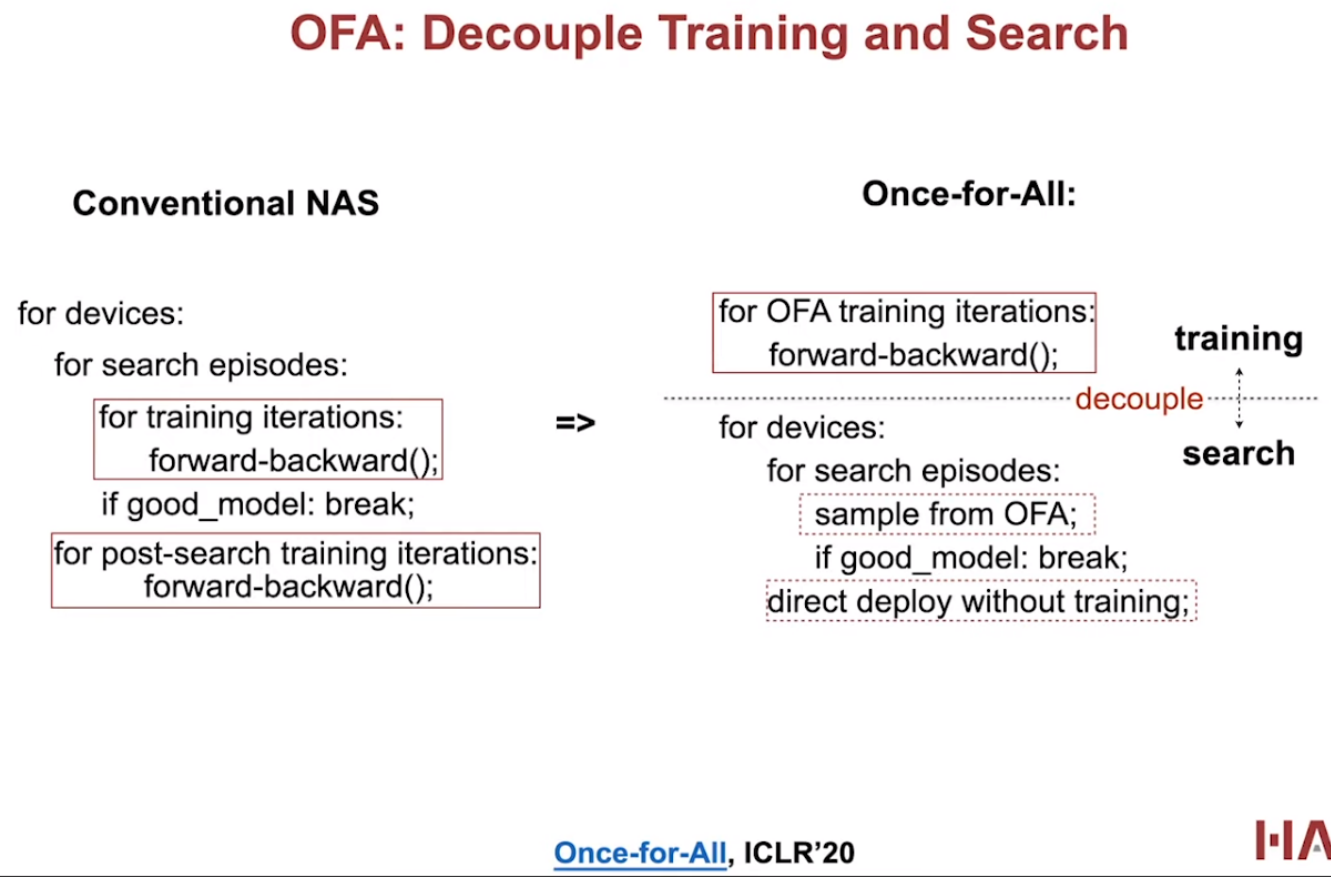
Figure 2: Comparison between OFA and state-of-the-art CNN models on ImageNet. OFA provides 80.0% ImageNet top1 accuracy under the mobile setting (< 600M MACs).

Fonte: CAI et al. (2020)

Challenge: Efficient Inference on Diverse Hardware Platforms



Fonte: CAI et al. (2020) (https://ofa.mit.edu)



Fonte: CAI et al. (2020) (https://ofa.mit.edu)

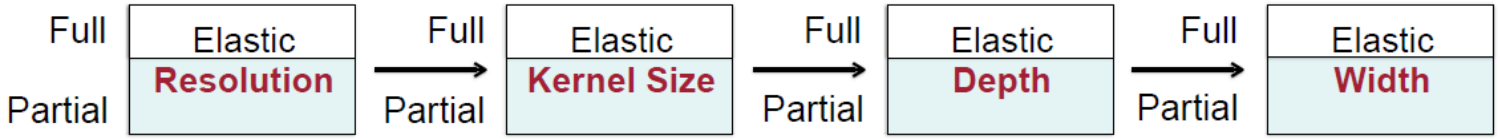


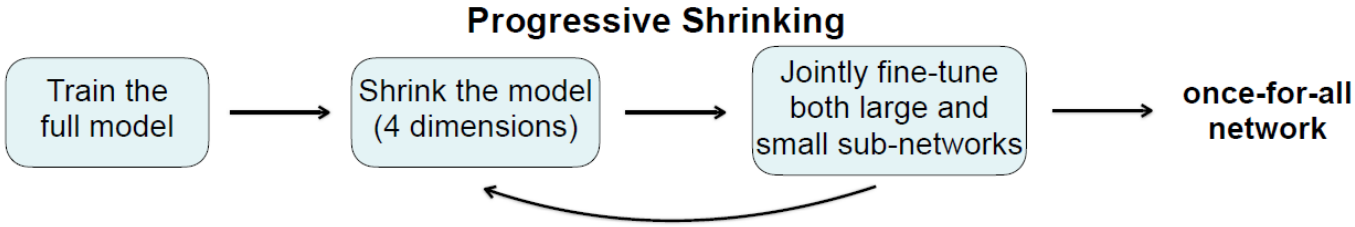
Figure 3: Illustration of the progressive shrinking process to support different depth D, width W, kernel size K and resolution R. It leads to a large space comprising diverse sub-networks ($> 10^{19}$).

Fonte: CAI et al. (2020)

- Sub-redes menores estão aninhadas em sub-redes maiores, incluindo quatro dimensões de encolhimento progressivo (do inglês *progressive shrinking*): resolução, tamanho do *kernel*, profundidade e largura.
- Como já havia sido evidenciado no caso da EfficientNet (que considera explicitamente três dimensões), as quatro dimensões estão correlacionadas, o que pode ser verificado empiricamente, não cabendo assim a realização de um ajuste de escala independente por dimensão.

Progressive Shrinking

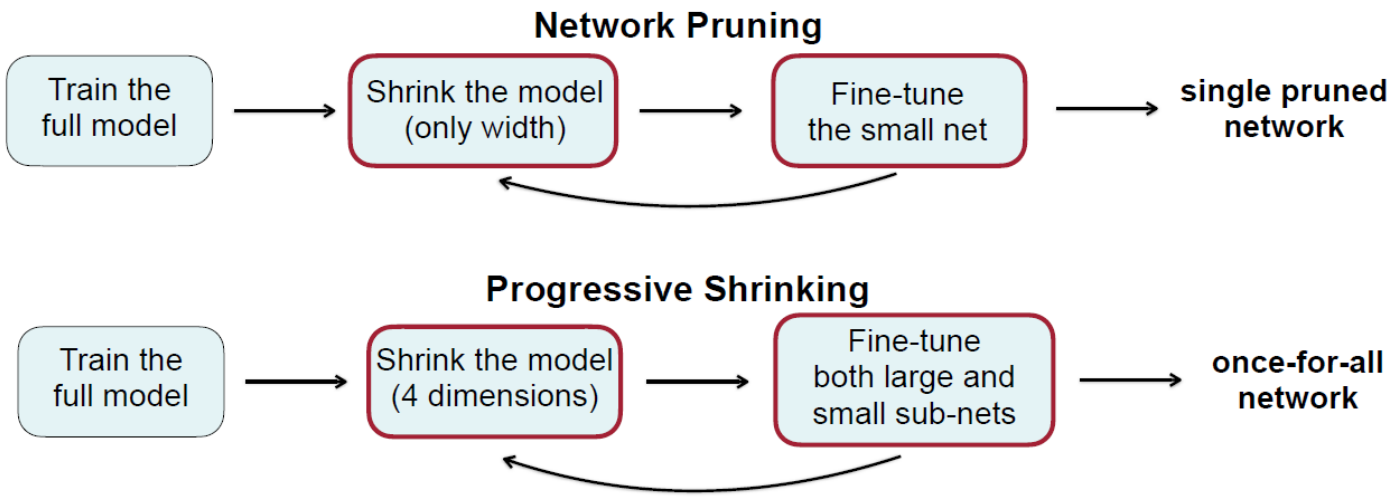
- More than 10^{19} **different sub-networks** in a single once-for-all network, covering 4 different dimensions: **resolution**, **kernel size**, **depth**, **width**.
- Directly optimizing the once-for-all network from scratch is much more challenging than training a normal neural network given so many sub-networks to support.



- Small sub-networks are nested in large sub-networks.
- Cast the training process of the once-for-all network as a progressive shrinking and joint fine-tuning process.

Fonte: CAI et al. (2020) (https://ofa.mit.edu)

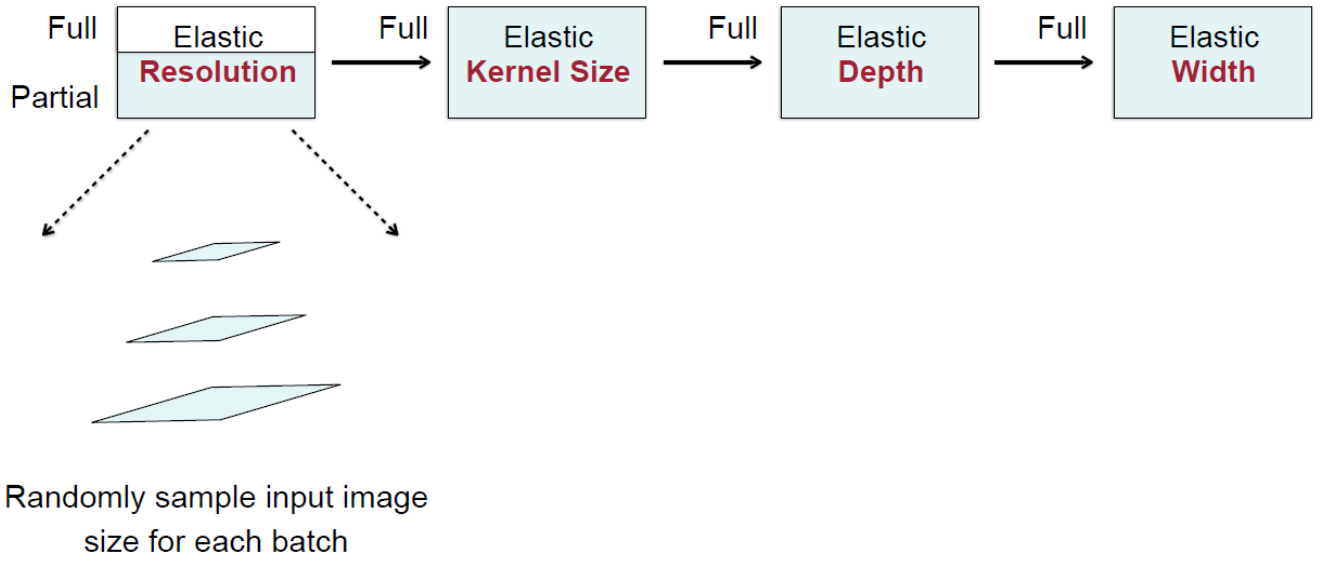
Connection to Network Pruning



- Progressive shrinking can be viewed as a generalized network pruning with much higher flexibility across 4 dimensions.

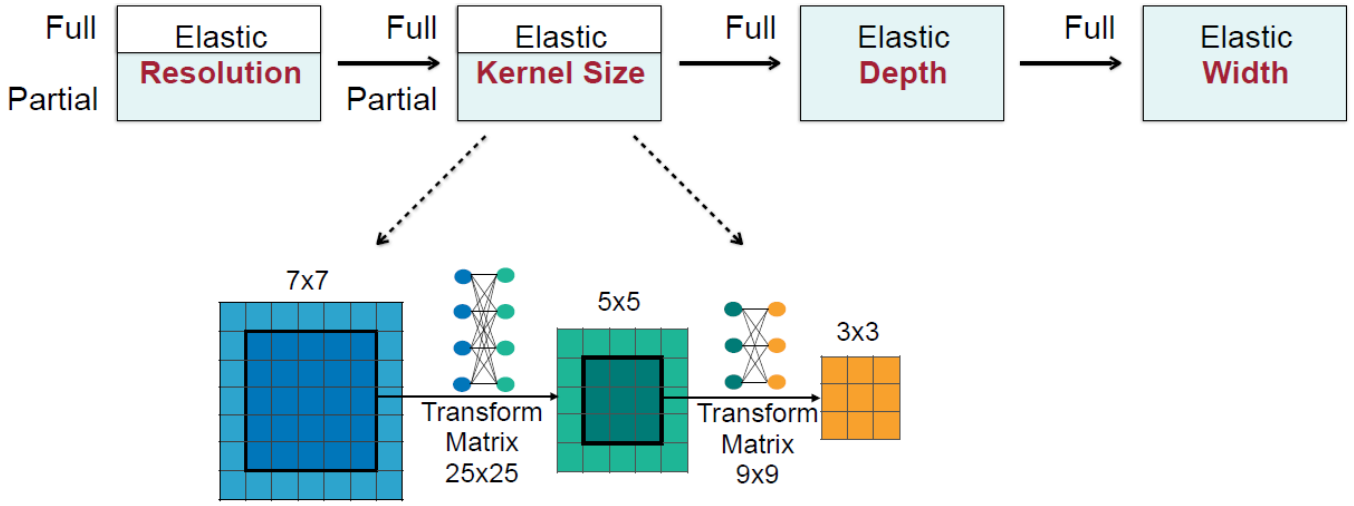
Fonte: CAI et al. (2020) (https://ofa.mit.edu)

Progressive Shrinking



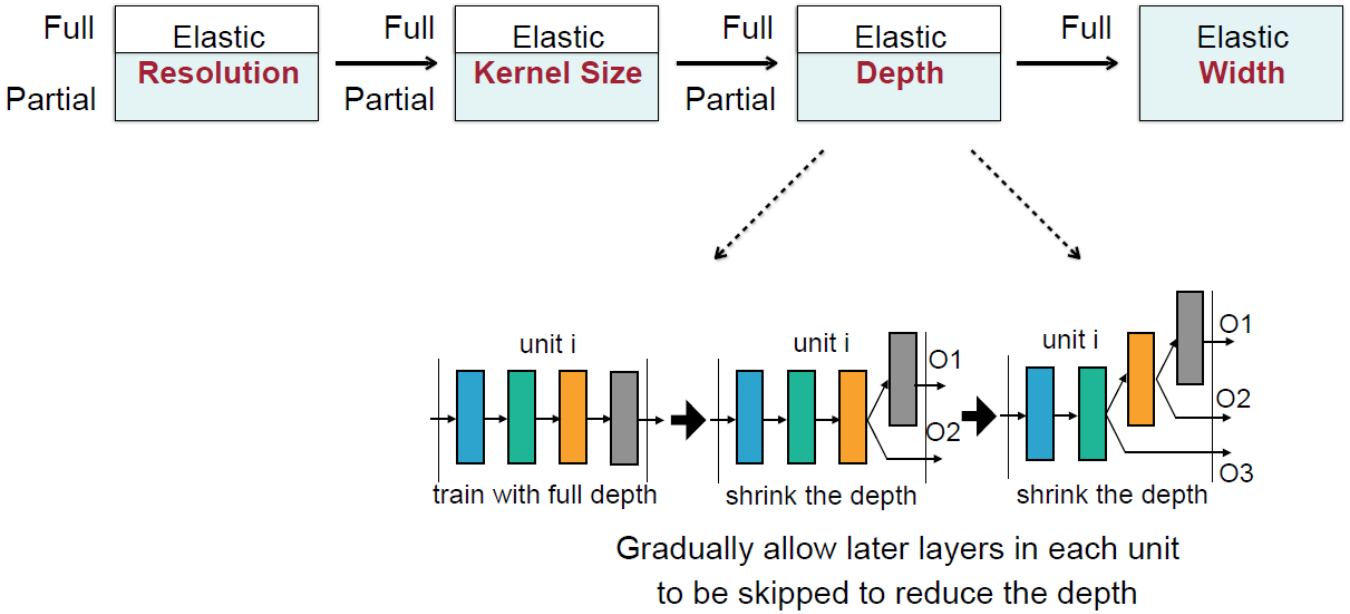
Fonte: CAI et al. (2020) (https://ofa.mit.edu)

Progressive Shrinking



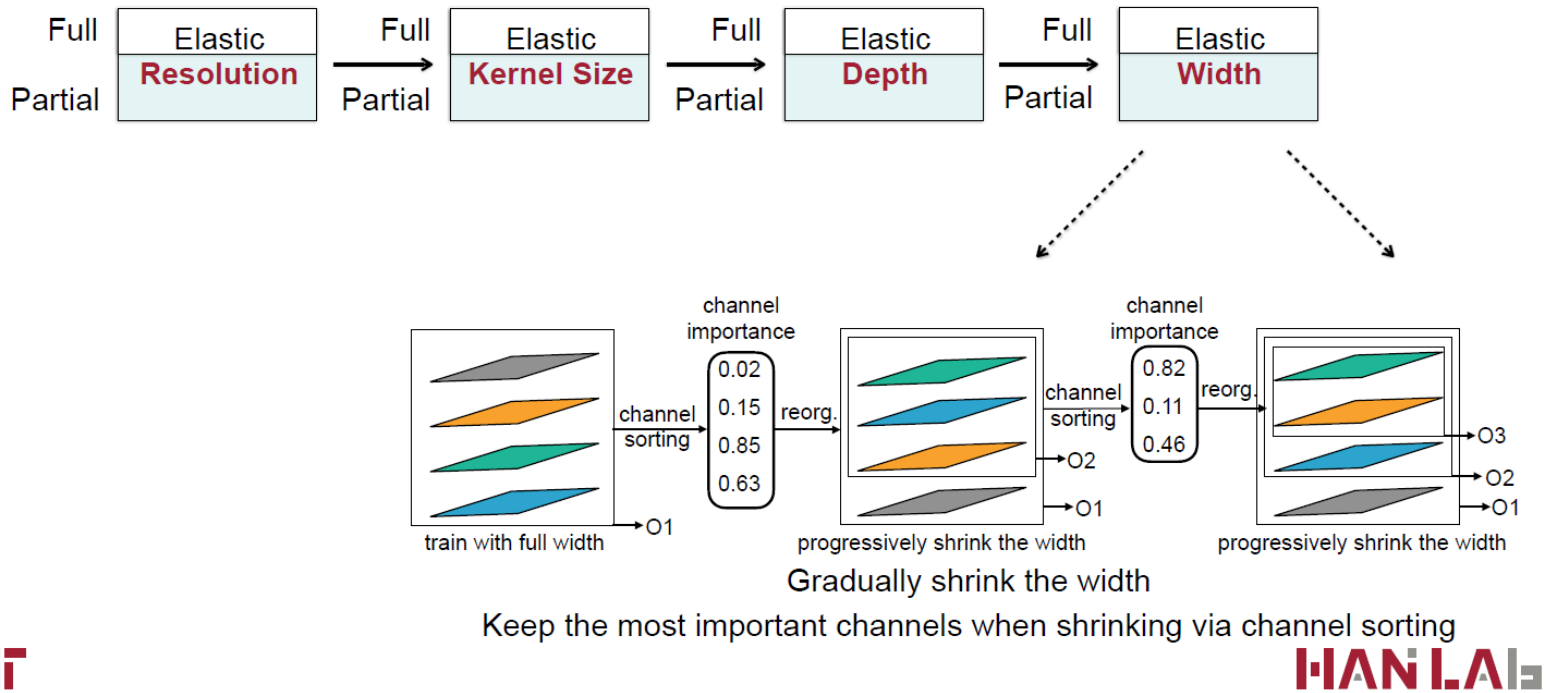
Fonte: CAI et al. (2020) (https://ofa.mit.edu)

Progressive Shrinking

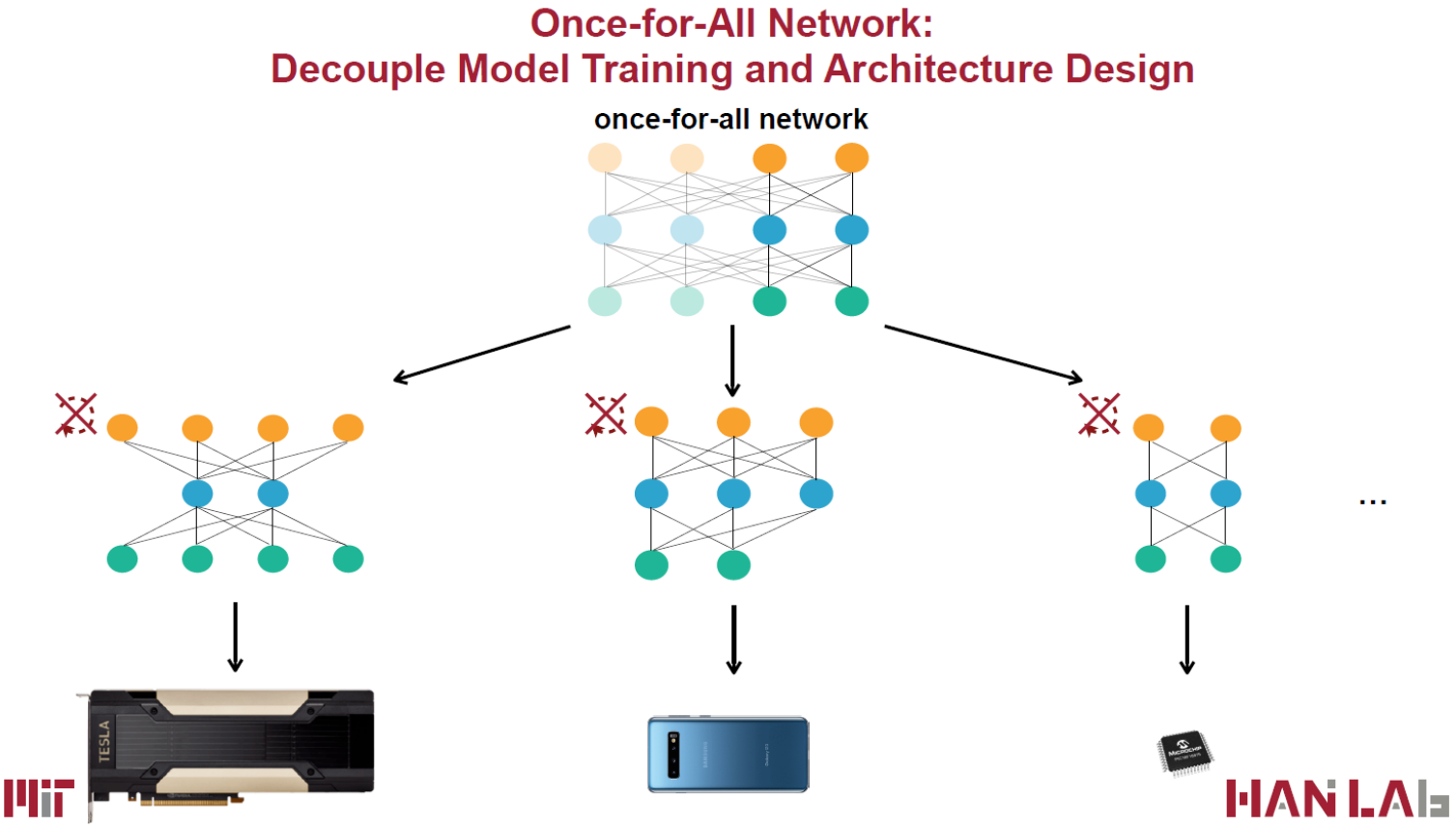


Fonte: CAI et al. (2020) (https://ofa.mit.edu)

Progressive Shrinking



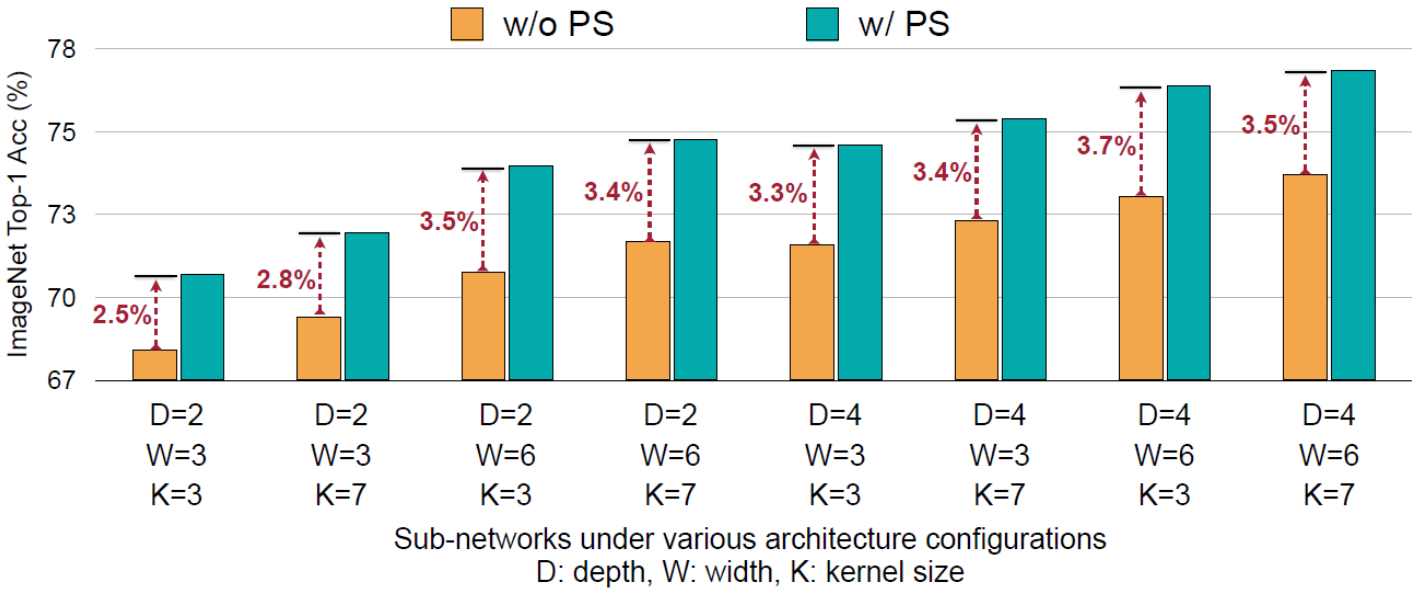
Fonte: CAI et al. (2020) (https://ofa.mit.edu)



Fonte: CAI et al. (2020) (https://ofa.mit.edu)

- A técnica de encolhimento progressivo (do inglês *progressive shrinking*) evita que os modelos exibam interferência cruzada, eliminando a necessidade de retreinamento, o que promove ganhos impressionantes de escala.

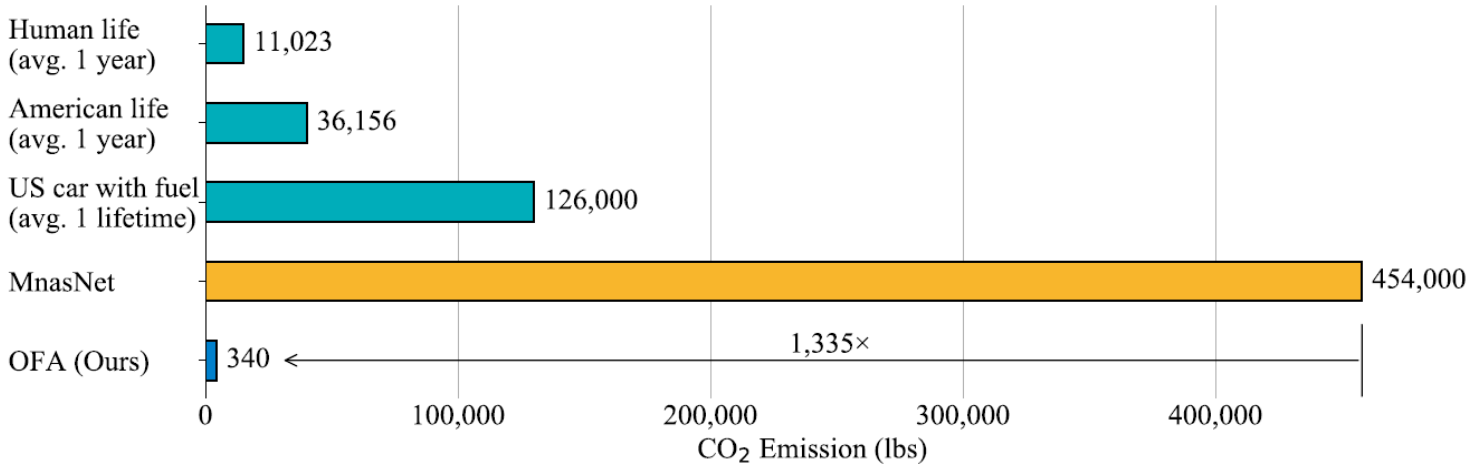
Performances of Sub-networks on ImageNet



- Progressive shrinking consistently improves accuracy of sub-networks on ImageNet.

Fonte: CAI et al. (2020) (https://ofa.mit.edu)

OFA Saves Orders of Magnitude Design Cost



- Geen AI is important. The computation cost of OFA stays constant with #hardware platforms, reducing the carbon footprint by **1,335x** compared to MnasNet under 40 platforms.

Fonte: CAI et al. (2020) (https://ofa.mit.edu)

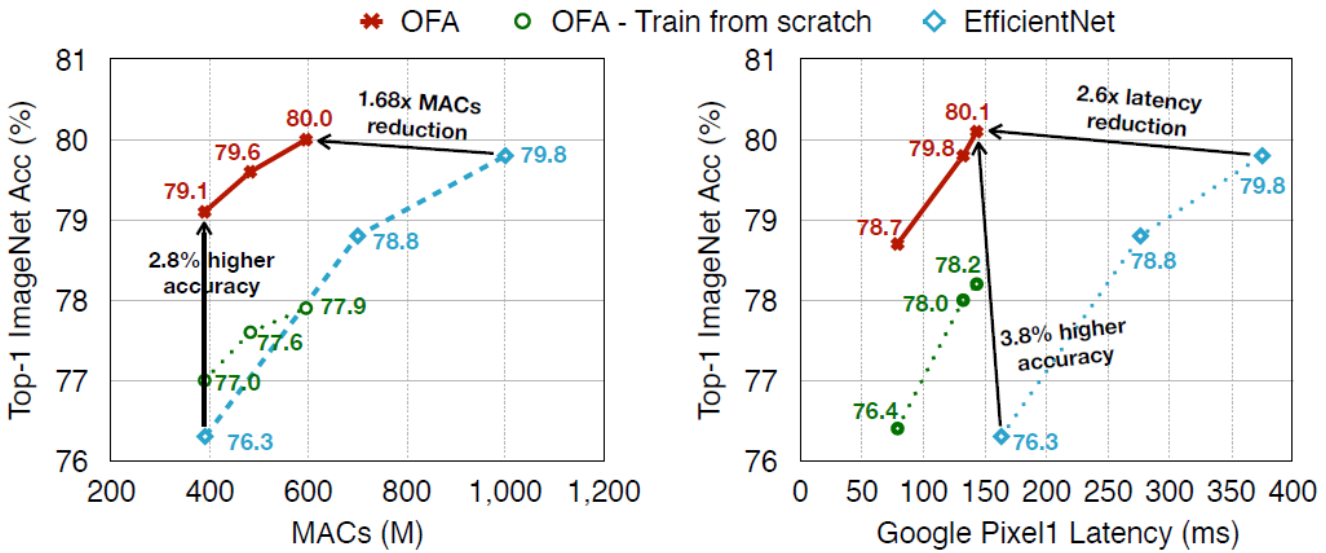
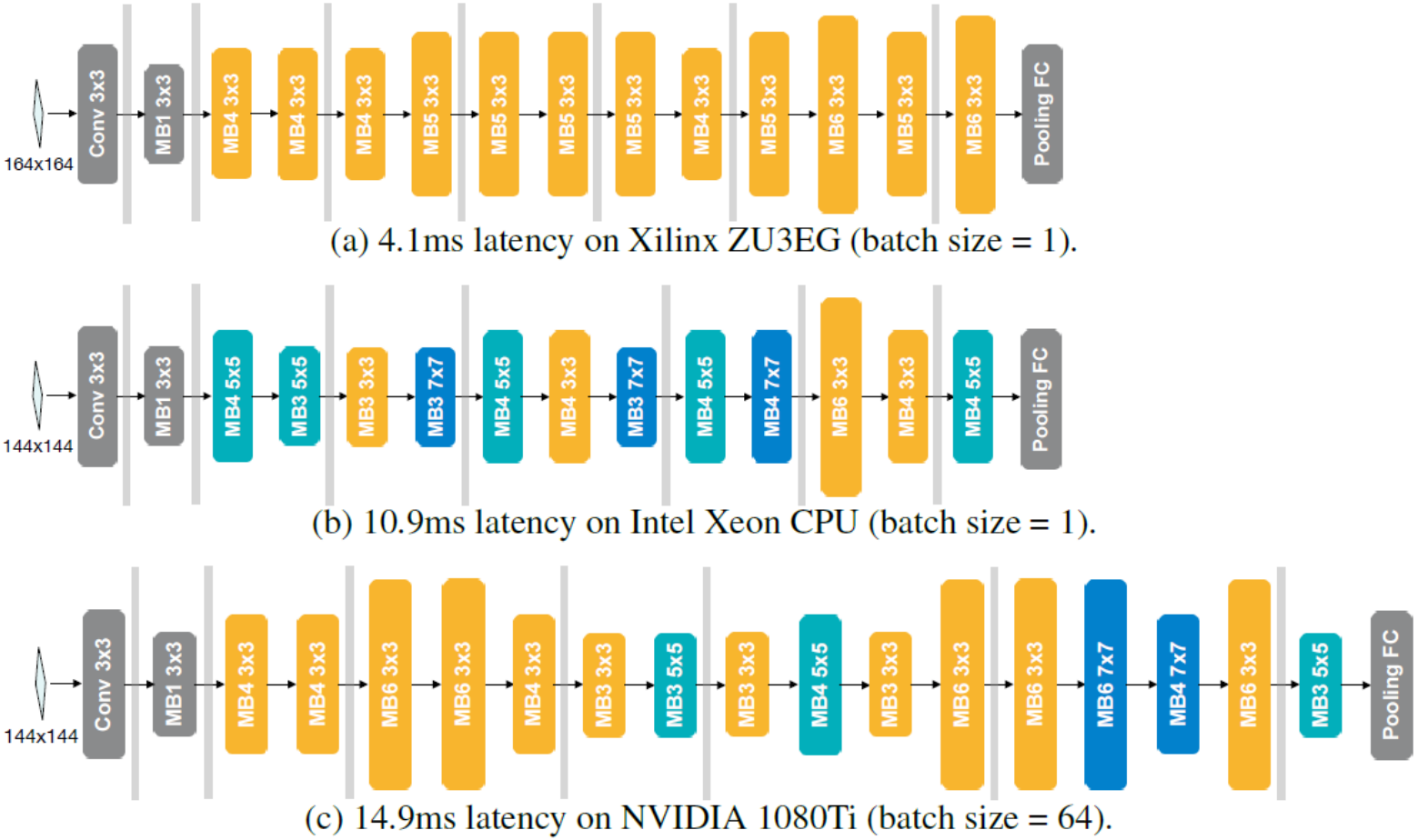


Figure 9: OFA achieves 80.0% top1 accuracy with 595M MACs and 80.1% top1 accuracy with 143ms Pixel1 latency, setting a new SOTA ImageNet top1 accuracy on the mobile setting.

Fonte: CAI et al. (2020)

- *"Released 50 different pre-trained OFA models on diverse hardware platforms (CPU/GPU/FPGA/DSP)."*
- *"Customize our models for each platform to achieve the best accuracy-efficiency trade-off, especially on resource-constrained edge devices."*
- *"Non-specialized neural networks do not fully utilize the hardware resource. There is a large room for improvement via neural network specialization."*



(a) 4.1ms latency on Xilinx ZU3EG (batch size = 1).

(b) 10.9ms latency on Intel Xeon CPU (batch size = 1).

(c) 14.9ms latency on NVIDIA 1080Ti (batch size = 64).

Figure 14: OFA can design specialized models for different hardware and different latency constraint. "MB4 3x3" means "mobile block with expansion ratio 4, kernel size 3x3". FPGA and GPU models are wider than CPU model due to larger parallelism. Different hardware has different cost model, leading to different optimal CNN architectures. OFA provides a unified and efficient design methodology.

Fonte: CAI et al. (2020)

7 Meta-heurísticas para sintonia de hiperparâmetros

- Meta-heurísticas de otimização representam um conjunto de técnicas usadas para resolver problemas de busca não-estruturada em espaços matemáticos. A não-convexidade do problema e a dificuldade na aplicação de técnicas de gradiente descendente no ajuste dos hiperparâmetros são as motivações mais fortes. Iniciativas de uso de meta-heurísticas de otimização no contexto de aprendizado profundo já tenham conduzido a resultados competitivos em outras frentes de aplicação (IBA & NOMAN, 2020).
- Por exemplo, a aplicação de algoritmos evolutivos para a síntese de *autoencoders* representa uma frente de estudo promissora (SUGANUMA et al., 2018; HAJEWSKI et al., 2020).
- Muitas meta-heurísticas poderiam ser consideradas, pois há várias propostas competentes na exploração de espaços contínuos de elevada dimensão (BÄCK et al., 2000a; BÄCK et al., 2000b; GASPAR-CUNHA et al., 2013).

- Basicamente, um algoritmo evolutivo opera no espaço de busca, definindo propostas de hiperparâmetros a cada geração, as quais são avaliadas com base no desempenho do modelo de aprendizado como um todo. As propostas que obtêm uma melhor avaliação têm mais chances de gerar descendentes para a próxima geração. Espera-se que o desempenho do melhor indivíduo a cada geração sofra melhoras continuadas ao longo das gerações.
- O material a seguir é uma síntese daquele utilizado na vídeo-aula disponível em:

https://www.youtube.com/watch?v=KbD0G0iS1SU&t=12s

7.1 Fundamentos de computação evolutiva

- A computação evolutiva é uma meta-heurística populacional baseada no princípio da seleção natural de Darwin (EIBEN & SMITH, 2015).
- Se baseiam na seguinte sequência básica comum: realização de reprodução, imposição de variações aleatórias, promoção de competição e execução de seleção de indivíduos de uma dada população.

- Logo, a evolução é caracterizada por um processo constituído de 4 passos:

    1. Reprodução com herança genética;
    2. Introdução de variação aleatória em uma população de indivíduos;
    3. Avaliação dos novos indivíduos gerados;
    4. Aplicação da "seleção natural" para a produção da próxima geração.

- A variação cria diversidade na população, a diversidade é transmitida por herança e a seleção elimina indivíduos menos aptos, quando comparados aos demais indivíduos da população.
- A seleção natural em si não tem um propósito definido, mas em computador é possível impor um propósito matematicamente definido, por exemplo, para implementar meta-heurísticas de otimização.
- Neste sentido, os algoritmos evolutivos se caracterizam como meta-heurísticas que abrangem metodologias de busca em espaços de soluções candidatas, capazes de

gerenciar operadores computacionais de busca local (que realizam intensificação) e de busca global (que realizam exploração).

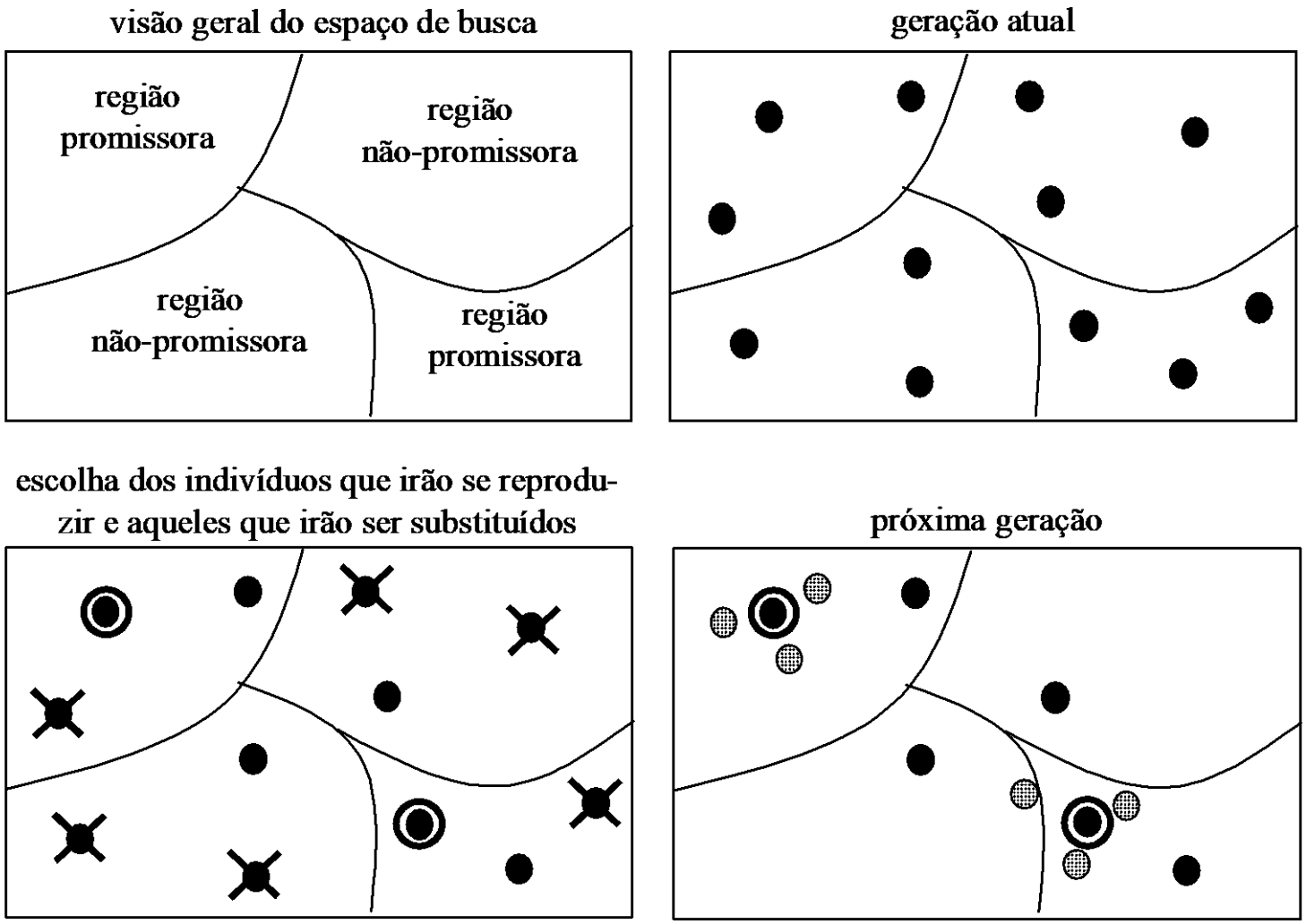
- Como praticamente todas as meta-heurísticas, os algoritmos evolutivos geralmente **NÃO** apresentam as seguintes propriedades:
  - ✓ Garantia de obtenção da solução ótima;
  - ✓ Garantia de convergência;
  - ✓ Garantia de custo máximo para se chegar a uma solução.

7.2 Formalização matemática

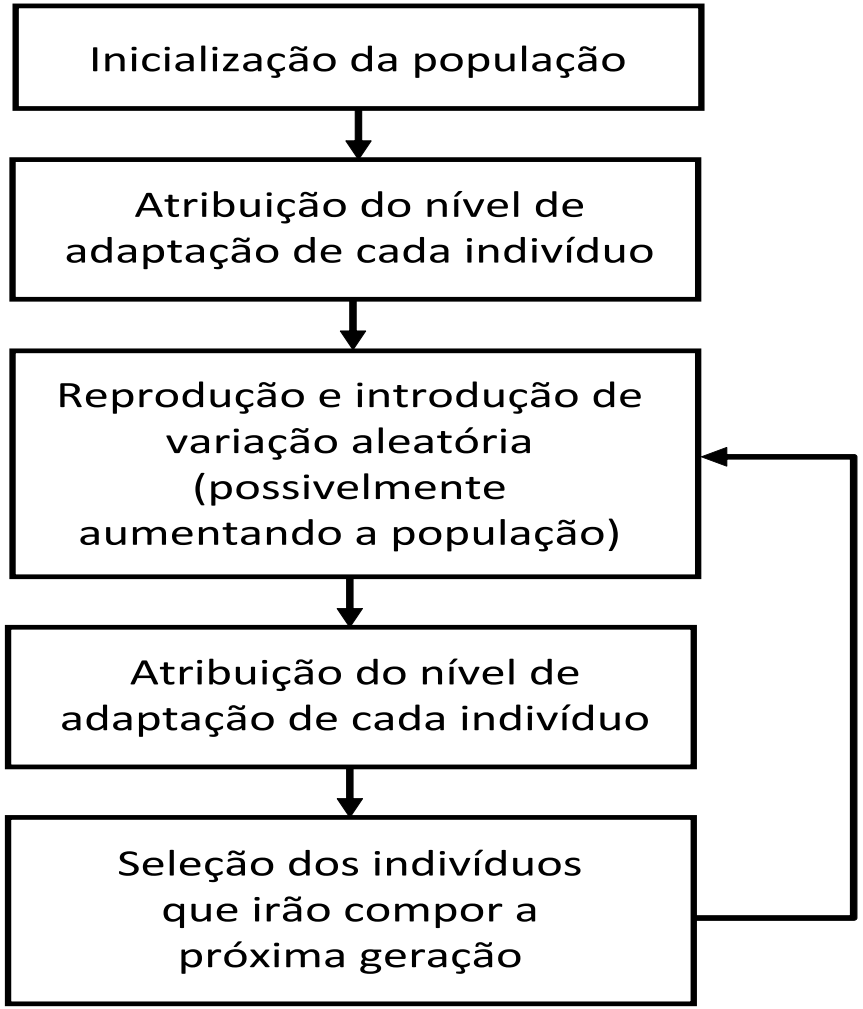
- Representação genotípica: Codificação dos candidatos à solução Podem existir vários tipos de representação para o mesmo problema, mas qualquer uma delas deve ser capaz de representar todas as soluções-candidatas, mesmo que nem sempre de forma única.
- Representação fenotípica: Interpretação da representação genotípica.
- Espaço de busca: Tem como elementos todos os possíveis candidatos à solução do problema e é definido a partir da representação genotípica.

- Função de adaptação ou *fitness*: Atribui a cada elemento do espaço de busca um valor de adaptação, que será usado como medida *relativa* de desempenho. Representa a pressão do ambiente sobre o fenótipo dos indivíduos.
- Operadores de inicialização: Produzem a primeira geração de indivíduos (população inicial), tomando elementos do espaço de busca.
- Operadores genéticos: Implementam o mecanismo de introdução de variabilidade aleatória no genótipo da população.
- Operadores de seleção: Implementam o mecanismo de "seleção natural".
- Índice de diversidade: Geralmente representa a distância média entre os indivíduos da população corrente. Pode ser medido no genótipo, no fenótipo ou levando-se em conta apenas o *fitness* (medida aproximada e de baixo custo computacional).
- É necessário atribuir valores aos parâmetros envolvidos: tamanho da população; probabilidade de aplicação dos operadores genéticos; argumentos dos operadores de seleção; e argumentos do critério de parada.

7.3 Visão pictórica da força evolutiva



7.4 Fluxograma de um algoritmo evolutivo



7.5 Pseudo-código de um algoritmo evolutivo

Procedimento [*P*] = algoritmo_evolutivo(*N*, *pc, pm*)
  P'' ← inicializa(*N*)
  fit ← avalia(*P''*)
  t ← 1
  Enquanto condição_de_parada for FALSO **faça**,
    P ← seleciona(*P''*, *fit*)
    P' ← reproduz(*P*, *fit, pc*)
    P'' ← varia(*P', pm*)
    fit ← avalia(*P''*)
    t ← t + 1
  Fim Enquanto
Fim Procedimento

- Quais são as diferenças entre este pseudo-código e o fluxograma do slide anterior?

8 Referências bibliográficas

BÄCK, T.; D.B. FOGEL; MICHALEVICZ, Z. (Eds.) "Evolutionary Computation 1: Basic Algorithms and Operators", CRC Press, 2000a.

BÄCK, T.; D.B. FOGEL; MICHALEVICZ, Z. (Eds.) "Evolutionary Computation 2: Advanced Algorithms and Operations", CRC Press, 2000b.

BERGSTRA, J.; BARDENET, R.; BENGIO, Y.; KÉGL, B. "Algorithms for hyper-parameter optimization", Advances in Neural Information Processing Systems (NIPS), vol. 24, pp. 2546–2554, 2011.

BERGSTRA, J.; BENGIO, Y. "Random search for hyper-parameter optimization", Journal of Machine Learning Research, vol. 13, no. 1, pp. 281–305, 2012.

CAI, H.; ZHU, L.;HAN, S. "ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware", arXiv:1812.00332v2, 2019.

CAI, H.; GAN, C.; WANG, T.; ZHANG, Z.; HAN, S. "Once-for-All: Train One Network and Specialize for Efficient Deployment", arXiv:1908.09791v5, 2020.

CHEN, X.; XIE, L.; WU, J.; TIAN, Q. "Progressive Differentiable Architecture Search: Bridging the Depth Gap between Search and Evaluation", arXiv:1904.12760v1, 2019.

EGGENSPERGER, K.; FEURER, M.; HUTTER, F.; BERGSTRA, J.; SNOEK, J.; HOOS, H.; LEYTON-BROWN, K. "Towards an empirical foundation for assessing Bayesian optimization of hyperparameters", BayesOpt Work, pp. 1-5, 2013.

EIBEN, A.E.; SMITH, J.E. "Introduction to Evolutionary Computing", Natural Computing Series, Springer, 2nd edition, 2015.

ELSKEN, T.; METZEN, J.H.; HUTTER, F. "Neural Architecture Search: A Survey", Journal of Machine Learning Research, vol. 20, pp. 1-21, 2019.

GASPAR-CUNHA, A., ANTUNES, C.H.; TAKAHASHI, R. (Eds.) "Manual de Computação Evolutiva e Meta-Heurística", Editora da UFMG, 2013.

GREEN, S.; VINEYARD, C.M.; HELINSKI, R.; KOÇ, Ç.K. "RAPDARTS: Resource-Aware Progressive Differentiable Architecture Search", arXiv:1911.05704v1, 2019.

HAJEWSKI, J.; OLIVEIRA, S.; XING, X. "Distributed Evolution of Deep Autoencoders", arXiv:2004.07607v1, 2020.

HONG, W.; LI, G.; ZHANG, W.; TANG, R.; WANG, Y.; LI, Z.; YU, Y. "DropNAS: Grouped Operation Dropout for Differentiable Architecture Search", Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI), 2020.

HUTTER, F.; KOTTHOFF, L.; VANSCHOREN, J. (Eds.) "Automatic Machine Learning: Methods, Systems, Challenges", Springer, 2019.

IBA, H.; NOMAN, N. "Deep Neural Evolution: Deep Learning with Evolutionary Computation", Springer, 2020.

KOLLER, D.; FRIEDMAN, N. "Probabilistic Graphical Models: Principles and Techniques", The MIT Press, 2009.

LIU, H.; SIMONYAN, K.; YANG, Y. "DARTS: Differentiable Architecture Search", Proceedings of the International Conference on Learning Representation", 2019.

LUO, G. "A review of automatic selection methods for machine learning algorithms and hyper-parameter values", Network Modeling Analysis in Health Informatics and Bioinformatics, vol. 5, pp. 1-16, 2016.

REAL, E.; AGGARWAL, A.; HUANG, Y.; LE, Q.V. "Regularized evolution for image classifier architecture search", arXiv:1802.01548, 2018.

REN, P.; XIAO, Y.; CHANG, X.; HUANG, P.-Y.; LI, Z.; CHEN, X.; WANG, X. "A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions", arXiv:2006.02903v1, 2020.

SHAWI, R.E.; MAHER, M.; SAKR, S. "Automated machine learning: State-of-the-art and open challenges", https://arxiv.org/abs/1906.02287, 2019.

SUGANUMA, M.; OZAY, M.; OKATANI, T. "Exploiting the Potential of Standard Convolutional Autoencoders for Image Restoration by Evolutionary Search", Proceedings of the 35th International Conference on Machine Learning, pp. 1-10, 2018.

TAN, M.; LE, Q.V. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", arXiv:1905.11946v5, 2020.

TOUVRON, H.; VEDALDI, A.; DOUZE, M.; JÉGOU, H. "Fixing the Train-Test Resolution Discrepancy: FixEfficientNet", arXiv:2003.08237v5, 2020.

TOUVRON, H.; VEDALDI, A.; DOUZE, M.; JÉGOU, H. "Fixing the train-test resolution discrepancy," Advances in Neural Information Processing Systems (NIPS), 2019.

WISTUBA, M.; RAWAT, A.; PEDAPATI, T. "A Survey on Neural Architecture Search", arXiv:1905.01392v2, 2019.

YANG, L.; SHAMI, A. "On hyperparameter optimization of machine learning algorithms: Theory and practice", Neurocomputing, vol. 415, pp. 295-316, 2020.

ZÖLLER, M.-A.; HUBER, M.F. "Benchmark and Survey of Automated Machine Learning Frameworks", https://arxiv.org/abs/1904.12054, 2019.

ZOPH, B.; VASUDEVAN, V.; SHLENS, J.; LE, Q.V. "Learning transferable architectures for scalable image recognition", arXiv:1707.07012v4, 2018.