电子科技大学
University of Electronic Science and Technology of China

Some Advances of Neural Network Architecture in Image Classification

Wei Han

Data Mining Lab,
Big Data Research Center, UESTC
Email：weihan@std.uestc.edu.cn

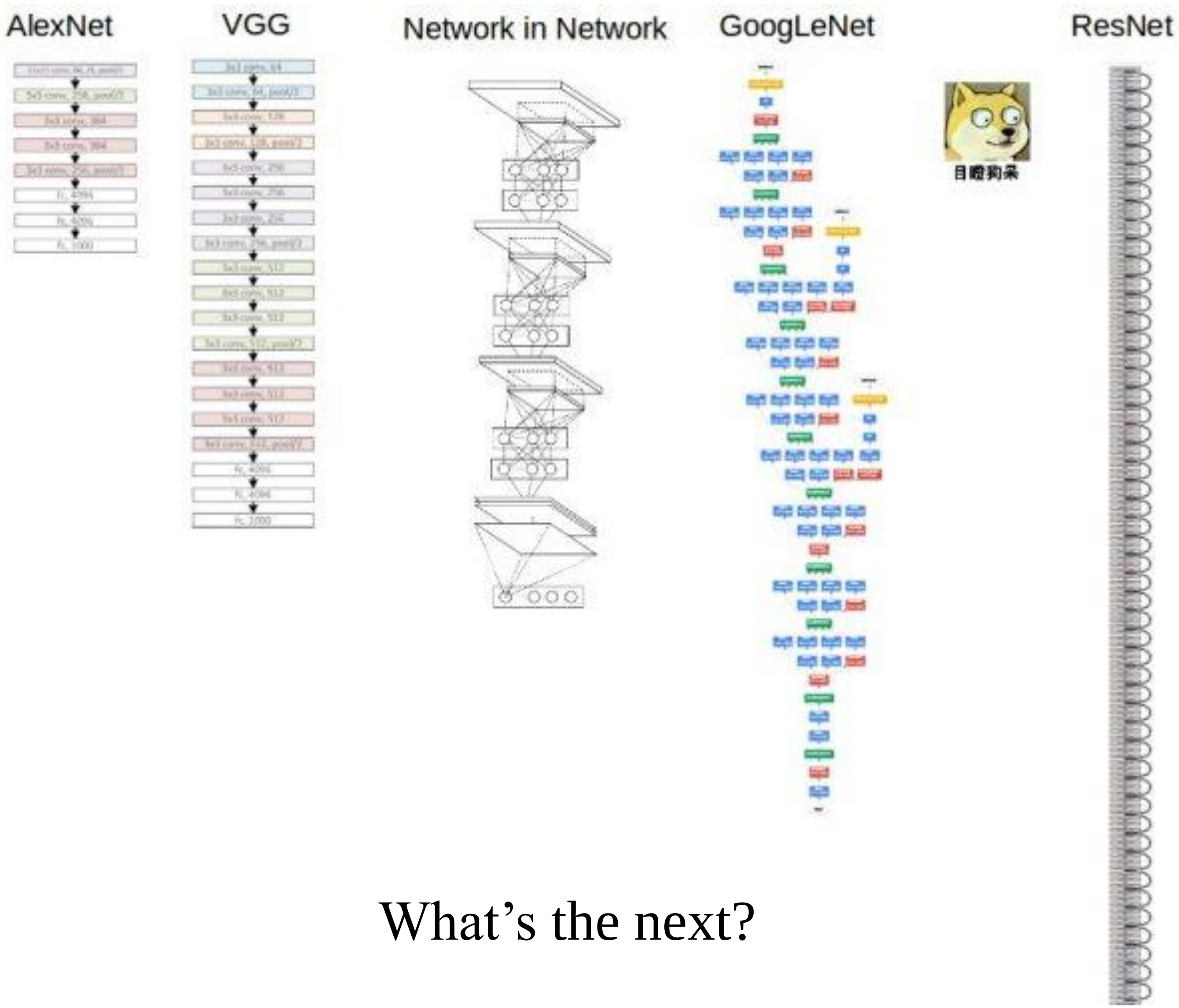
Outline

- **Manual Designed**
  - Inception
  - DenseNet
  - DPN
  - ResNeXt
  - SENet
  - EfficientNet
  - Fixing Resolution Discrepancy

- **NAS**
  - NASNet
  - PNASNet
  - AmoebaNet
  - DARTS

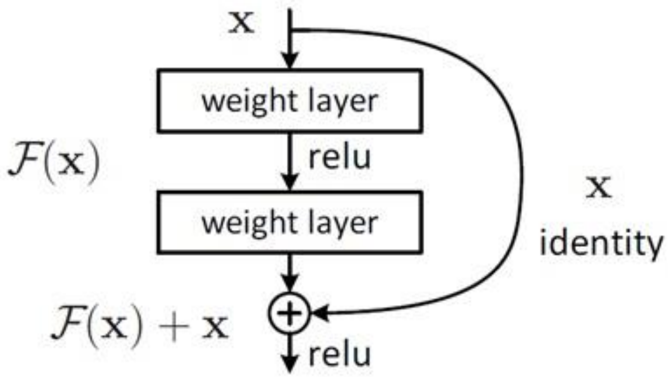
Background

Development of DNN:



What’s the next?

- Architecture



- Why it works?

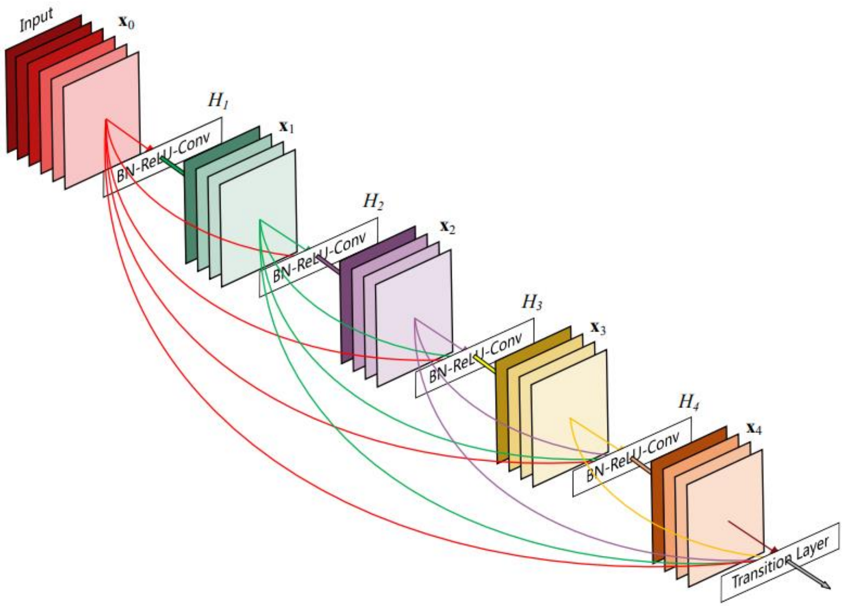
$$x_L = x_l + \sum_{i=l}^{L-1} F(x_i, W_i)$$

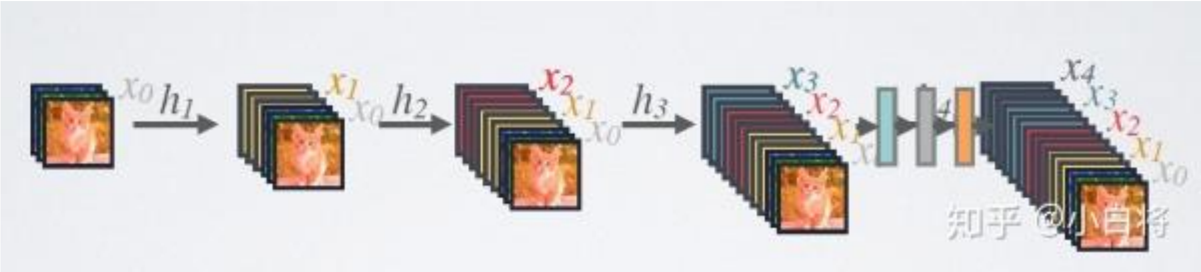
$$\frac{\partial loss}{\partial x_l} = \frac{\partial loss}{\partial x_L} \cdot \frac{\partial x_L}{\partial x_l} = \frac{\partial loss}{\partial x_L} \cdot \left(1 + \frac{\partial}{\partial x_L} \sum_{i=l}^{L-1} F(x_i, W_i)\right)$$

DenseNet

数据挖掘实验室
Data Mining Lab

- Feature reuse

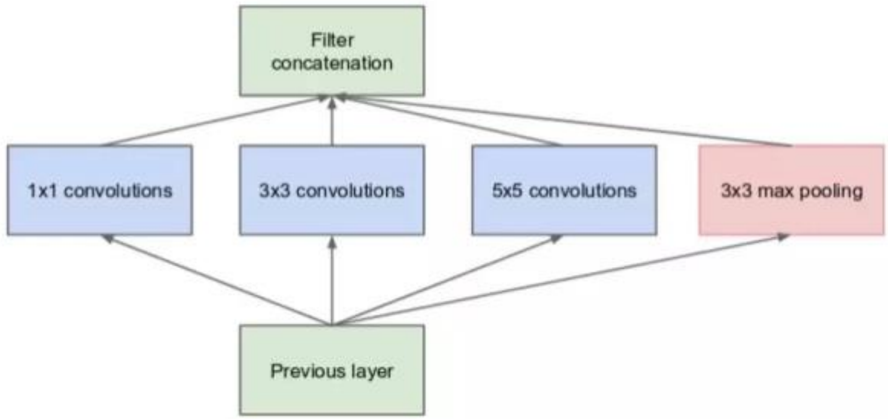




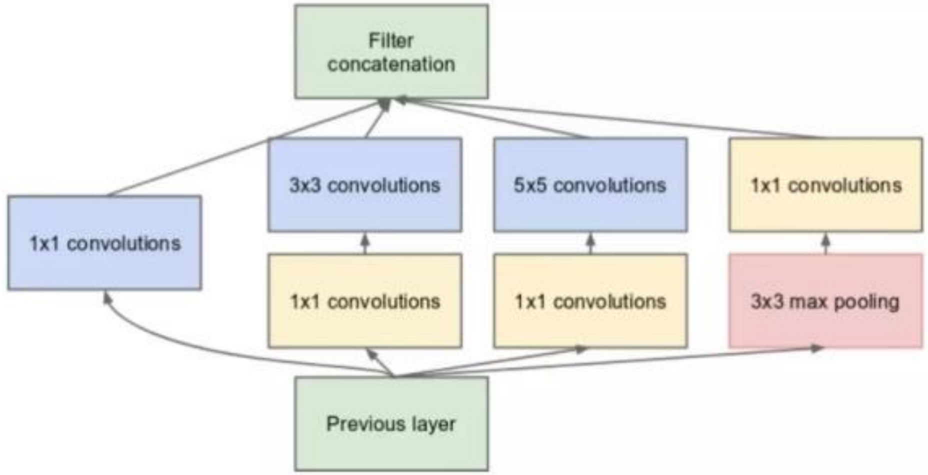
- Bound

Inception

- Inception v1 (GoogLeNet)



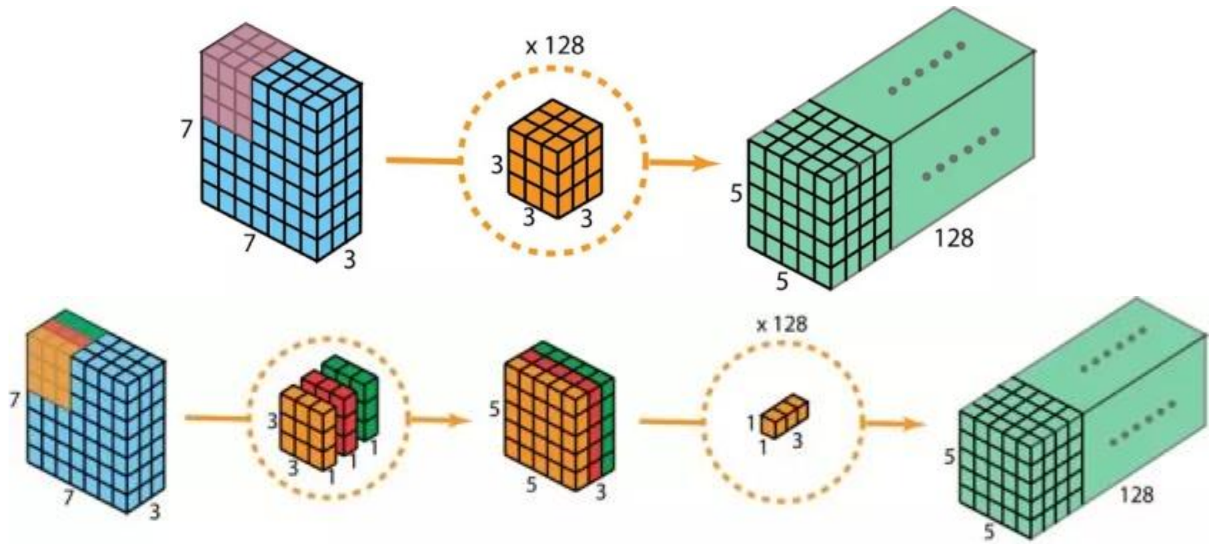
(a) Inception module, naïve version



(b) Inception module with dimension reductions

Inception

- 1*1 convolution



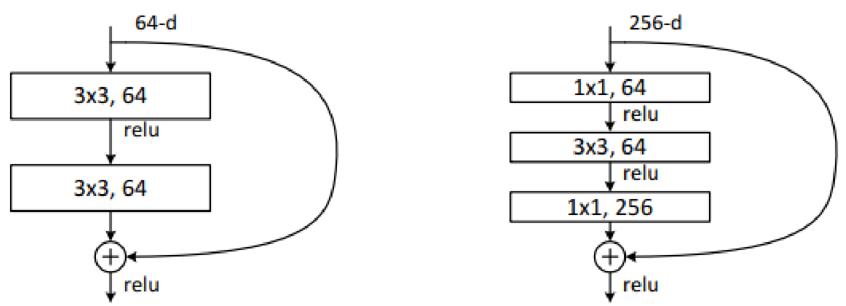
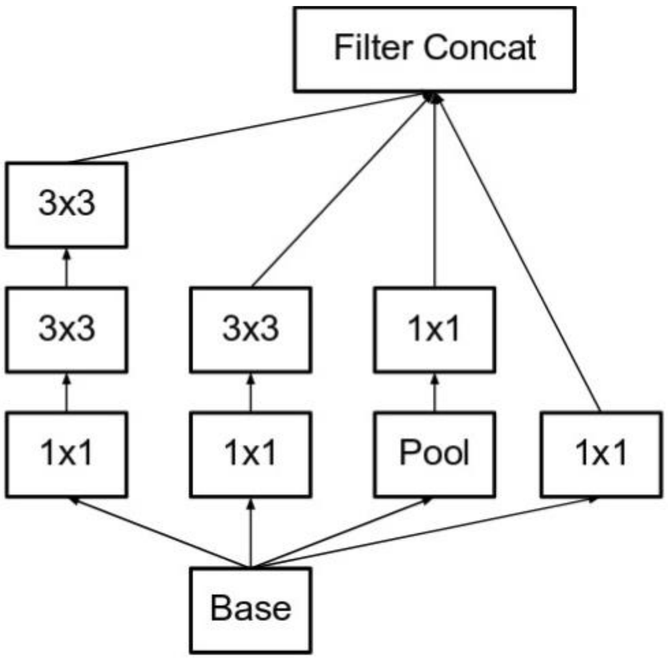
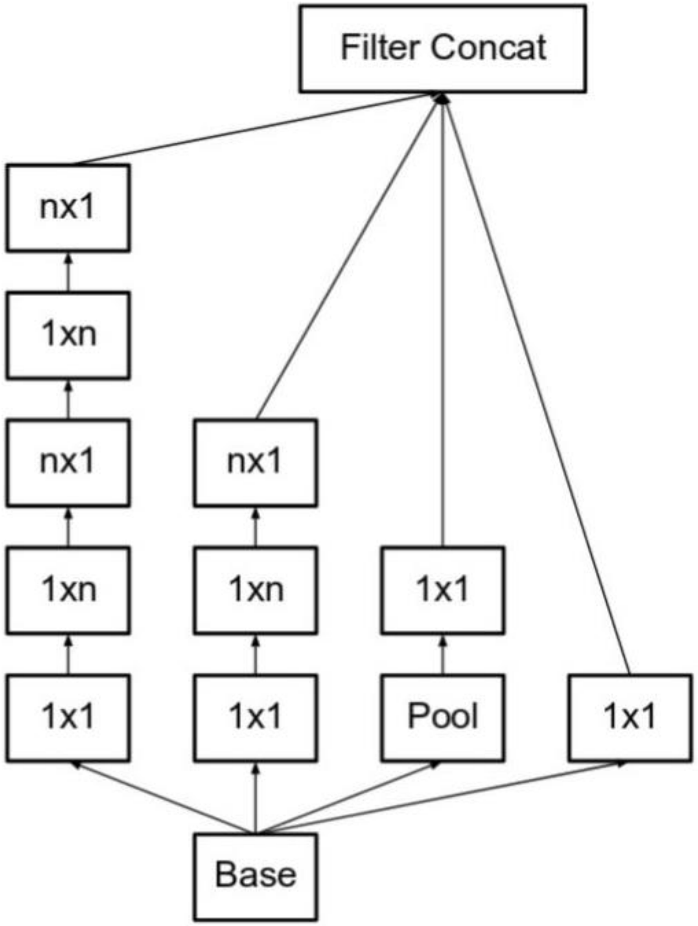


Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

Inception

- Inception v2 & v3





Inception

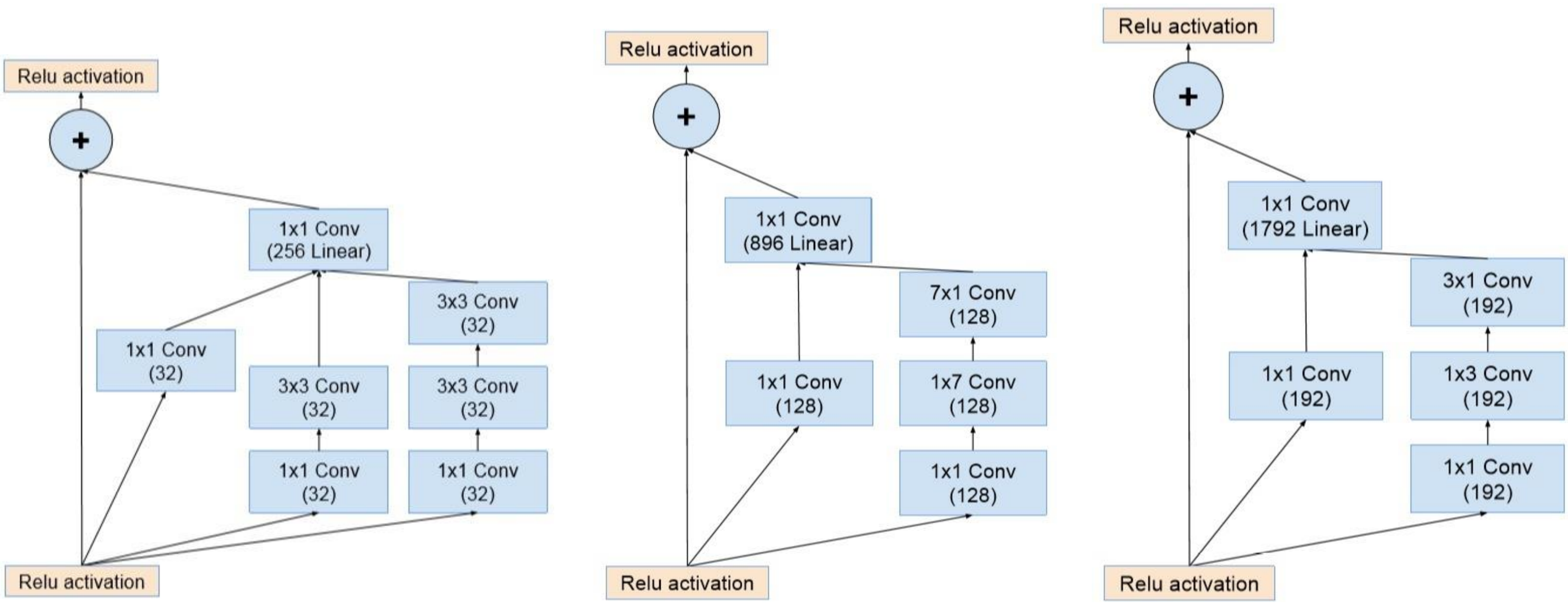
- Inception v2 & v3

  - **Inception Net v3** incorporated all of the above upgrades stated for Inception v2, and in addition used the following:

1. RMSProp Optimizer.
2. Factorized 7x7 convolutions.
3. BatchNorm in the Auxillary Classifiers.
4. Label Smoothing (A type of regularizing component added to the loss formula that prevents the network from becoming too confident about a class. Prevents over fitting).

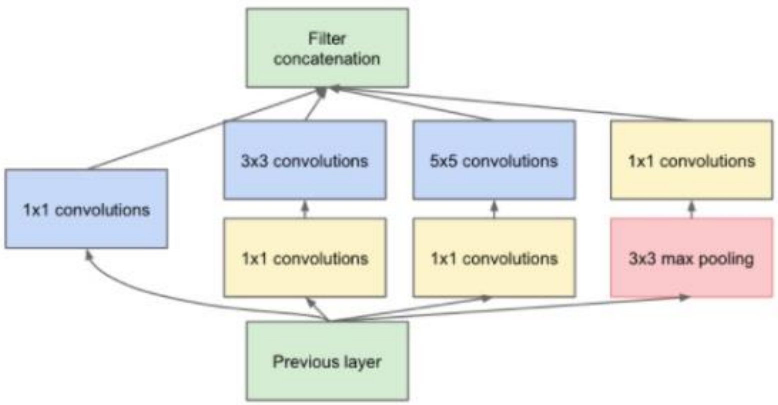
Inception

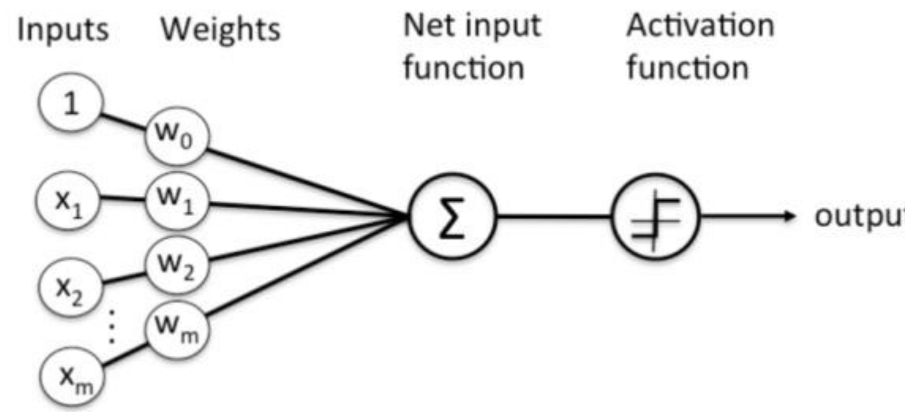
- Inception v4



ResNeXt

- Split-transform-merge

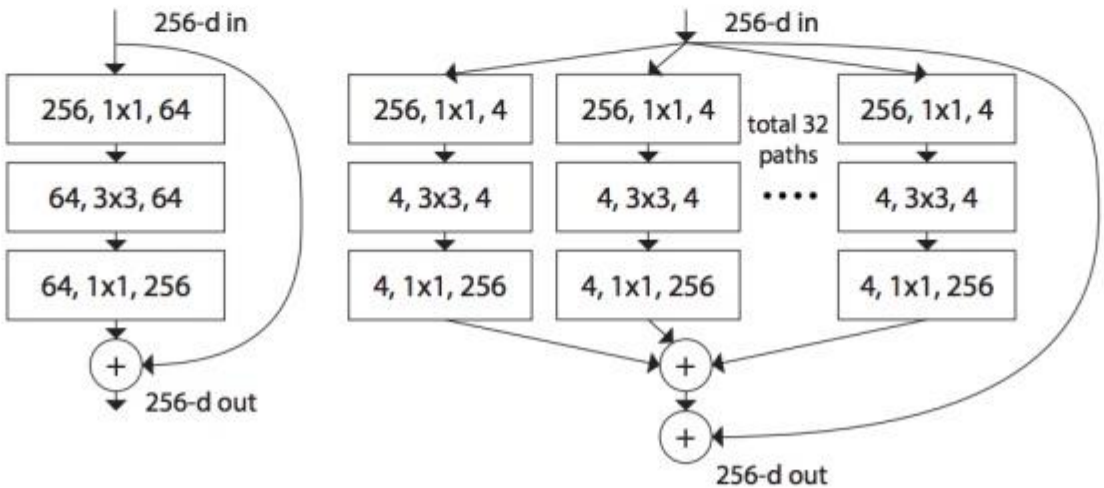




$$\mathcal{F}(\mathbf{x}) = \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x})$$

ResNeXt

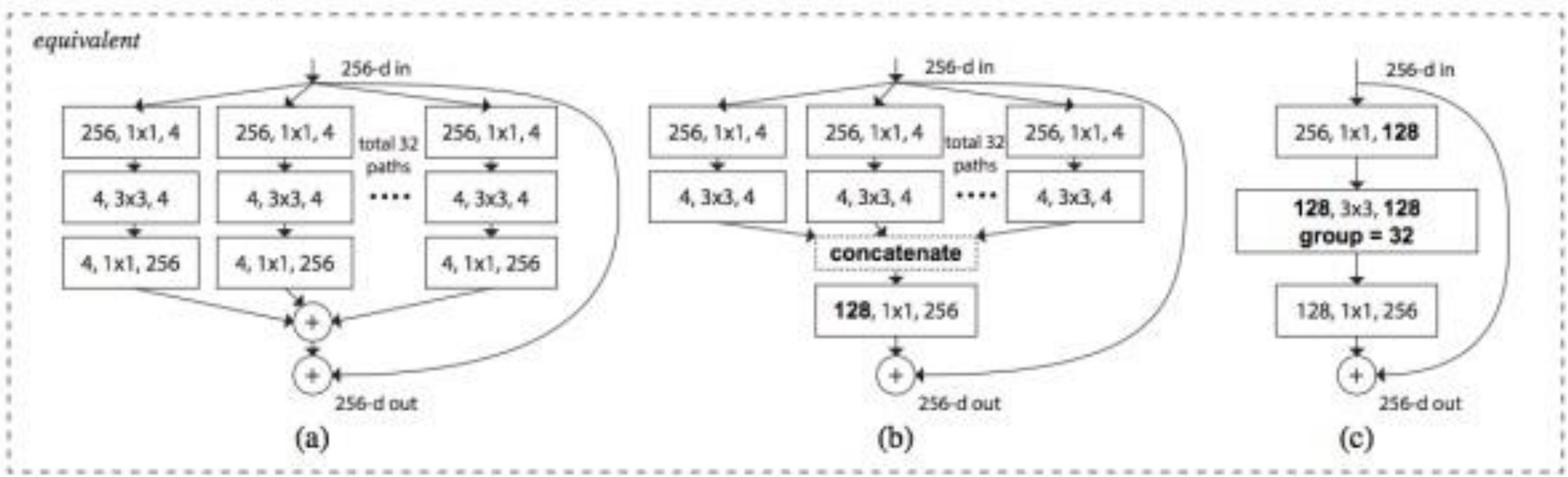
- Split-transform-merge



$$\mathbf{y} = \mathbf{x} + \sum_{i=1}^{C} \mathcal{T}_i(\mathbf{x})$$

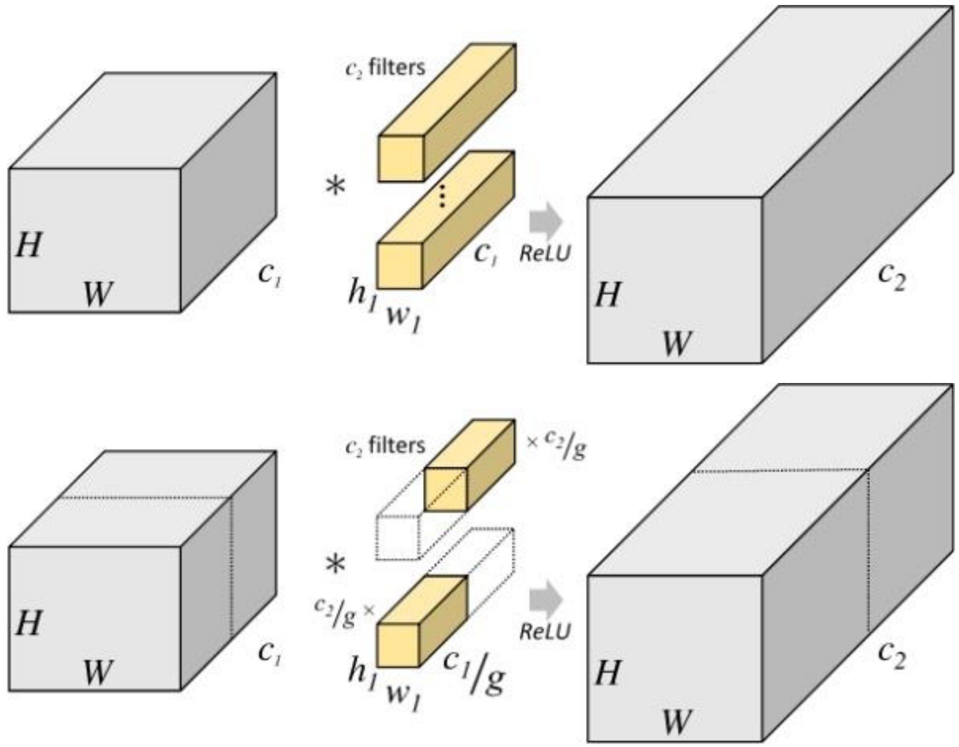
ResNeXt

- split-transform-merge



ResNeXt

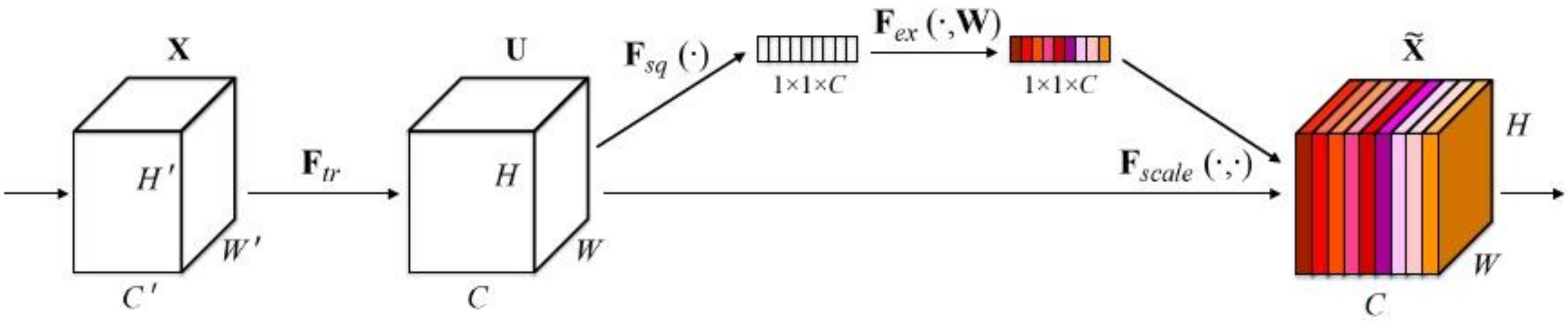
- Group convolution



- What helps？
  - Fewer parameters
  - Localization

SEnet

- Squeeze-and-Excitation Networks

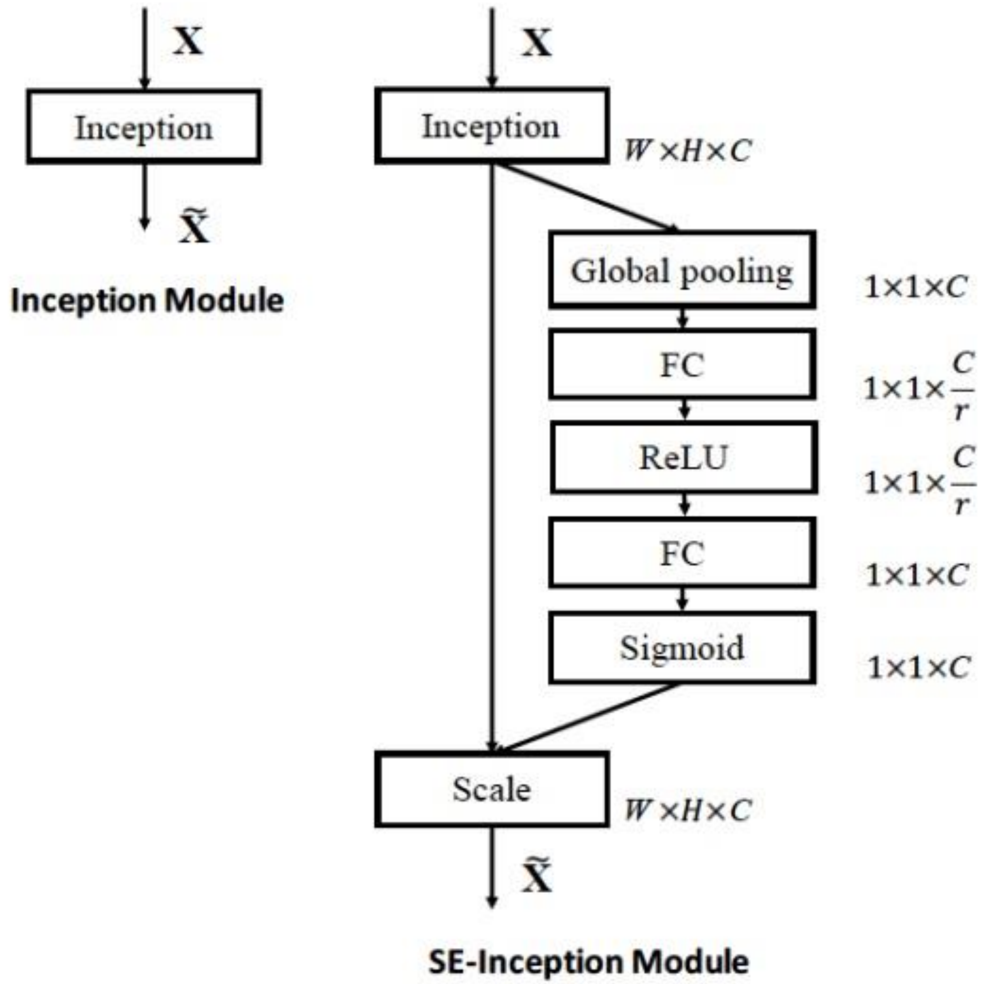


- Channel Attention

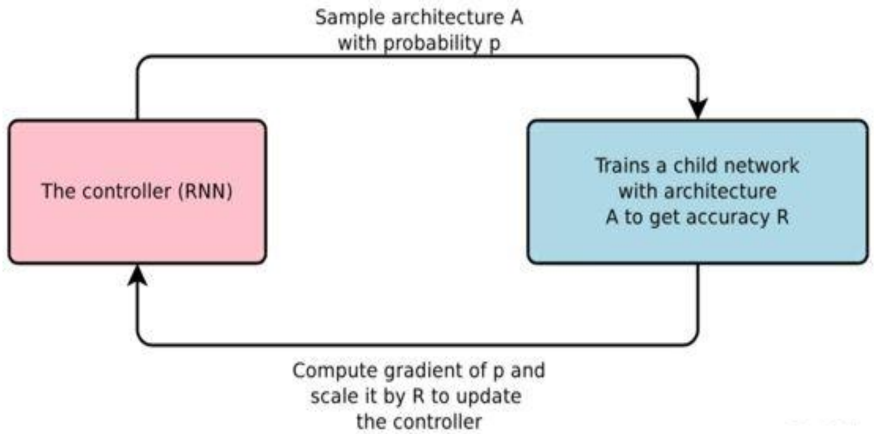
SENet

数据挖掘实验室
Data Mining Lab

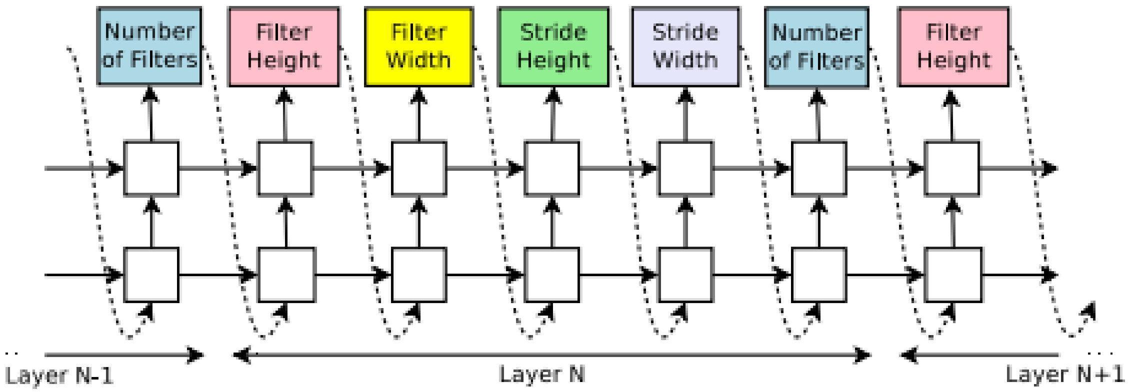
- Squeeze-and-Excitation Networks



- Network Architecture Search

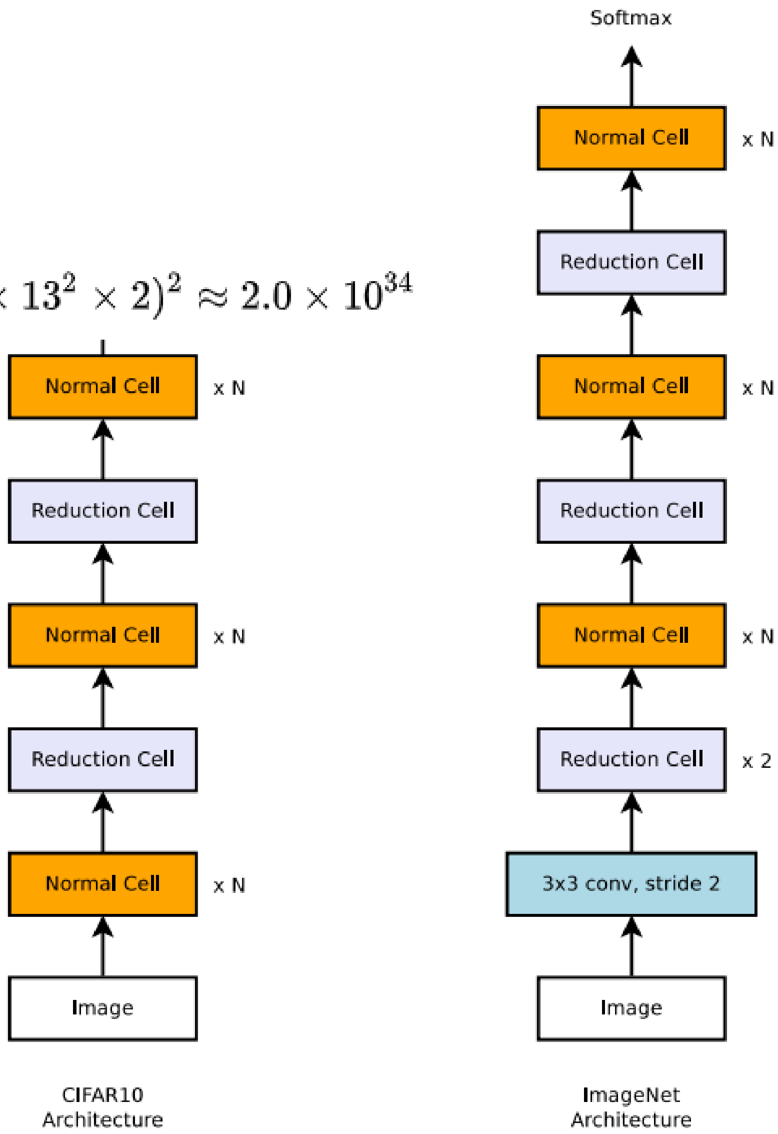


$$J(\theta_c) = E_{P(a_{1:T};\theta_c)}[R]$$



NASNet

- Stacking cells
  - Normal cell
  - Reduction cell
- $(2^2 \times 13^2 \times 3^2 \times 13^2 \times 4^2 \times 13^2 \times 5^2 \times 13^2 \times 6^2 \times 13^2 \times 2)^2 \approx 2.0 \times 10^{34}$
  - identity
  - 1x7 then 7x1 convolution
  - 3x3 average pooling
  - 5x5 max pooling
  - 1x1 convolution
  - 3x3 depthwise-separable conv
  - 7x7 depthwise-separable conv
  - 1x3 then 3x1 convolution
  - 3x3 dilated convolution
  - 3x3 max pooling
  - 7x7 max pooling
  - 3x3 convolution
  - 5x5 depthwise-seperable conv



PNASNet

数据挖掘实验室
Data Mining Lab

- Smaller search space
  - Only normal cell
  -
  -
  - 2^2
- Se
  -
- Su
  - RNN

Algorithm 1 Progressive Neural Architecture Search (PNAS).

Inputs: B (max num blocks), E (max num epochs), F (num filters in first layer), K (beam size), N (num times to unroll cell), trainSet, valSet.

$\mathcal{S}_1 = \mathcal{B}_1$ // *Set of candidate structures with one block*
$\mathcal{M}_1 =$ cell-to-CNN($\mathcal{S}_1$, N, F) // *Construct CNNs from cell specifications*
$\mathcal{C}_1 =$ train-CNN($\mathcal{M}_1$, E, trainSet) // *Train proxy CNNs*
$\mathcal{A}_1 =$ eval-CNN($\mathcal{C}_1$, valSet) // *Validation accuracies*
$\pi =$ fit($\mathcal{S}_1$, $\mathcal{A}_1$) // *Train the reward predictor from scratch*
for $b = 2 : B$ **do**
  $\mathcal{S}'_b =$ expand-cell($\mathcal{S}_{b-1}$) // *Expand current candidate cells by one more block*
  $\hat{\mathcal{A}}'_b =$ predict($\mathcal{S}'_b$, π) // *Predict accuracies using reward predictor*
  $\mathcal{S}_b =$ top-K($\mathcal{S}'_b$, $\hat{\mathcal{A}}'_b$, K) // *Most promising cells according to prediction*
  $\mathcal{M}_b =$ cell-to-CNN($\mathcal{S}_b$, N, F)
  $\mathcal{C}_b =$ train-CNN($\mathcal{M}_b$, E, trainSet)
  $\mathcal{A}_b =$ eval-CNN($\mathcal{C}_b$, valSet)
  $\pi =$ update-predictor($\mathcal{S}_b$, $\mathcal{A}_b$, π) // *Finetune reward predictor with new data*
end for
Return top-K($\mathcal{S}_B$, $\mathcal{A}_B$, 1)

- Basic parts

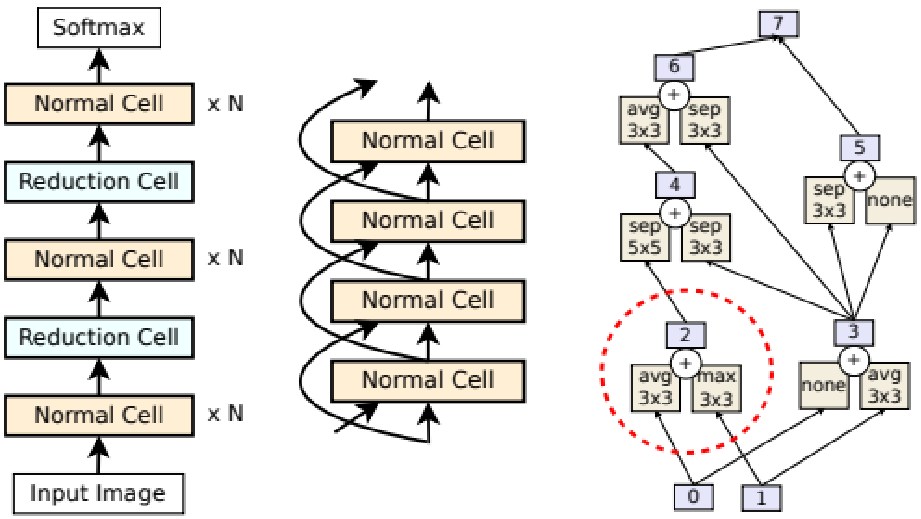


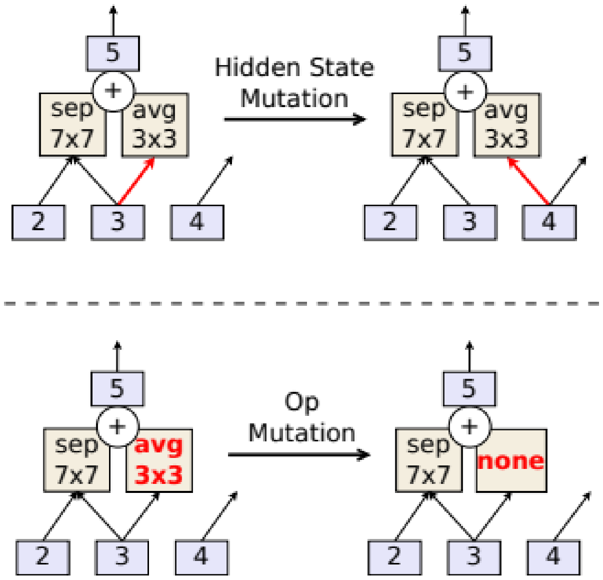
Figure 1: NASNet Search Space [54]. LEFT: the full outer structure (omitting skip inputs for clarity). MIDDLE: detailed view with the skip inputs. RIGHT: cell example. Dotted line demarcates a pairwise combination.

AmoebaNet

- Aging Evolution

Algorithm 1 Aging Evolution

```
population ← empty queue                              ▷ The population.
history ← ∅                                           ▷ Will contain all models.
while |population| < P do                             ▷ Initialize population.
    model.arch ← RANDOMARCHITECTURE()
    model.accuracy ← TRAINANDEVAL(model.arch)
    add model to right of population
    add model to history
end while
while |history| < C do                                ▷ Evolve for C cycles.
    sample ← ∅                                        ▷ Parent candidates.
    while |sample| < S do
        candidate ← random element from population
                                ▷ The element stays in the population.
        add candidate to sample
    end while
    parent ← highest-accuracy model in sample
    child.arch ← MUTATE(parent.arch)
    child.accuracy ← TRAINANDEVAL(child.arch)
    add child to right of population
    add child to history
    remove dead from left of population               ▷ Oldest.
    discard dead
end while
return highest-accuracy model in history
```



- Stacking cells

$$\min_{\alpha} \mathcal{L}_{val}\left(w^{*}(\alpha), \alpha\right) \qquad (3)$$

$$s.t. \quad w^{*}(\alpha) = \operatorname{argmin}_{w} \mathcal{L}_{\text{train}}(w, \alpha) \qquad (4)$$

$$\nabla_{\alpha} \mathcal{L}_{val}\left(w^{*}(\alpha), \alpha\right) \qquad (5)$$

$$\approx \nabla_{\alpha} \mathcal{L}_{val}\left(w - \xi \nabla_{w} \mathcal{L}_{\text{train}}(w, \alpha), \alpha\right) \qquad (6)$$

$$\nabla_{\alpha} \mathcal{L}_{val}\left(\omega - \xi \nabla_{\omega} \mathcal{L}_{train}(\omega, \alpha), \alpha\right)$$

$$= \nabla_{\alpha} \mathcal{L}_{val}(\omega', \alpha) - \xi \nabla^{2}_{\alpha,\omega} \mathcal{L}_{train}(\omega, \alpha) \cdot \nabla_{\omega'} \mathcal{L}_{val}(\omega', \alpha)$$

$$\nabla^{2}_{\alpha,\omega} \mathcal{L}_{train}(\omega, \alpha) \cdot \nabla_{\omega'} \mathcal{L}_{val}(\omega', \alpha) \approx \frac{\nabla_{\alpha} \mathcal{L}_{train}(\omega^{+}, \alpha) - \nabla_{\alpha} \mathcal{L}_{train}(\omega^{-}, \alpha)}{2\epsilon}$$

$$f(x_0 + hA) = f(x_0) + \frac{f'(x_0)}{1!} hA + \ldots$$

$$f(x_0 - hA) = f(x_0) - \frac{f'(x_0)}{1!} hA + \ldots$$

DARTS

- Stacking cells

$$\nabla^2_{\alpha,\omega}\mathcal{L}_{train}(\omega,\alpha)\cdot\nabla_{\omega'}\mathcal{L}_{val}(\omega',\alpha)\approx\frac{\nabla_\alpha\mathcal{L}_{train}(\omega^+,\alpha)-\nabla_\alpha\mathcal{L}_{train}(\omega^-,\alpha)}{2\epsilon}$$

$$f(x_0+hA)=f(x_0)+\frac{f'(x_0)}{1!}hA+\ldots$$
$$f(x_0-hA)=f(x_0)-\frac{f'(x_0)}{1!}hA+\ldots$$

$$f'(x_0)\cdot A\approx\frac{f(x_0+hA)-f(x_0-hA)}{2h}$$

- Stacking cells

Algorithm 1: DARTS – Differentiable Architecture Search

Create a mixed operation $\bar{o}^{(i,j)}$ parametrized by $\alpha^{(i,j)}$ for each edge (i, j)
while *not converged* **do**
1. Update architecture α by descending $\nabla_\alpha \mathcal{L}_{val}(w - \xi \nabla_w \mathcal{L}_{train}(w, \alpha), \alpha)$
   ($\xi = 0$ if using first-order approximation)
2. Update weights w by descending $\nabla_w \mathcal{L}_{train}(w, \alpha)$

Derive the final architecture based on the learned α.

Table 3: Comparison with state-of-the-art image classifiers on ImageNet in the mobile setting.

Architecture	Test Error (%) top-1	Test Error (%) top-5	Params (M)	$+\times$ (M)	Search Cost (GPU days)	Search Method
Inception-v1 (Szegedy et al., 2015)	30.2	10.1	6.6	1448	–	manual
MobileNet (Howard et al., 2017)	29.4	10.5	4.2	569	–	manual
ShuffleNet 2× ($g = 3$) (Zhang et al., 2017)	26.3	–	∼5	524	–	manual
NASNet-A (Zoph et al., 2018)	26.0	8.4	5.3	564	2000	RL
NASNet-B (Zoph et al., 2018)	27.2	8.7	5.3	488	2000	RL
NASNet-C (Zoph et al., 2018)	27.5	9.0	4.9	558	2000	RL
AmoebaNet-A (Real et al., 2018)	25.5	8.0	5.1	555	3150	evolution
AmoebaNet-B (Real et al., 2018)	26.0	8.5	5.3	555	3150	evolution
AmoebaNet-C (Real et al., 2018)	24.3	7.6	6.4	570	3150	evolution
PNAS (Liu et al., 2018a)	25.8	8.1	5.1	588	∼225	SMBO
DARTS (searched on CIFAR-10)	26.7	8.7	4.7	574	4	gradient-based

EfficientNet

数据挖掘实验室
Data Mining Lab

- Scaling

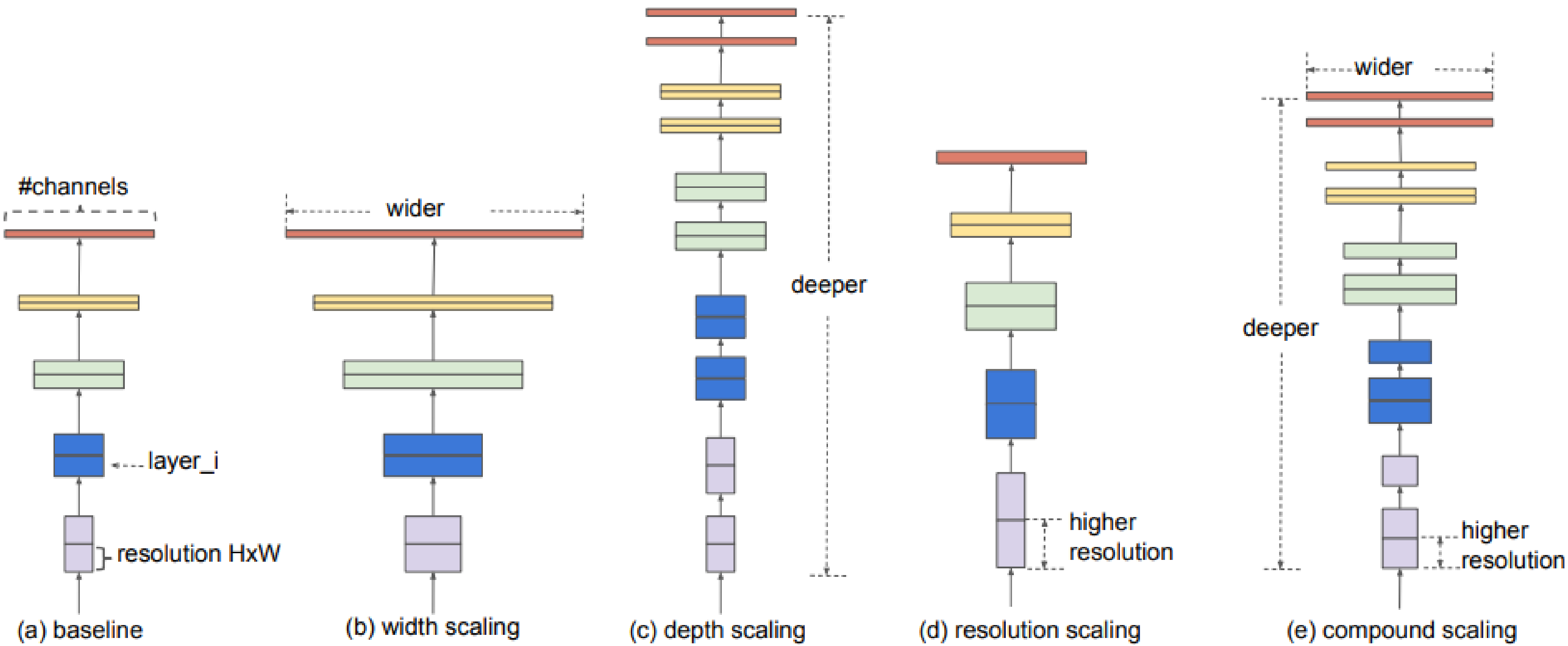


Figure 2. **Model Scaling.** (a) is a baseline network example; (b)-(d) are conventional scaling that only increases one dimension of network width, depth, or resolution. (e) is our proposed compound scaling method that uniformly scales all three dimensions with a fixed ratio.

- MNAS :)

$$
\begin{aligned}
\max_{d,w,r} \quad & Accuracy\big(\mathcal{N}(d,w,r)\big) \\
s.t. \quad & \mathcal{N}(d,w,r) = \bigodot_{i=1\ldots s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\big(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\big) \\
& \text{Memory}(\mathcal{N}) \leq \text{target\_memory} \\
& \text{FLOPS}(\mathcal{N}) \leq \text{target\_flops}
\end{aligned}
$$

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

$$\alpha = 1.2，\ \beta = 1.1，\ \gamma = 1.15$$

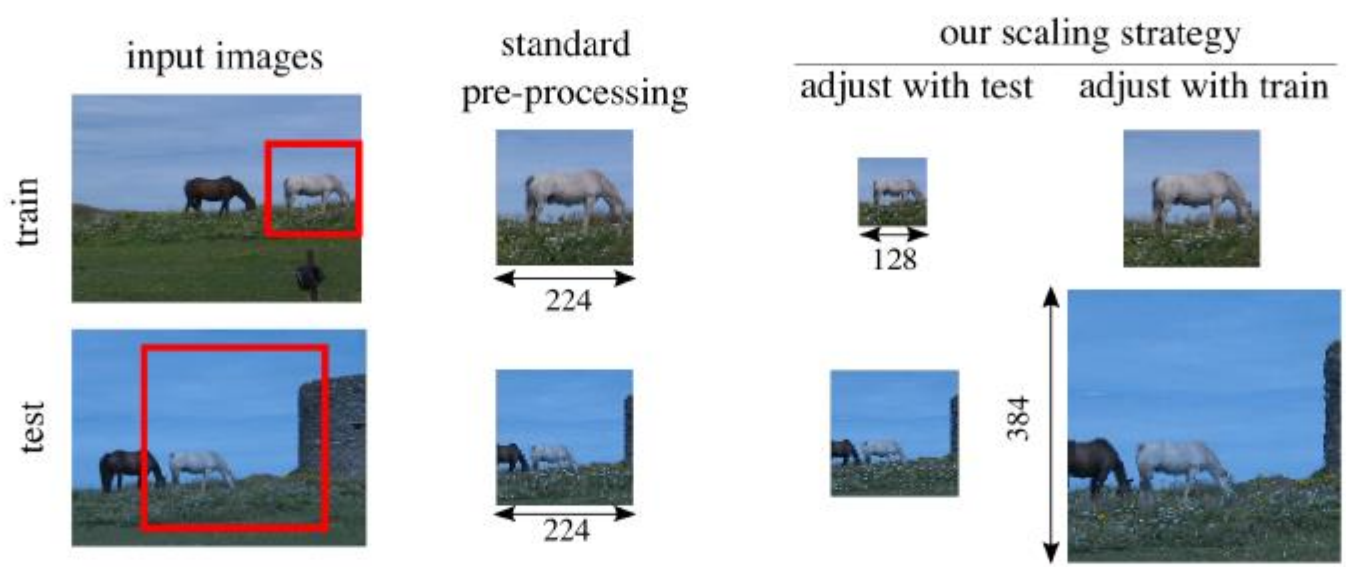
EfficientNet

Model	Top-1 Acc.	Top-5 Acc.	#Params	Ratio-to-EfficientNet	#FLOPS	Ratio-to-EfficientNet
EfficientNet-B0	**76.3%**	**93.2%**	**5.3M**	**1x**	**0.39B**	**1x**
ResNet-50 (He et al., 2016)	76.0%	93.0%	26M	4.9x	4.1B	11x
DenseNet-169 (Huang et al., 2017)	76.2%	93.2%	14M	2.6x	3.5B	8.9x
EfficientNet-B1	**78.8%**	**94.4%**	**7.8M**	**1x**	**0.70B**	**1x**
ResNet-152 (He et al., 2016)	77.8%	93.8%	60M	7.6x	11B	16x
DenseNet-264 (Huang et al., 2017)	77.9%	93.9%	34M	4.3x	6.0B	8.6x
Inception-v3 (Szegedy et al., 2016)	78.8%	94.4%	24M	3.0x	5.7B	8.1x
Xception (Chollet, 2017)	79.0%	94.5%	23M	3.0x	8.4B	12x
EfficientNet-B2	**79.8%**	**94.9%**	**9.2M**	**1x**	**1.0B**	**1x**
Inception-v4 (Szegedy et al., 2017)	80.0%	95.0%	48M	5.2x	13B	13x
Inception-resnet-v2 (Szegedy et al., 2017)	80.1%	95.1%	56M	6.1x	13B	13x
EfficientNet-B3	**81.1%**	**95.5%**	**12M**	**1x**	**1.8B**	**1x**
ResNeXt-101 (Xie et al., 2017)	80.9%	95.6%	84M	7.0x	32B	18x
PolyNet (Zhang et al., 2017)	81.3%	95.8%	92M	7.7x	35B	19x
EfficientNet-B4	**82.6%**	**96.3%**	**19M**	**1x**	**4.2B**	**1x**
SENet (Hu et al., 2018)	82.7%	96.2%	146M	7.7x	42B	10x
NASNet-A (Zoph et al., 2018)	82.7%	96.2%	89M	4.7x	24B	5.7x
AmoebaNet-A (Real et al., 2019)	82.8%	96.1%	87M	4.6x	23B	5.5x
PNASNet (Liu et al., 2018)	82.9%	96.2%	86M	4.5x	23B	6.0x
EfficientNet-B5	**83.3%**	**96.7%**	**30M**	**1x**	**9.9B**	**1x**
AmoebaNet-C (Cubuk et al., 2019)	83.5%	96.5%	155M	5.2x	41B	4.1x
EfficientNet-B6	**84.0%**	**96.9%**	**43M**	**1x**	**19B**	**1x**
EfficientNet-B7	**84.4%**	**97.1%**	**66M**	**1x**	**37B**	**1x**
GPipe (Huang et al., 2018)	84.3%	97.0%	557M	8.4x	-	-

We omit ensemble and multi-crop models (Hu et al., 2018), or models pretrained on 3.5B Instagram images (Mahajan et al., 2018).

Fixing Resolution Discrepancy

- Re-scaling



Fixing Resolution Discrepancy

数据挖掘实验室
Data Mining Lab

Table 2: State of the art on ImageNet with ResNet-50 architectures and with all types of architecture (Single Crop evaluation)

Models	Extra Training Data	Train	Test	# Parameters	Top-1 (%)	Top-5 (%)
ResNet-50 Pytorch	_	224	224	25.6M	76.1	92.9
ResNet-50 mix up [44]	_	224	224	25.6M	77.7	94.4
ResNet-50 CutMix [43]	_	224	224	25.6M	78.4	94.1
ResNet-50-D [15]	_	224	224	25.6M	79.3	94.6
MultiGrain R50-AA-500 [5]	_	224	500	25.6M	79.4	94.8
ResNet-50 Billion-scale [42]	✓	224	224	25.6M	81.2	96.0
Our ResNet-50	_	224	384	25.6M	79.1	94.6
Our ResNet-50 CutMix	_	224	320	25.6M	79.8	94.9
Our ResNet-50 Billion-scale@160	✓	160	224	25.6M	81.9	96.1
Our ResNet-50 Billion-scale@224	✓	224	320	25.6M	82.5	96.6
PNASNet-5 (N = 4, F = 216) [24]	_	331	331	86.1M	82.9	96.2
MultiGrain PNASNet @ 500px [5]		331	500	86.1M	83.6	96.7
AmoebaNet-B (6,512) [17]	_	480	480	577M	84.3	97.0
EfficientNet-B7 [37]	_	600	600	66M	84.4	97.1
Our PNASNet-5	_	331	480	86.1M	83.7	96.8
ResNeXt-101 32x8d [25]	✓	224	224	88M	82.2	96.4
ResNeXt-101 32x16d [25]	✓	224	224	193M	84.2	97.2
ResNeXt-101 32x32d [25]	✓	224	224	466M	85.1	97.5
ResNeXt-101 32x48d [25]	✓	224	224	829M	85.4	97.6
Our ResNeXt-101 32x48d	✓	224	320	829M	**86.4**	**98.0**

Conclusion

- Go Deeper
- Go Wider
- Rethink the problem!

Thank you