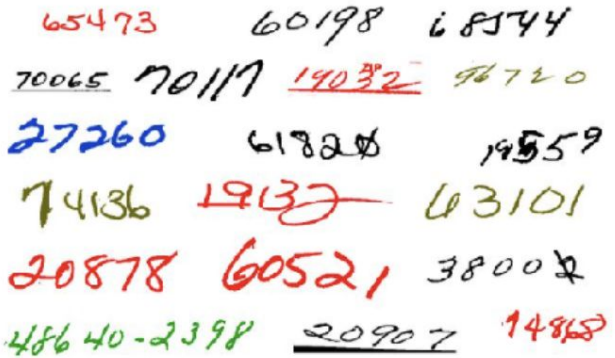
進階影像視覺模型

授課老師: 楊景明

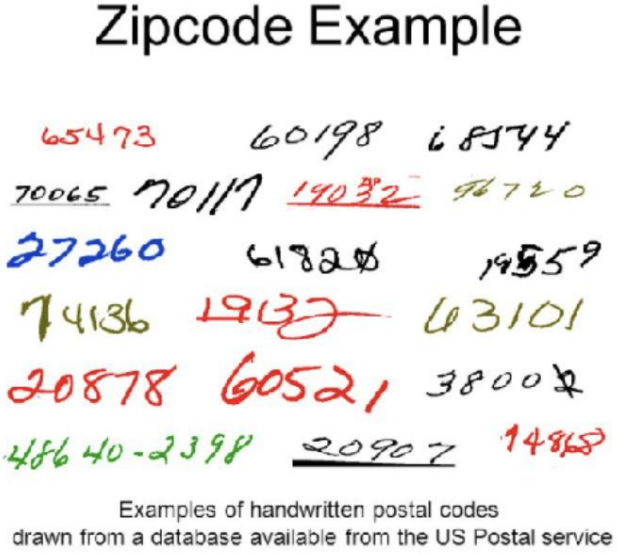
LeNet

Zipcode Example



Examples of handwritten postal codes
drawn from a database available from the US Postal service

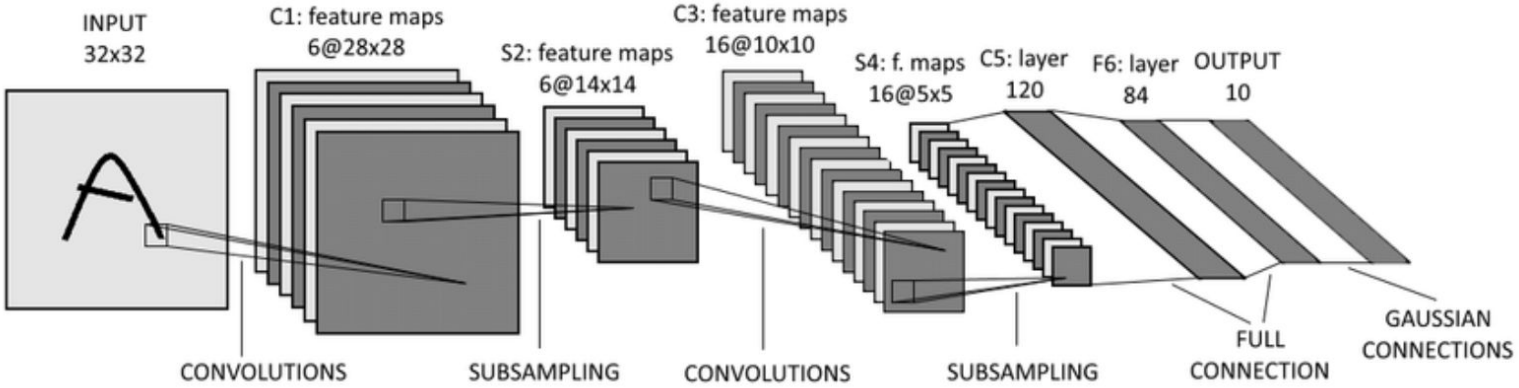
LeNet



- It was developed for handwritten digit recognition for US zip codes (MNIST)
- Introduced in 1995

Source: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

LeNet Architecture

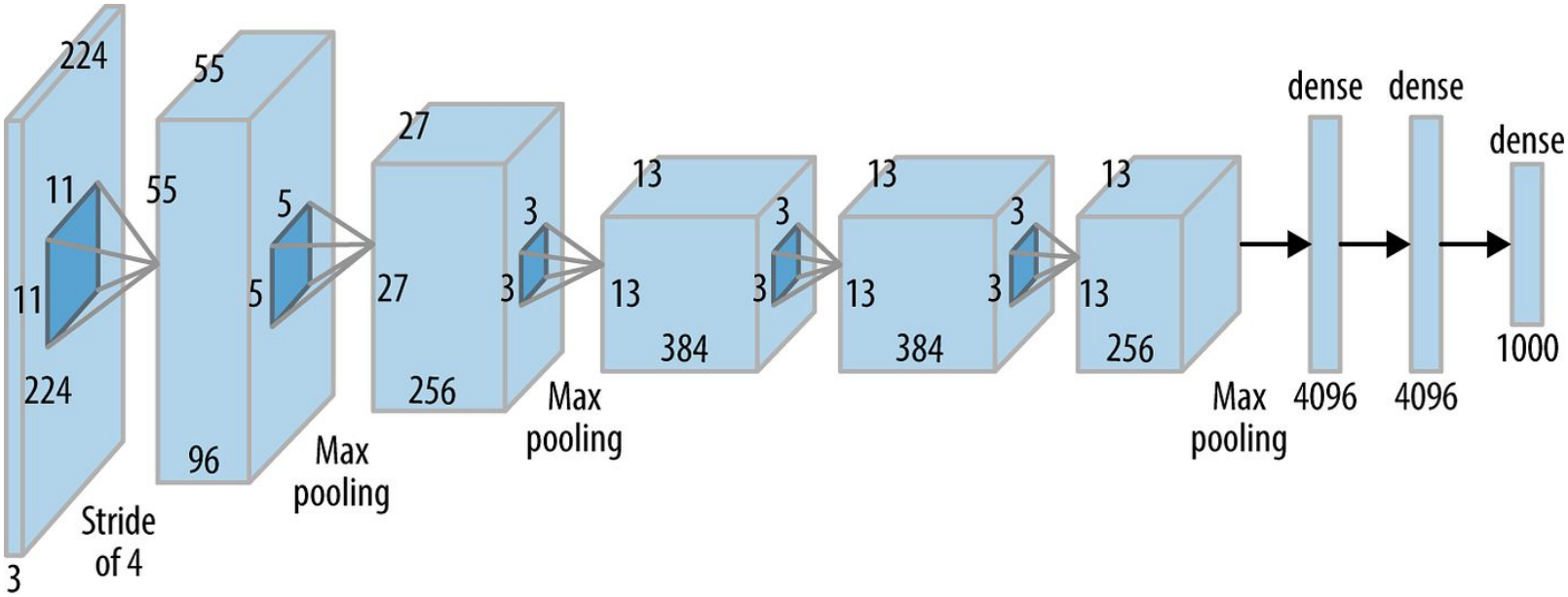


- 兩個 convolution layers 和 Max Pool (subsampling of 2x2 kernel and stride 2)
- 第一個 Conv layer 有 6 個大小為 5x5 的 filters/kernels (stride 1, padding 0)
- 第二個 Conv layer 有 16 個大小為 5x5 的 filters/kernels (stride 1, padding 0)
- 做完第二次 Max Pool 後, 把 5x5x16 layer 攤平成 400 個節點並連接到第一個有 120 個節點的 FC layer, 再接另一格有 84 個節點的 FC layer, 再接最後有 10 個節點的輸出層 (10 classes)
- LeNet 在 MNIST 資料上達到 99.3% Accuracy

AlexNet

- Introduced in 2012 (17 years after LeNet)
- Winner of ILSVRX in 2012
- 共有 8 層：前 5 層為Convolutional Layers，後 3 層為 FC Layers
- 超過 60 Million parameters 並使用兩個 GPU 訓練了超過一個禮拜

AlexNet Architecture

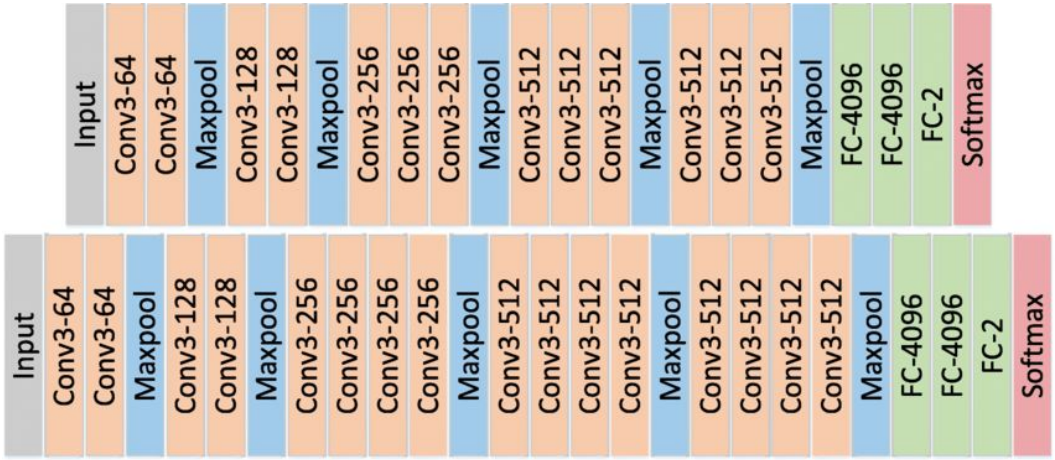


VGGNet

- Introduced in 2014
- Achieved 92.7% **top-5 Accuracy** in ImageNet (1000 classes)
- VGG16 has 13 Conv Layers with 3 FC Layers
- VGG19 has 16 Conv Layers with 3 FC Layers

VGGNet

- VGG 是按照我們現在認為的 “典型CNN” 結構所設計，也就是在多個 Conv Layers 後接 Pooling Layer
- Filters/Feature Maps 的數量逐漸增加，直到 FC Layers



Network structures of VGG16 (top) and VGG19 (bottom)

Source: https://www.researchgate.net/figure/Network-structures-of-VGG16-top-and-VGG19-bottom_fig9_343137490

龐大的參數量

- High accuracy, but very slow to train

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Source: https://arxiv.org/pdf/1409.1556.pdf

VGGNet 優缺點

優點:

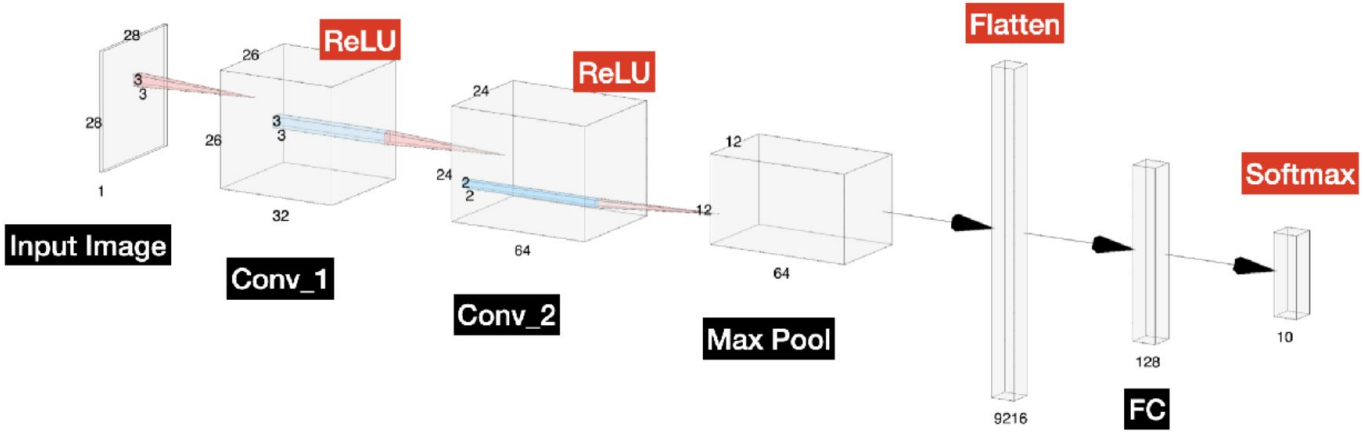
- “深度”網路設計
- 規範的結構

缺點:

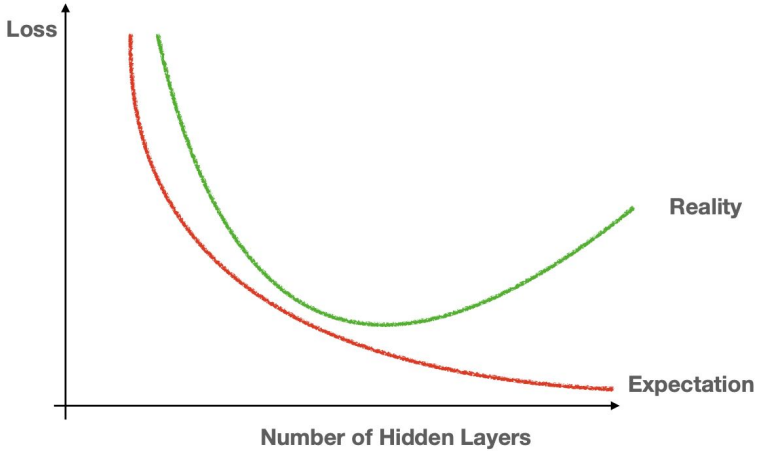
- 計算需求大
- 參數冗餘

ResNet

一般的CNN



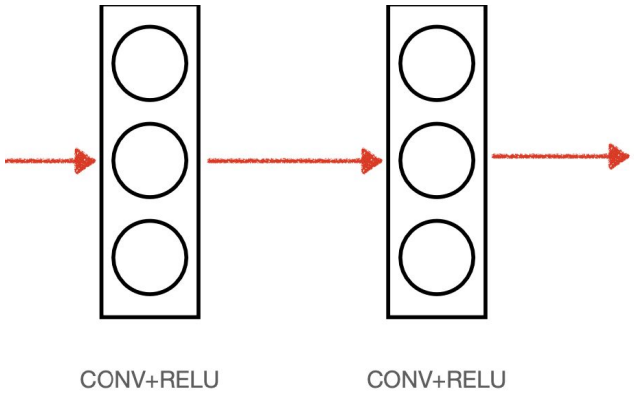
- 線性順序的序列
- 當模型很深時，效能會下降



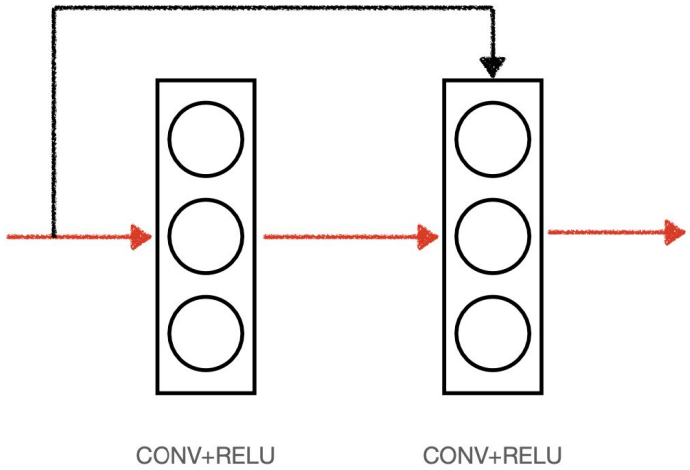
Exploding and Vanishing Gradients

- 在具有 N 層的深度網路中，必須將 N 個導數相乘才能執行梯度更新
- 如果導數很大，梯度會呈指數增長或“**爆炸**”
- 同樣，如果導數很小，它們就會呈指數下降或“**消失**”

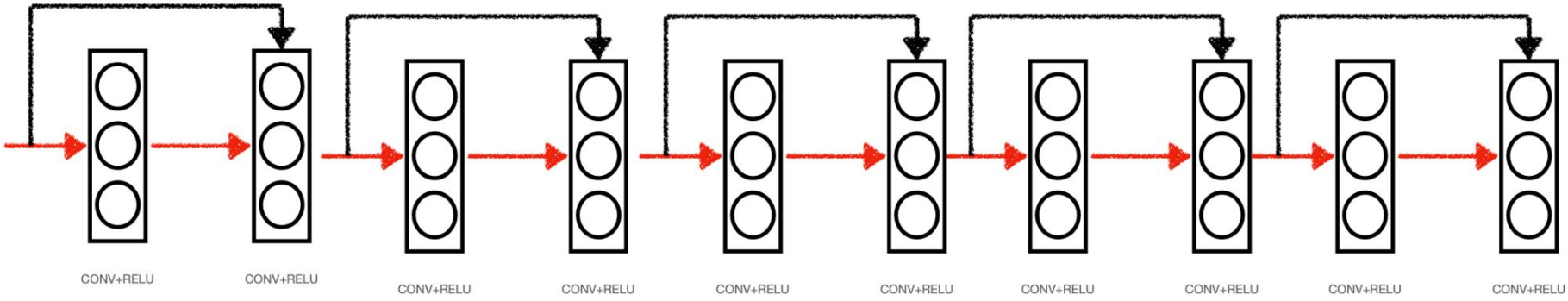
如何解决

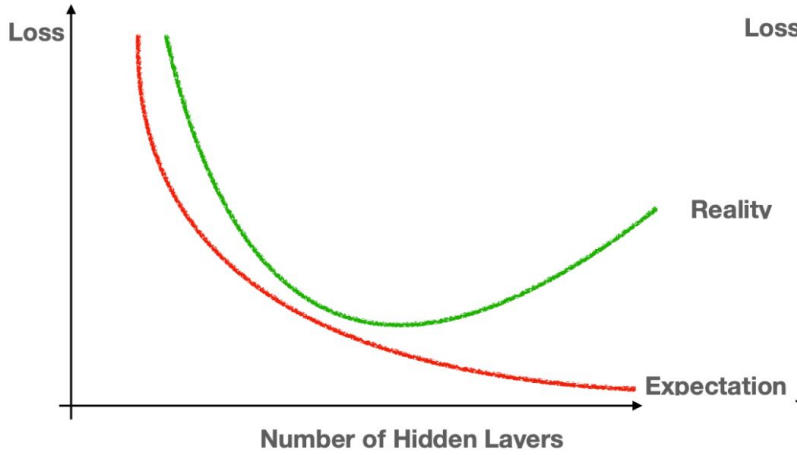


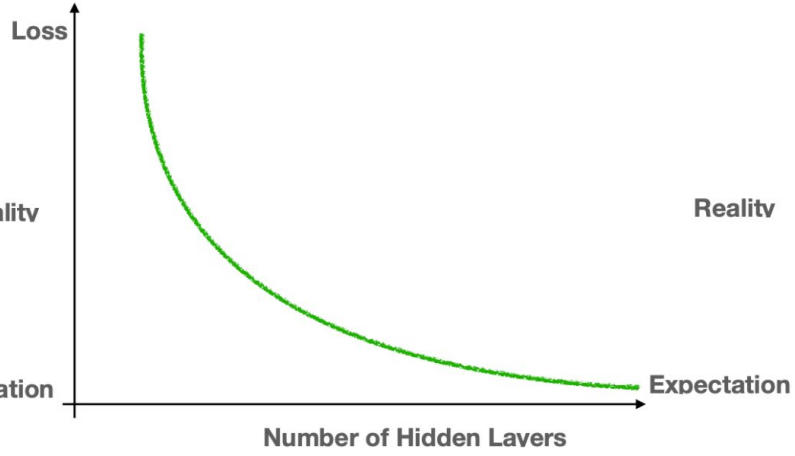
如何解決



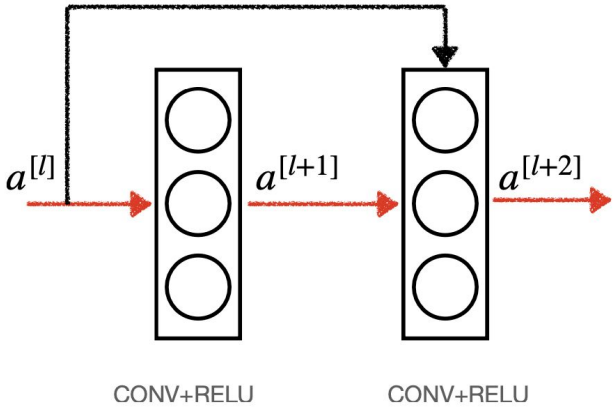
- 將前一層的輸入連接到前一層的輸出


CONV+RELU
CONV+RELU
CONV+RELU
CONV+RELU
CONV+RELU
CONV+RELU
CONV+RELU
CONV+RELU
CONV+RELU
CONV+RELU


Loss
Realitv
Expectation
Number of Hidden Lavers


Loss
Realitv
Expectation
Number of Hidden Lavers

Why Does ResNets Work? (Mathematics)



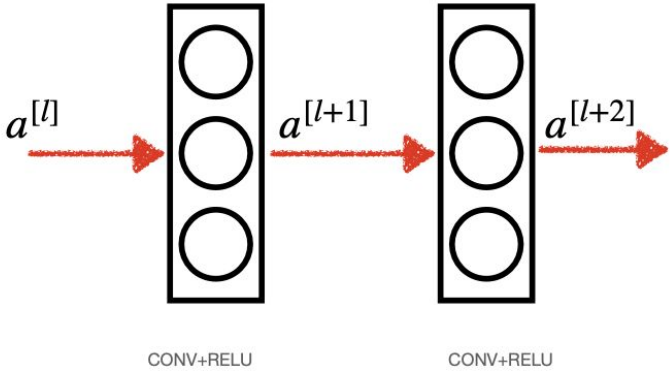
$$z^{[l+1]} = W^{[l+1]}a^{[l]} + b^{[l+1]}$$ First Linear Operation

$$a^{[l+1]} = g(z^{[l+1]})$$ After ReLU operation

$$z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]}$$ Second Linear Operation

$$a^{[l+2]} = g(z^{[l+2]})$$ Output without Short Circuit

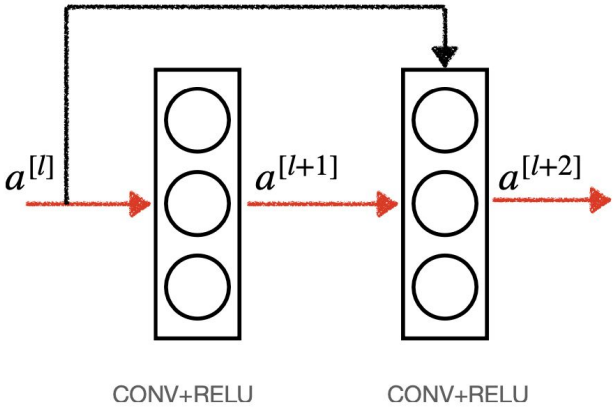
$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$ Output **with** Short Circuit



Output **without** Short Circuit

$$a^{[l+2]} = g(z^{[l+2]})$$

$$a^{[l+2]} = g(W^{[l+2]}a^{[l+1]} + b^{[l+2]})$$



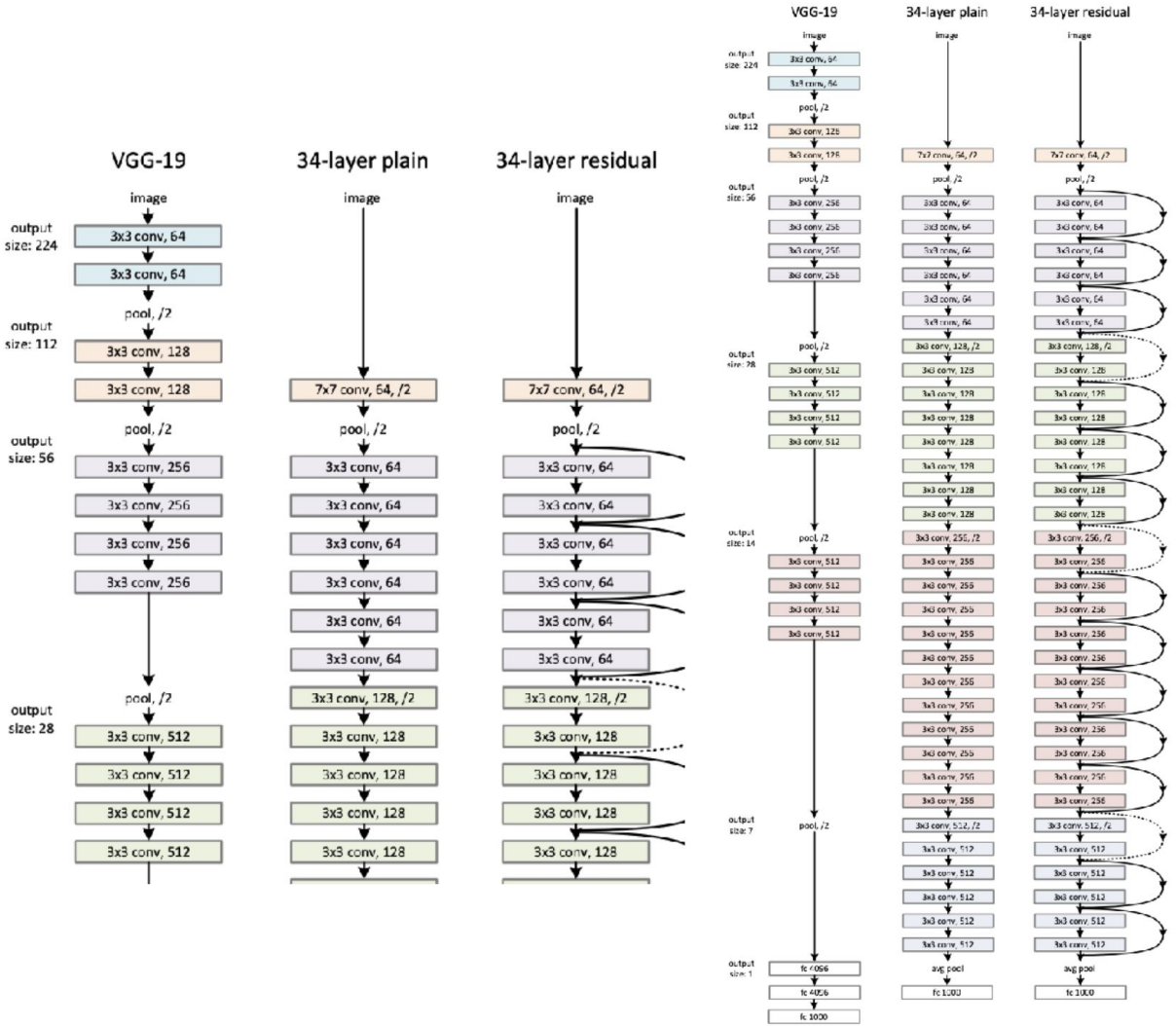
Output **with** Short Circuit

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

$$a^{[l+2]} = g(W^{[l+2]}a^{[l+1]} + b^{[l+2]} + a^{[l]})$$

- If W or b is small or near 0, $a^{[l+2]} = g(a^{[l]})$

- ResNet34 和 ResNet50 具有多個連續的 3x3 卷積層，具有不同大小特徵圖（64、128、256、512），每 2 個卷積會繞過一次
- 它們的輸出尺寸保持不變（padding=1, stride =1）
- ResNet 由多個 residual units 建構，並具有多種不同層數：18、34、50、101、152 和 1202
- 只需要少量甚至不需要 FC Layers，因此能夠使模型更深，學習更多特徵



ResNet 優缺點

優點:

- 有效訓練深層網絡
- 更好的性能
- 模型擴展性強

缺點:

- 計算需求大 (due to deep structure)
- 結構複雜 (for mobile or embedding devices)

MobileNet

- Introduced in 2017
- MobileNet 是輕量版的 CNN，為了能夠被用在嵌入式裝置或手機
- 好的 CNNs 非常龐大（很難放入嵌入式裝置或手機）
- 推論 (Inference) 速度相對較慢 (i.e. forward propagation)

MobileNet Use Cases

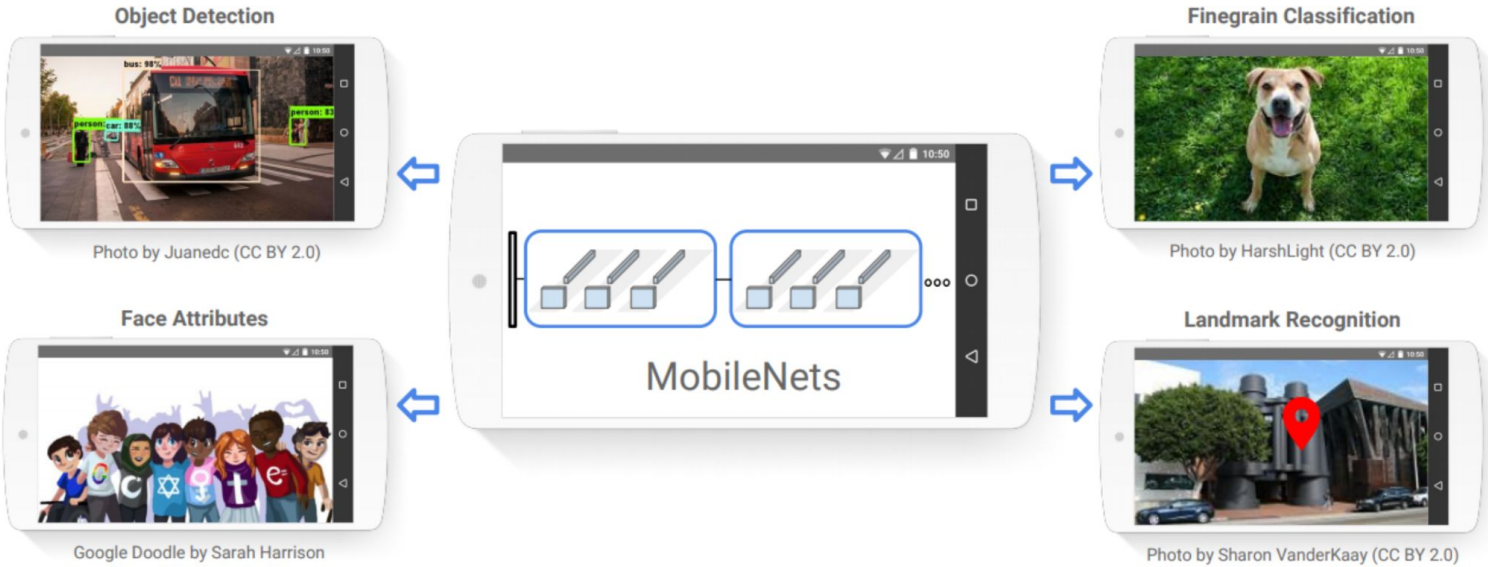


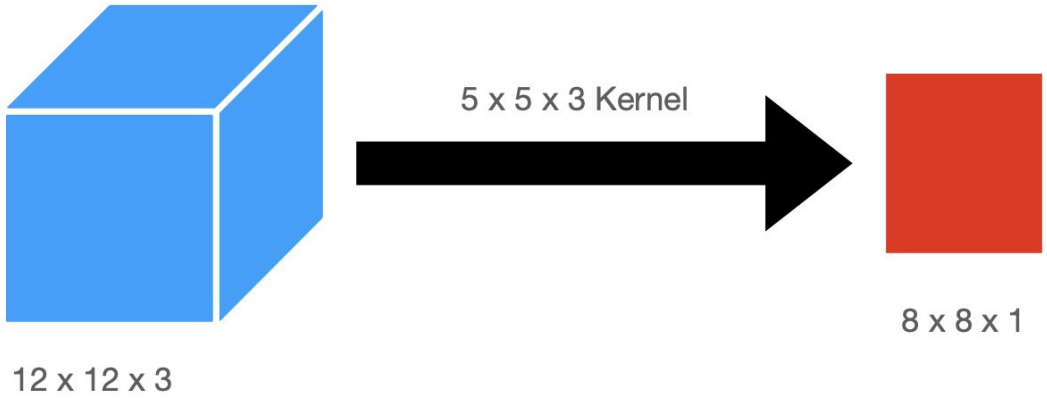
Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

Source: https://arxiv.org/pdf/1704.04861.pdf

針對行動/嵌入裝置的 CNNs

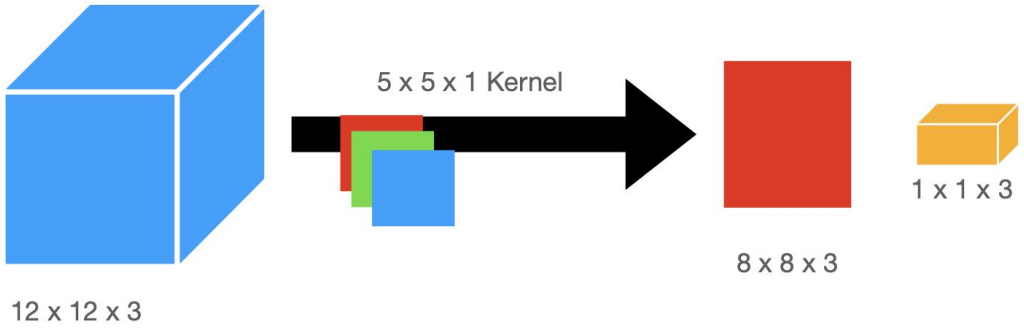
- 行動或嵌入式系統通常具有較低的運算能力，因為它們成本低廉且節能
- 在這些系統上使用 CNNs 需要：
  - 訓練較小的模型
  - 壓縮模型
- MobileNet 透過使用以下方式實現了適合手機的模型：
  - Depthwise Separable Convolutions
  - Two Hyper-Parameters

一般 CNNs 的 Convolutions



- 12x12x3 的輸入圖片與 5x5x3 的 Kernel 卷積產生大小為 8x8x1 的特徵圖
- 當步幅為 1，我們必須執行 5 * 5 * 3 * 64 次操作
- 如果我們有 128 個過濾器，結果是 75 * 64 * 128 = 614,400

Depth Wise Convolutions



- 我們使用 3 個 5x5x1 的 Kernel，並將每個 Kernel 與輸入圖片 (12x12x1) 中的單一頻道相乘
- 得到 8 x 8 x 3 的特徵圖
- 接著使用 Pointwise Convolutions 來得到相同的輸出形狀
  - 將輸出乘以 1x1x3 層
  - 將執行 64 次，並得到 8x8x1 輸出
  - 對輸出進行線性組合

5 x 5 x 3 * 64 = 75 * 64 = 4,800
3 x 64 x 128 = 24,576
4800 + 24,576 = 29,376 Operations

- 大約減少了 20 倍的操作

Two Hyper Parameters

- MobileNet 也透過兩個超參數來有效的降低模型大小
- Width Multiplier: 縮減每一層的深度 (filters數量)
- Resolution Multiplier: 減少輸入影像的大小, 從而減少每個後續層的大小

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Parameters
1.0 MobileNet-224	70.6%	4.2
GoogleNet	69.8%	6.8
VGG 16	71.5%	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Parameters
0.50 MobileNet-160	60.2%	1.32
Squeezenet	57.5%	1.25
AlexNet	57.2%	60

Table 10. MobileNet for Stanford Dogs

Model	Top-1 Accuracy	Million Parameters
Inception V3 [18]	84%	23.2
1.0 MobileNet-224	83.3%	3.3
0.75 MobileNet-224	81.9%	1.9
1.0 MobileNet-192	81.9%	3.3
0.75 MobileNet-192	80.5%	1.9

MobileNet 優缺點

優點:

- 輕量級
- 高效的卷積操作
- 靈活性:可調整width multiplier 和 resolution multiplier

缺點:

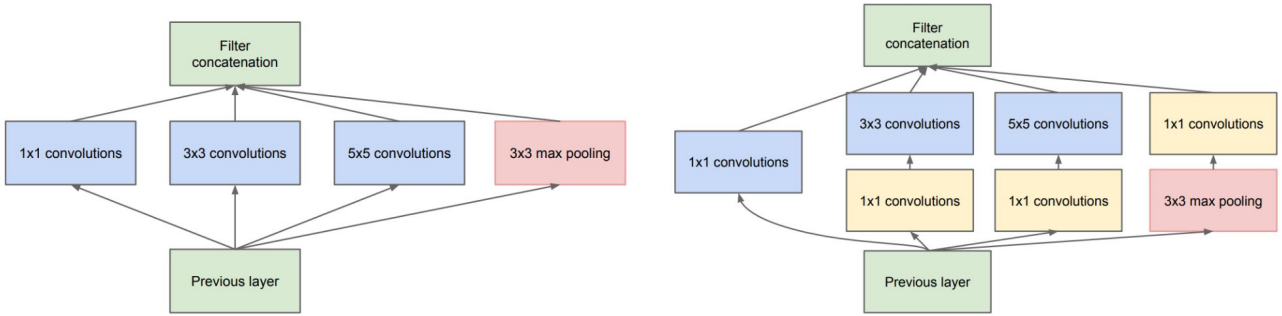
- 準確度相對較低

Inception Network

- 我們知道CNN 涉及大量參數調整
- Filter Sizes, stride, depth, padding, FC layers etc.
- Inception 希望解決 Filter Size 的選擇問題

Inception Network

- The Inception V1 (a.k.a GoogleLeNet) Network was introduced by Google in 2014
- 它在 ImageNet (ILSVRC14) 競賽中取得了最佳的表現

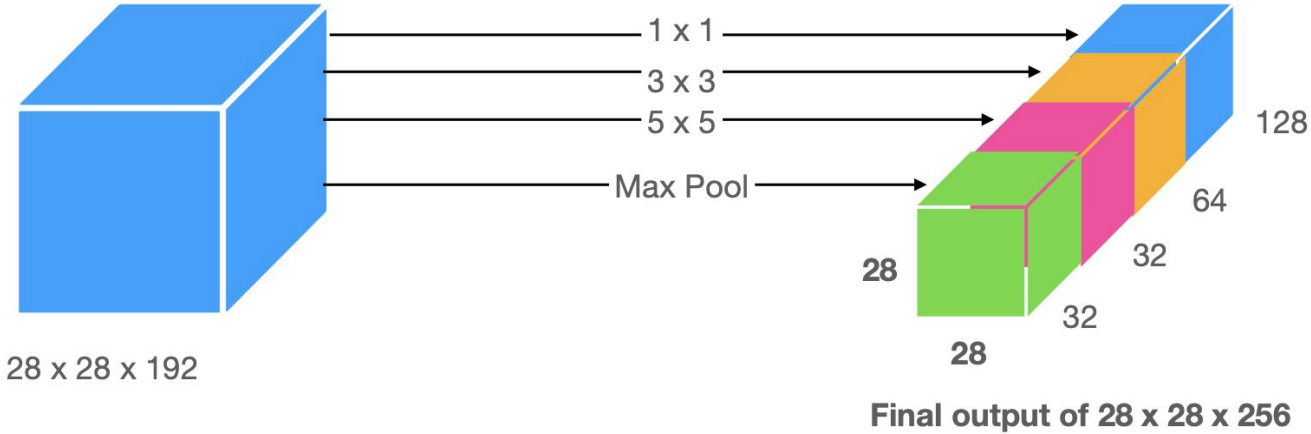


(a) Inception module, naïve version

(b) Inception module with dimension reductions

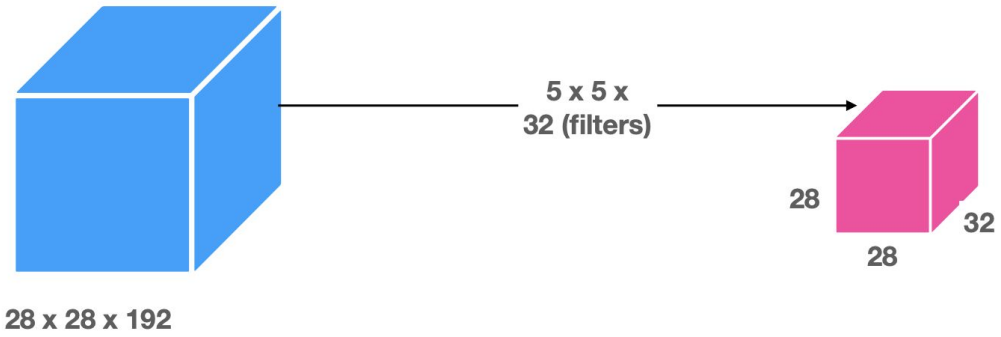
Figure 2: Inception module

使用不同的 Filter Size



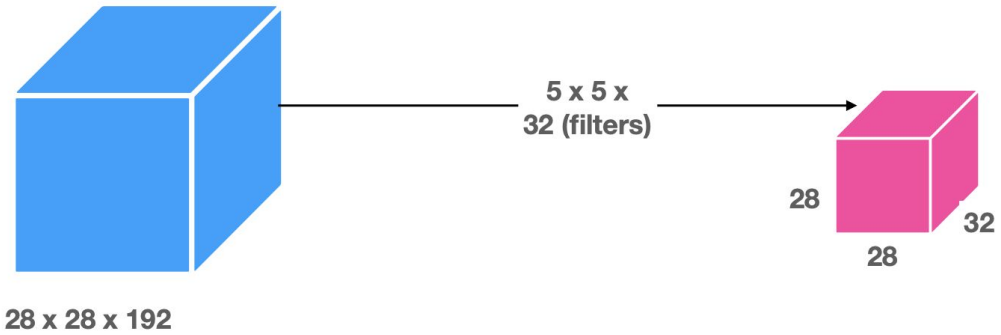
- Inception Network 讓我們可以同時使用幾種不同大小的 Conv Filters
- 使用 'same' padding 和 stride=1 來保持尺寸大小的一致性
- 我們可以執行所有大小的 Filters, 甚至是 Max Pool, 然後將它們堆疊在一起
- 這使得模型能夠學習高階和低階特徵的組合

Heavy Computation



Computations Required
28 x 28 x 32 x 5 x 5 x 192
= 120,000,000

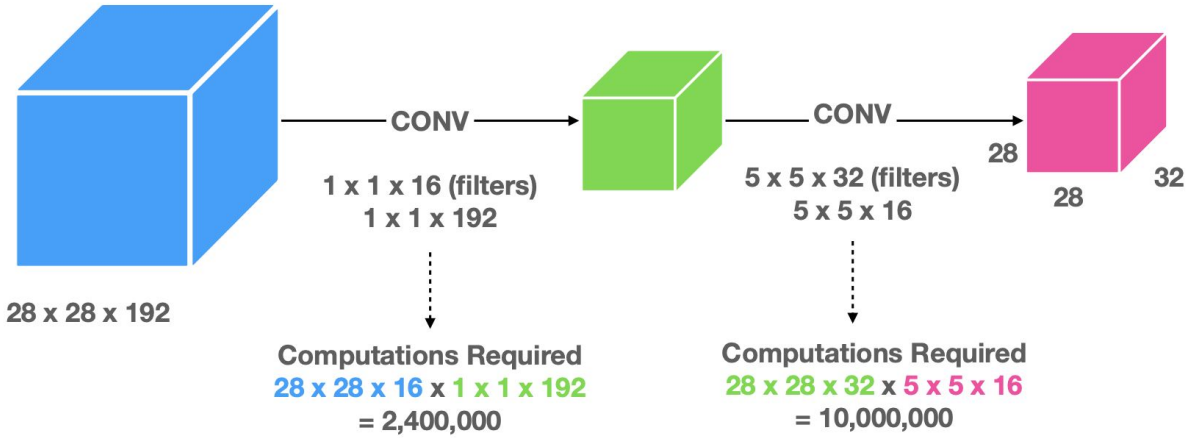
Heavy Computation



Computations Required
28 x 28 x 32 x 5 x 5 x 192
= 120,000,000

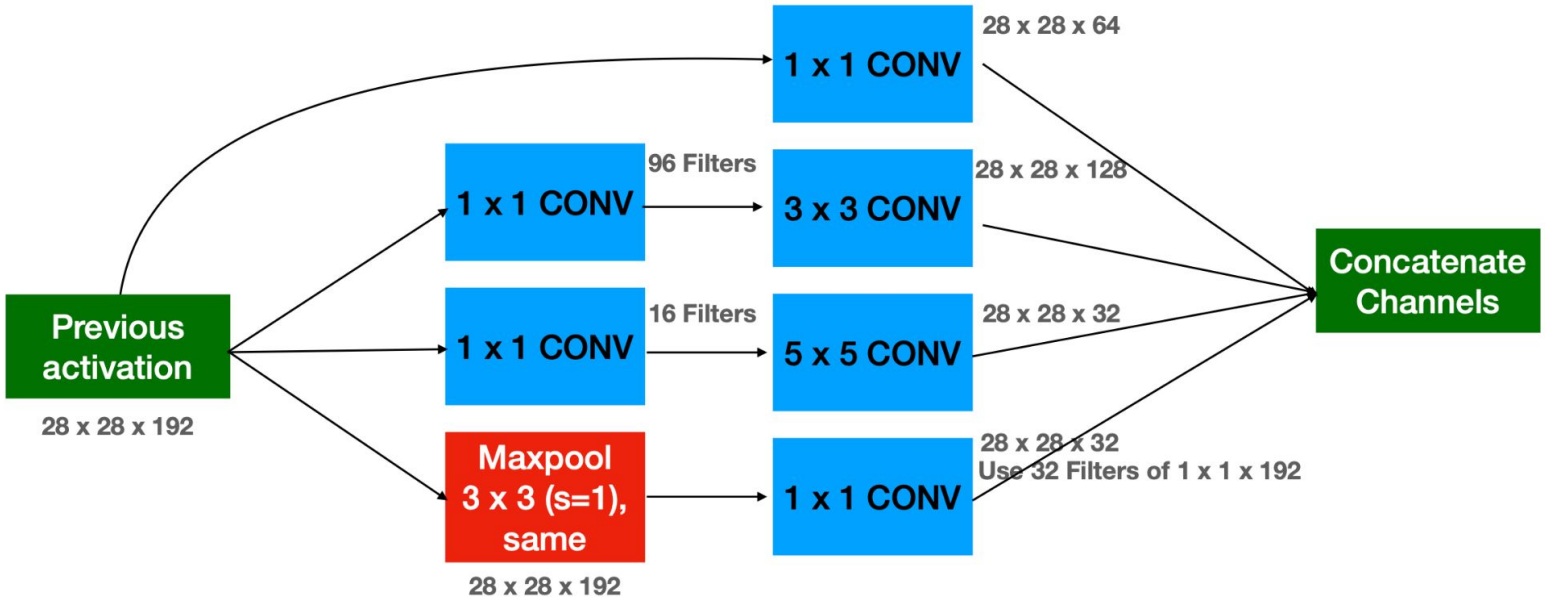
- Use 1x1 Convolutions to reduce the computation cost

使用 Bottleneck Layer

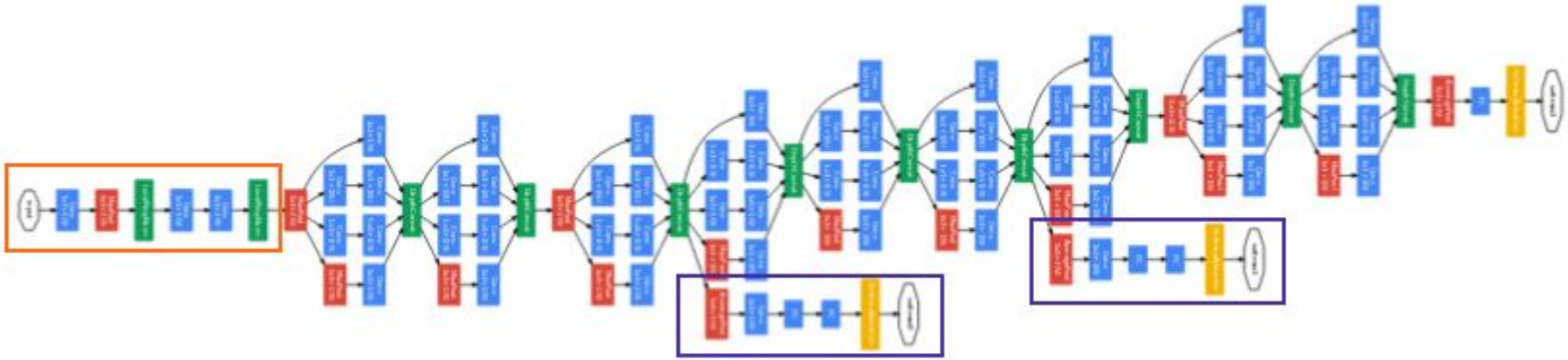


- 先縮小再放大
- 少了 10 倍的計算成本，現在為 2.4M + 10M = 12.4M，而之前為 120M

Inception Block



Inception Design



Stem Inception Block Auxiliary Classifier (非每個版本都有)

Fun Fact: Why is Called Inception

- 作者在論文裡引用了這個迷因，並提到需要使用更深的 CNN
- Inception Network 讓我們能夠非常有效地使用更深的網絡

Inception Network 優缺點

優點:

- 多尺度特徵提取能力
- 高效的計算資源利用 (through 1x1 conv)
- 深層網絡的可訓練性 (e.g. auxiliary classifier)

缺點:

- 結構較為複雜
- 難以移植到資源有限的設備
- 參數和架構選擇複雜

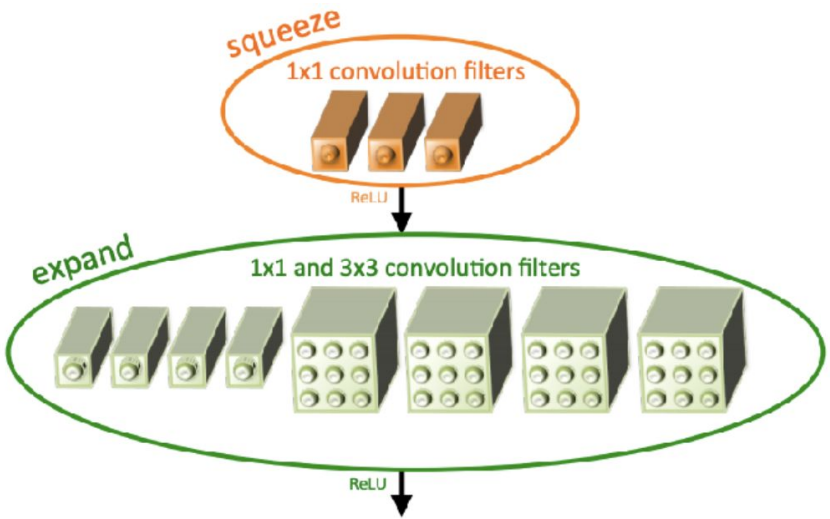
SqueezeNet

- Introduced in 2016
- 希望在維持相同 accuracy 情況下，使用較小的 CNN 架構
  - 在分佈式訓練期間，較小的CNN 只需要少量的communication across servers
  - 較小的 CNN 可以使用更少的頻寬來透過雲端更快地更新模型
  - 較小的 CNN 更適合部署在嵌入式系統
- 它的參數比 AlexNet 少 50 倍，執行速度快 3 倍

SqueezeNets Architectural Design Strategies

- 將 3x3 過濾器替換為 1x1 - 參數比 3x3 過濾器少 9 倍
- 將輸入到 3x3 Filters 的頻道數減少 - 每層中參數的數量為(輸入頻道 * Filters 數 * 3 * 3)
- 晚一點才在網路中進行 Downsampling，以便卷積層具有更大的 Feature Maps

Fire Module - Squeeze and Expand Layers

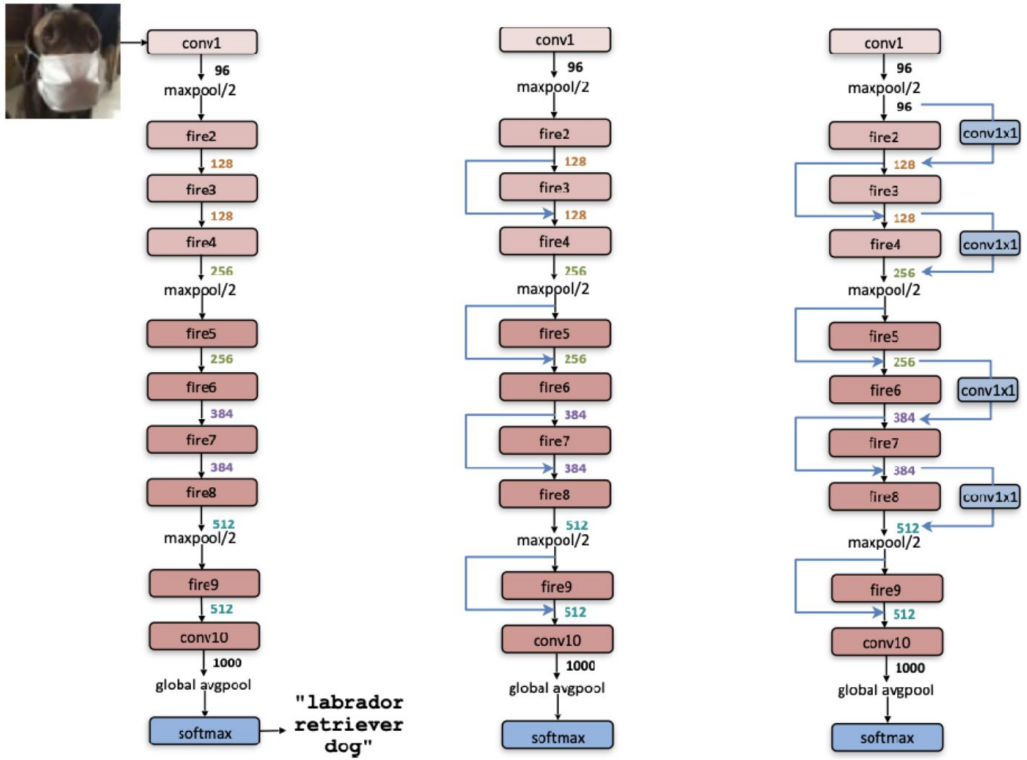


Fire Module with hyperparameters:
s1x1 = 3, e1x1 = 4, and e3x3 = 4

- Fire Module 由以下部分組成：squeeze convolution layer（只有1×1 filter），送入expand layer，expand layer 混合了1×1和3×3 convolution filters
- Fire module中有三個可調維度（超參數）：s1×1、e1×1和e3×3。
- s1×1：squeeze layer中1×1 filter的數量。
- e1×1 和 e3×3: expand layer中1×1和3×3 filter的數量
- 當使用 Fire Module 時，我們將s1×1設定為小於（e1×1+e3×3），因此squeeze layer有助於限制 3×3 Filters 的input channel數量

SqueezeNet

- 完整的 SqueezeNet 架構由一個獨立的 Conv Layer 和後面的 8 個 Fire Module 組成



SqueezeNet (Left), SqueezeNet with simple bypass (Middle), SqueezeNet with complex bypass (Right)

SqueezeNet Performance

Table 2: Comparing SqueezeNet to model compression approaches. By *model size*, we mean the number of bytes required to store all of the parameters in the trained model.

CNN architecture	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

SqueezeNet 優缺點

優點:

- 極小的模型大小
- 相對高的分類性能
- 方便移植到低資源設備

缺點:

- 性能略低於更大的網絡 (compared to VGGNet, Resnet)
- 特徵提取能力有限 (e.g. space feature)
- 相對難以擴展到更大、更深的網路 (compared to Resnet, DenseNet)

EfficientNet

- Introduced in 2019
- Motivation behind EfficientNet
  - CNN 通常以固定的資源成本進行設計，然後進行擴展(e.g. ResNet-18 to ResNet-200)
  - 透過增加深度（層數）或寬度（Filters 數量）來實現縮放
  - 實驗通常很繁瑣，需要手動調整並不容易達到最優結果
- 我們需要一種更有原則的方法來擴展 CNN
- Compound scaling and EfficientNet-B0

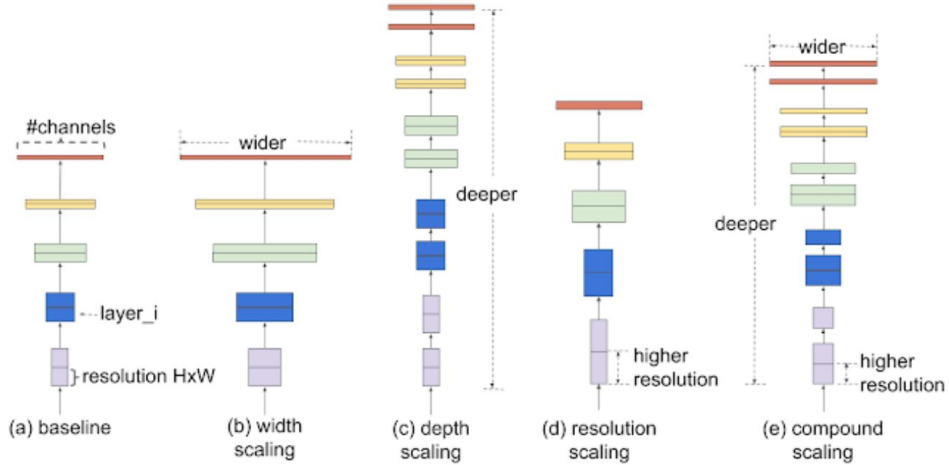
Compound Scaling

- 使用一組固定的縮放係數統一縮放每個維度（寬度、深度、解析度）
  - $depth=\alpha^{\phi}$, $width=\beta^{\phi}$, $resolution=\gamma^{\phi}$

- EfficientNet 系列模型能夠達到 state-of-the-art accurary，並且效率提高 10 倍
- 研究人員研究了放大不同維度的影響。
- 結果發現，平衡所有維度的縮放會帶來最佳的整體效能

Grid Search (網格搜尋)

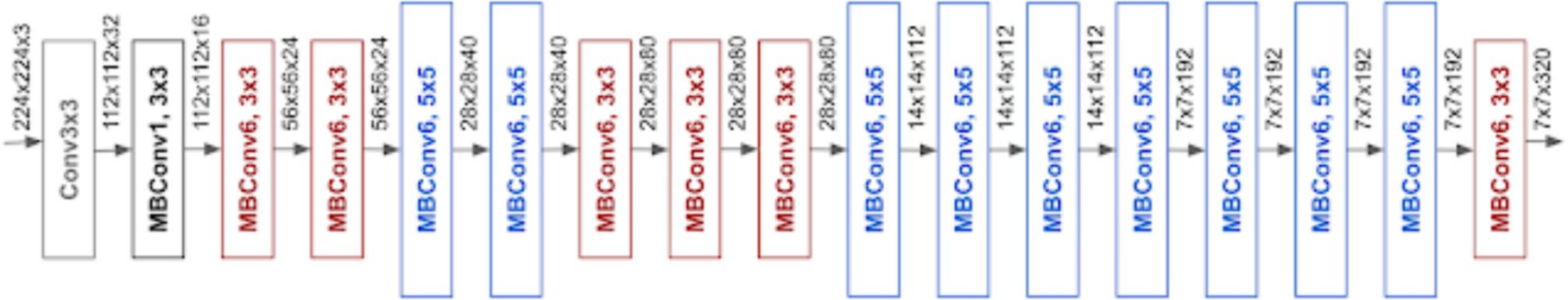
- 使用網格搜尋尋找在固定資源（例如 2 倍以上的 FLOPS）下，對 Baseline Network 進行不同維度縮放之間的關係
- 找到每個維度最合適的縮放係數
- 針對這個固定資源，應用係數將 Baseline Network 擴大到所需的目標模型大小



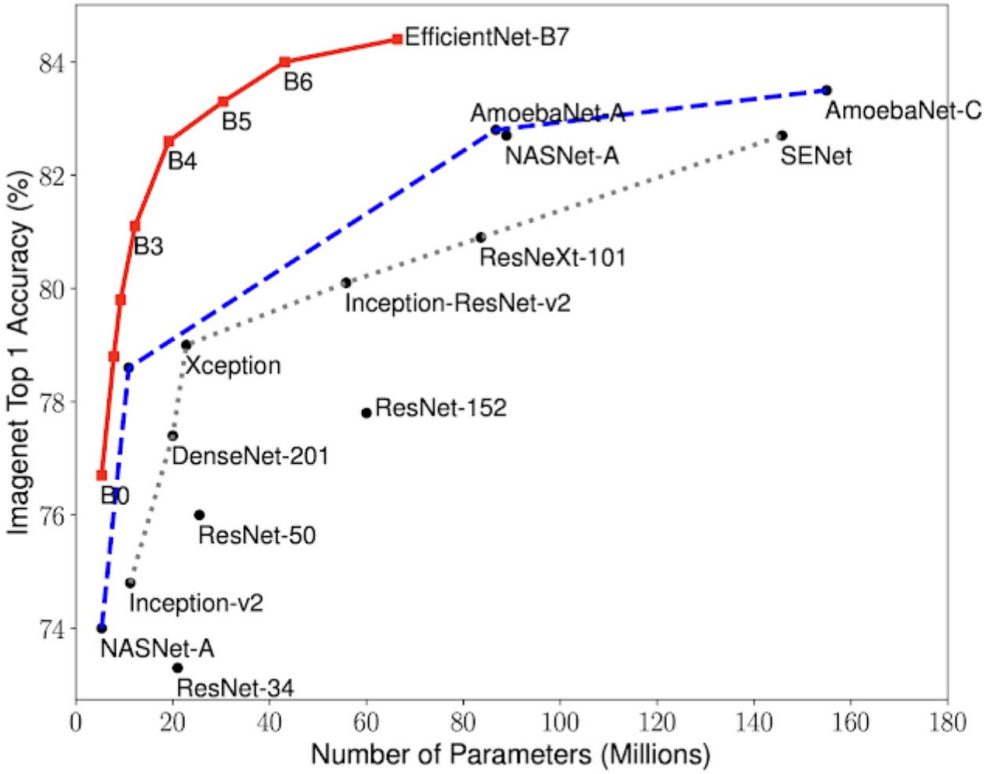
EfficientNet-B0 Architecture

- 複合縮放方法可以應用於任何 CNN（例如 MobileNet 的準確率提高了 1.4%，ResNet 提高了 0.7%）
- 模型縮放的有效性在很大程度上取決於 Baseline Network
- EfficientNet-B0 是使用 Google 的 (NAS, Neural Architecture Search) 技術設計
  - 目的是找到一個在計算資源和性能之間達到平衡的高效卷積神經網路
  - 使用了 MobileNetV2 的 MBConv（Mobile Inverted Bottleneck Convolution）
  - MBConv ≈ depthwise convolution + pointwise convolution + skip connection

EfficientNet Architecture



EfficientNet Performance



EfficientNet 優缺點

優點:

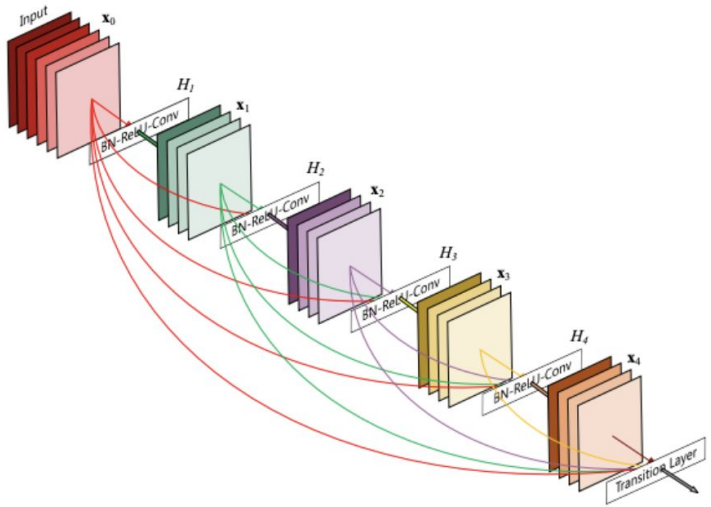
- 更高的性能/計算效率比
- 適應多種應用場景 (B0~B7)

缺點:

- 設計相對複雜 (e.g. rely on NAS)
- 模型訓練難度 (especially for B6, B7)
- 特定於影像分類的優化

DenseNet

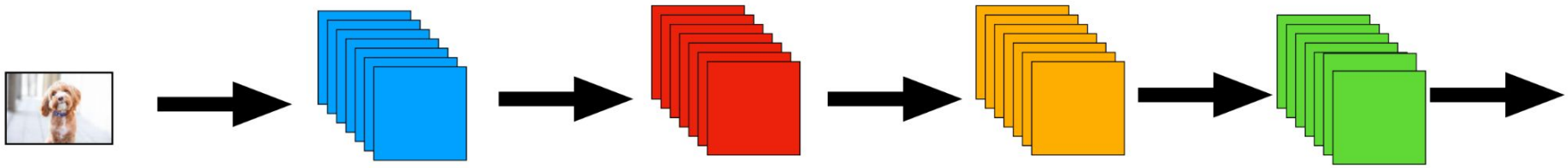
- ResNet but better
- Densely connected convolutional neural networks
- Introduced in 2016 and won Best Paper Award at 2017 CVPR conference
- It was able to attain higher accuracy than ResNet with fewer parameters



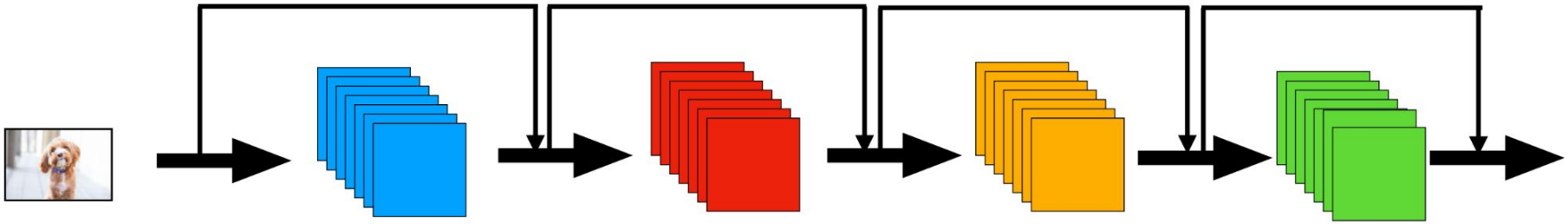
Motivations

- Training Deep CNNs is problematic due to vanishing gradients
- 因為深度網路的路徑變得很長，梯度在完成路徑之前就變為零（vanish）
- DenseNets 透過使用 “Collective Knowledge” 的概念來解決這個問題，其中每一層都接收來自所有先前層的信息

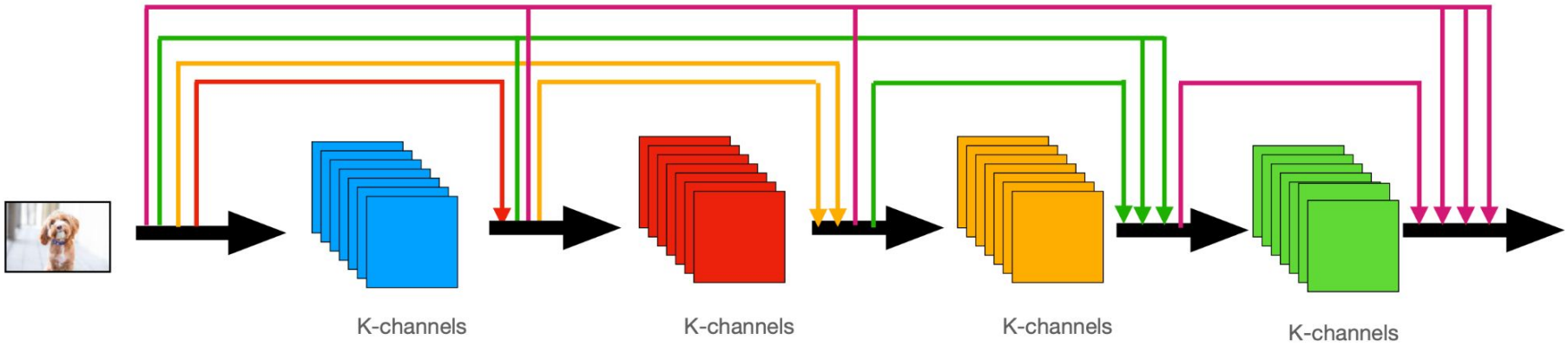
Classical CNN



ResNet Conolution

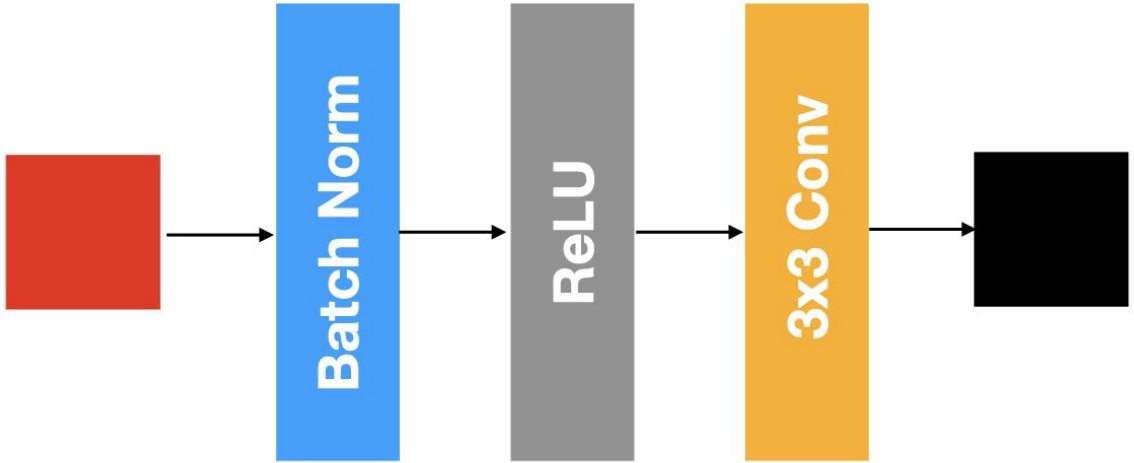


DenseNet Architecture - Dense Block



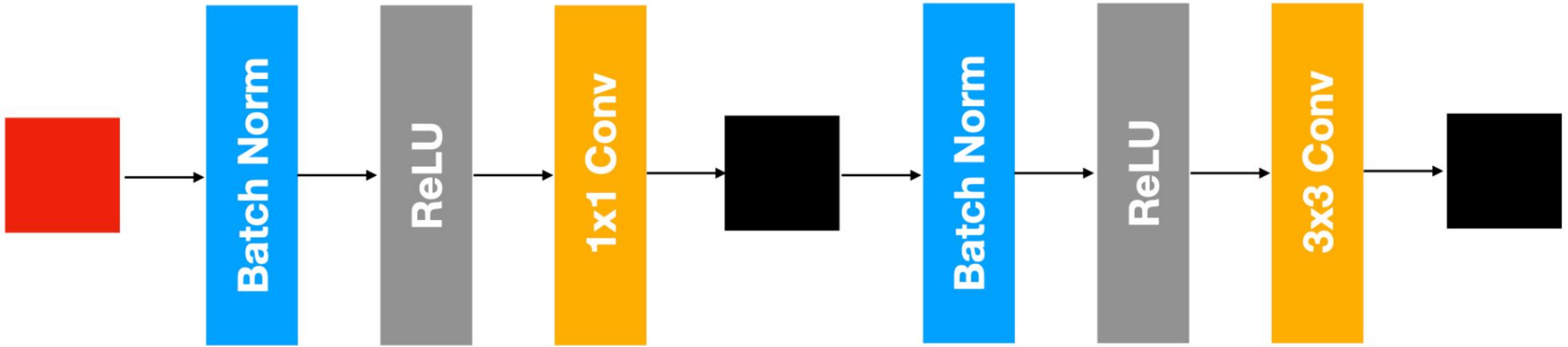
- 每層接收來自所有先前層的信息
- 同一個 Dense Block 裡的特徵圖大小不變

DenseNet Composition Layer



- 基本 DenseNet Composition Layer 包含 Batch Norm、ReLU 和 3x3 Conv Layer

Bottleneck Layer



- BN-ReLU 1x1 Conv is done before BN-ReLU 3x3 Layer

Multiple Dense Blocks with Transition Layers

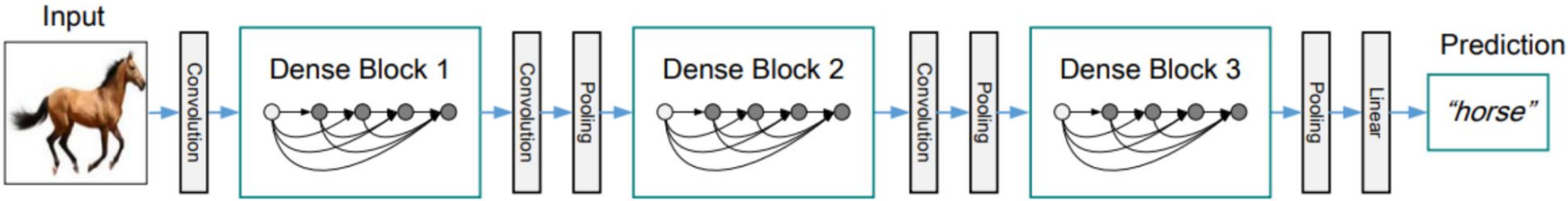


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

- 使用 1x1 Conv 和 2x2 Average Pooling 作為兩個連續密集區塊之間的 “過渡層”

DenseNet Performance

Model	top-1	top-5
DenseNet-121	25.02 / 23.61	7.71 / 6.66
DenseNet-169	23.80 / 22.08	6.85 / 5.92
DenseNet-201	22.58 / 21.46	6.34 / 5.54
DenseNet-264	22.15 / 20.80	6.12 / 5.29

Table 3: The top-1 and top-5 error rates on the ImageNet validation set, with single-crop / 10-crop testing.

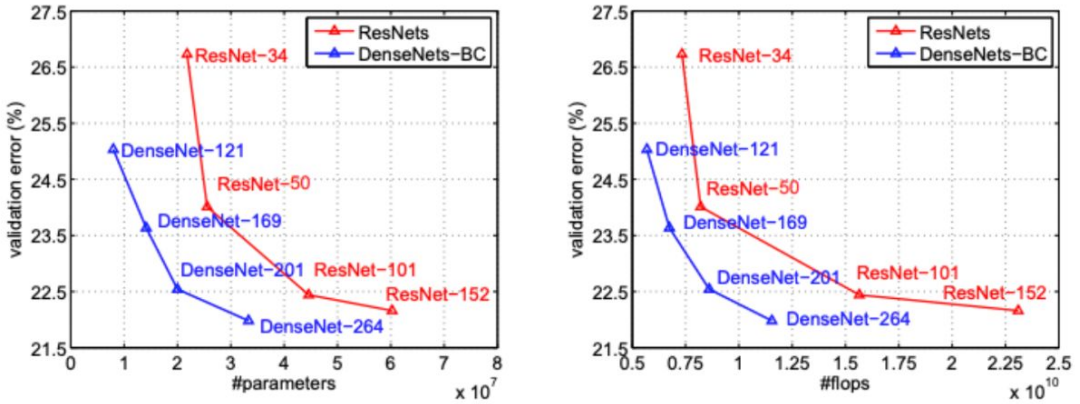


Figure 3: Comparison of the DenseNets and ResNets top-1 error rates (single-crop testing) on the ImageNet validation dataset as a function of learned parameters (*left*) and FLOPs during test-time (*right*).

- DenseNet-B - DenseNets with a bottleneck layer
- DenseNet-BC - Bottleneck + Compression (C) Factor theta (Controls the feature map reduction)

DenseNet 優缺點

優點:

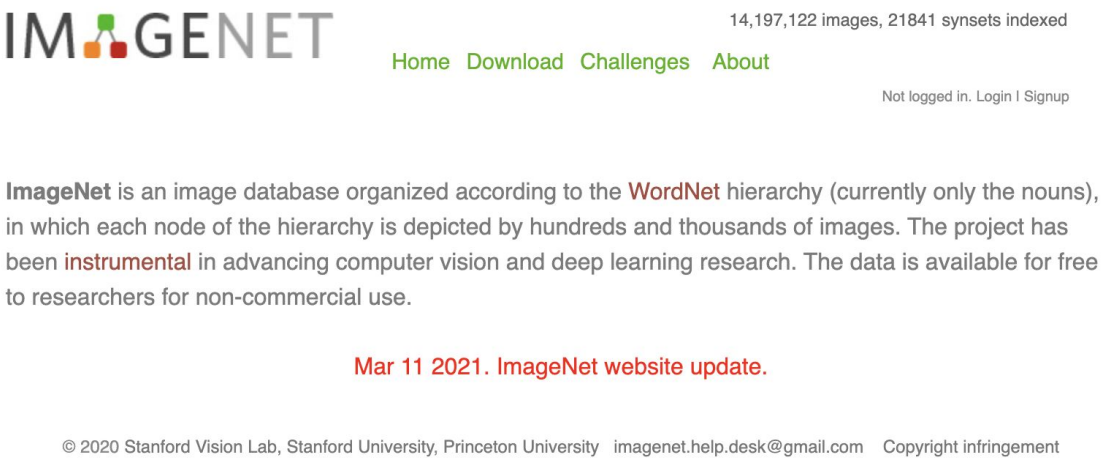
- 減少梯度消失問題
- 促進特徵重用
- 更小的參數量

缺點:

- 高內存需求 (需保存之前所有層的輸出)
- 計算代價較高 (≠ 參數量)

ImageNet

- 目前世界上最大的標記影像資料集



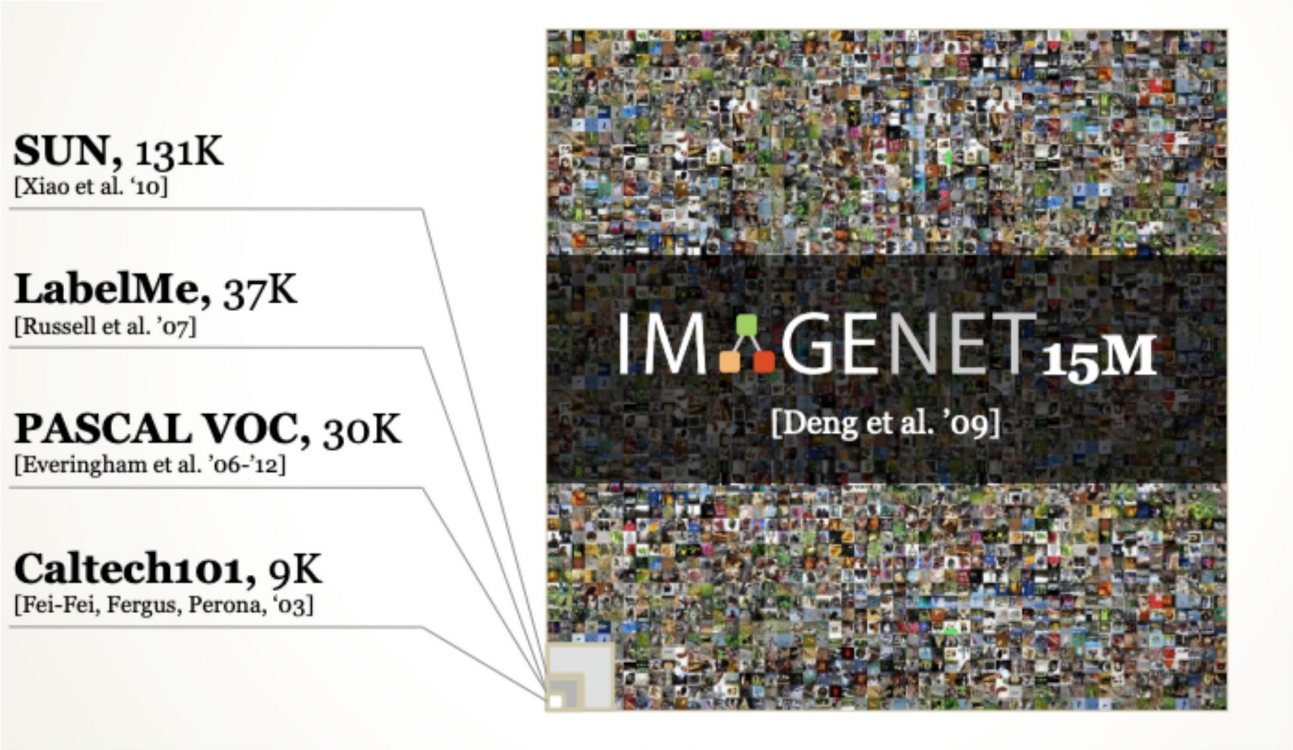
Source: https://www.image-net.org/

ImageNet - ILSVR

- The most highly-used subset of ImageNet is the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012-2017 image classification and localisation dataset
- 此資料集涵蓋 1000 個物件類別，包含 1,281,167 個訓練影像、50,000 個驗證影像和 100,000 個測試影像。此子集可在 Kaggle 上使用
- 它是評估新的 CNN 模型時最常用的基準

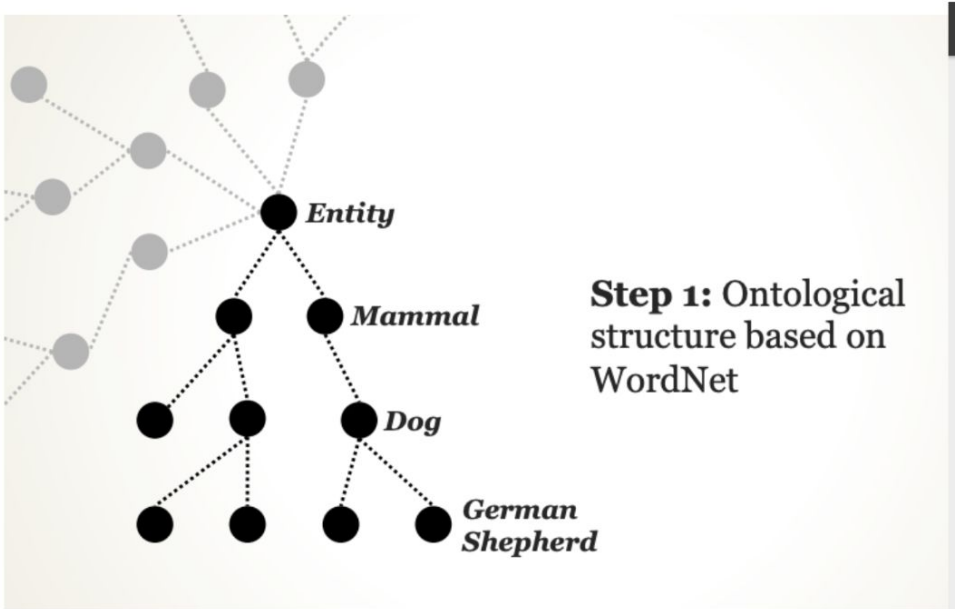
What Makes ImageNet so Good?

- Size

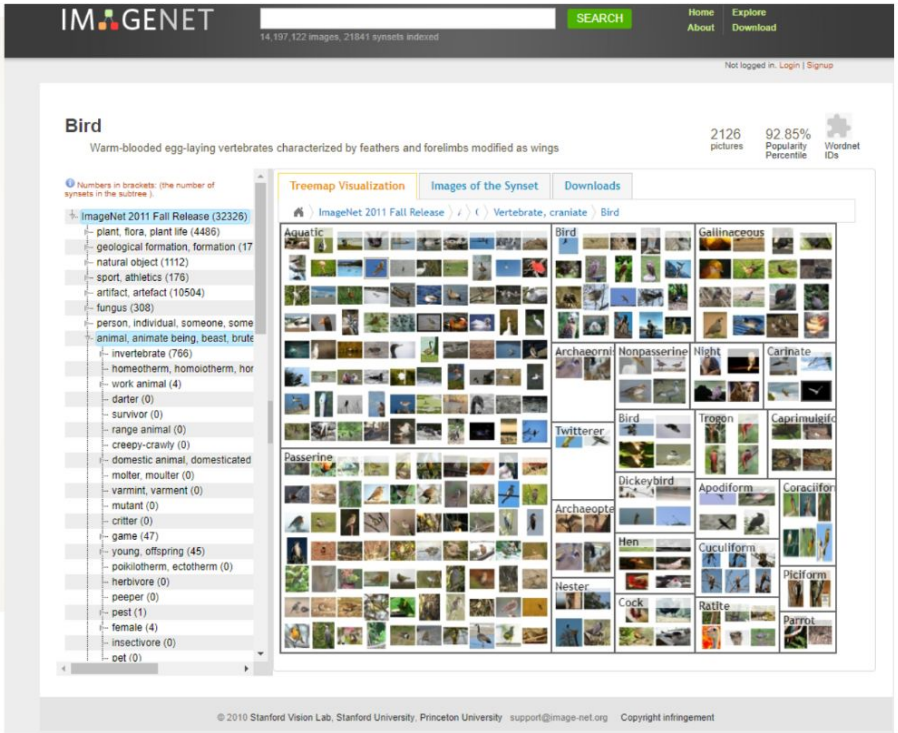


Source: https://www.image-net.org/static_files/files/imagenet_ilsvrc2017_v1.0.pdf

WordNet Hierarchy

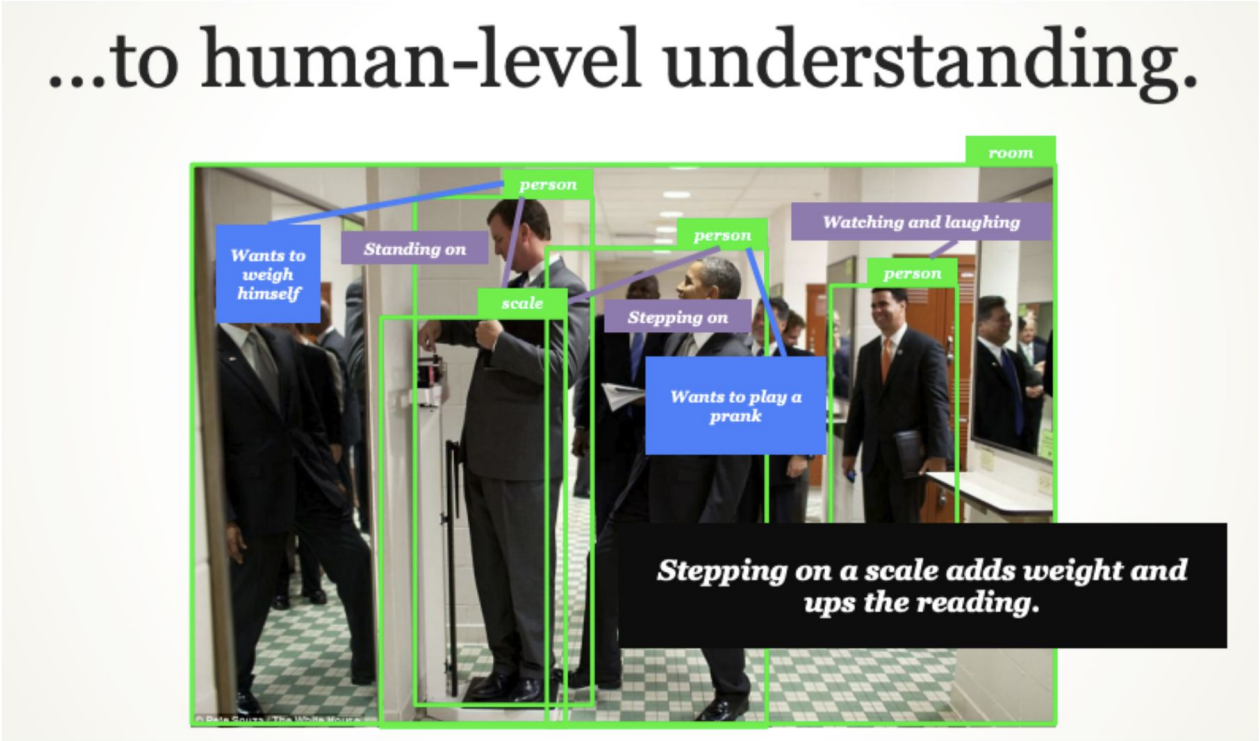


Step 1: Ontological structure based on WordNet



Source: https://www.image-net.org/static_files/files/imagenet_ilsvrc2017_v1.0.pdf

幫助 AI 理解圖片裡的資訊



Source: https://www.image-net.org/static_files/files/imagenet_ilsvrc2017_v1.0.pdf

Current State of Art Performance

- https://paperswithcode.com/sota/image-classification-on-imagenet

Rank-N or Top-N Accuracy

- Rank-N Accuracy 是一種具有更多空間的評估分類器準確性的方法
- 有時候分類器仍然做得很好，但如果我們只查看最上面的預測類別，則不會反映出來
- Rank-N Accuracy 考慮機率最高的前 N 個類別

Class Name	Probability
Shetland sheepdog	0.44
collie	0.31
chow	0.1
wire-haired fox terrie	0.09
lion	0.06

Implementation

Download notebook at:
https://github.com/albert831229/nchu-computer-vision/tree/main/113/day