Machine Learning

Lecture 10
Network Design and Optimization

Chen-Kuo Chiang (江 振 國)
ckchiang@cs.ccu.edu.tw
中正大學 資訊工程學系

The Storyline

- Pre-train versus Fine-tune
- Factors of transferability 遷移模型

Network Design and Optimization

- Question ? 當要遷移已知模型到其他地方，所需注意的部分
  - How can the performance of a ConvNet representation be *maximized* for a particular target task?

- The ways a deep ConvNet representation can be learned and adjusted to allow better transfer learning from a source task producing a generic representation to a specific target task.

Network Design and Optimization

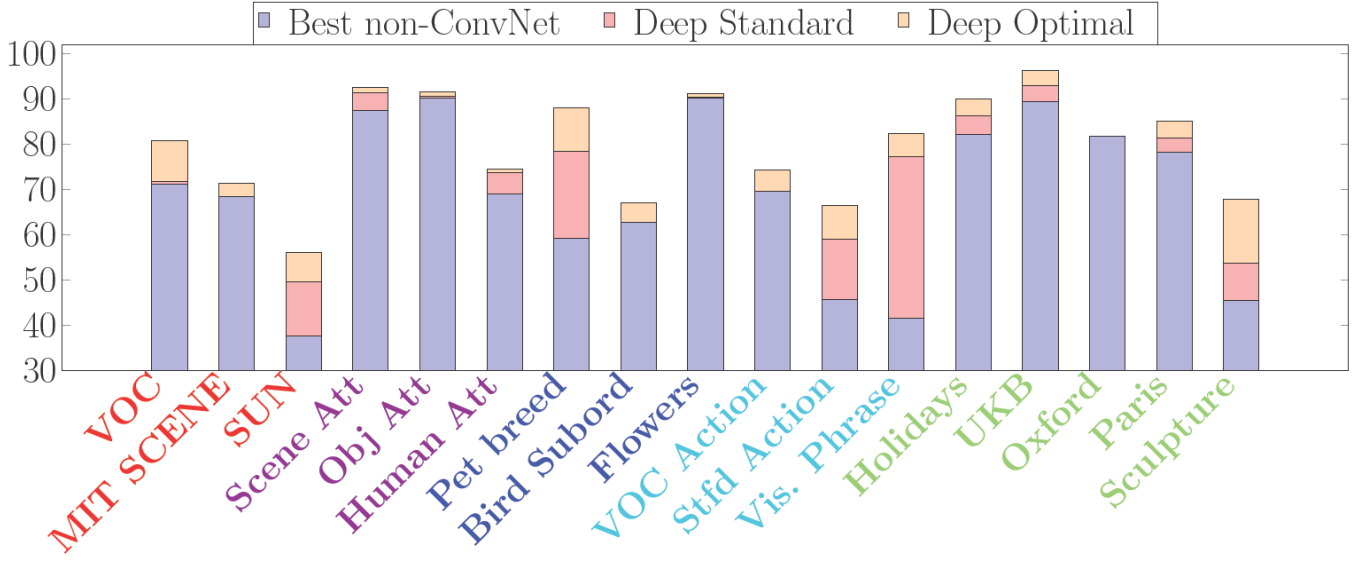
- Answer
  - Factors of transferability

    - VGGNet, GoogleNet or AlexNet are usually trained by ImageNet dataset. How to exploit such model to your own research problem?

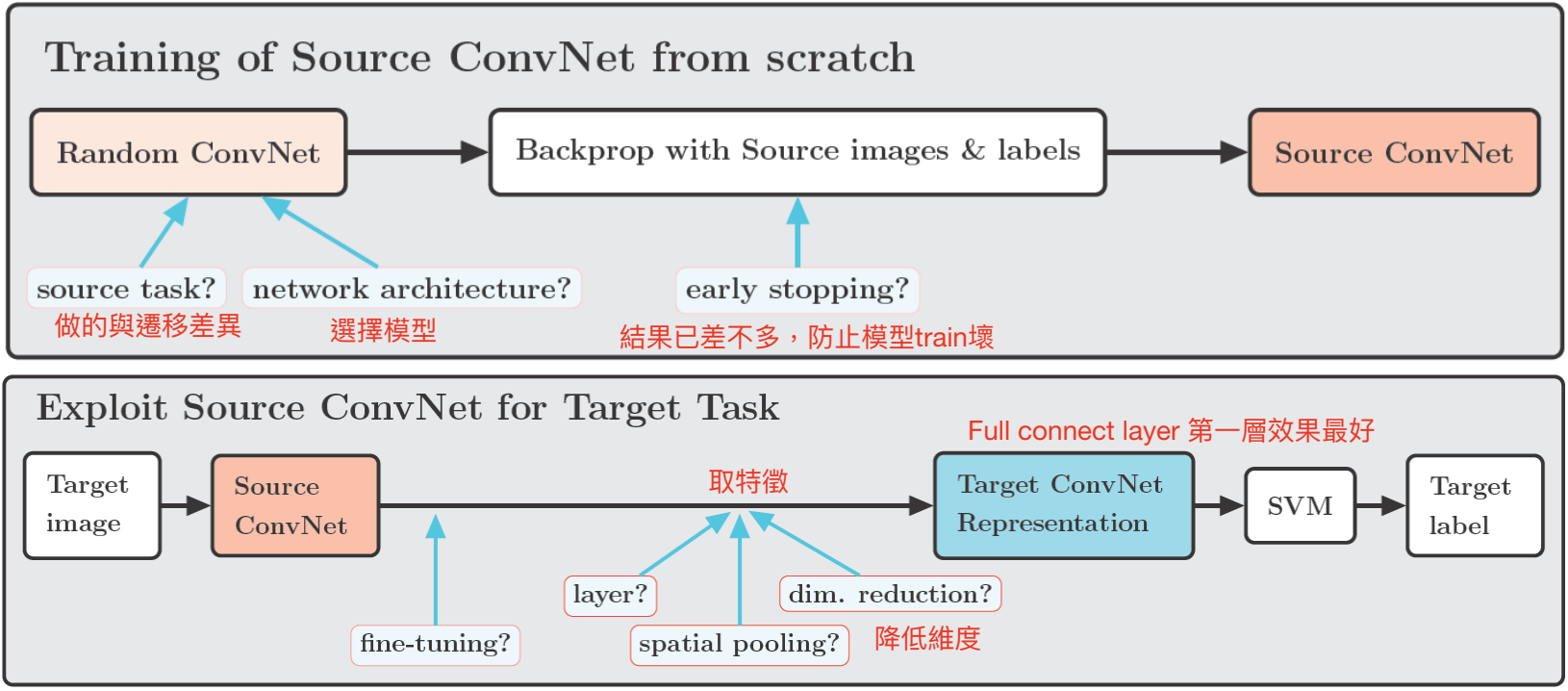
Transferability Factors

- Best non-ConvNet : non-ConvNet state of the art systems 傳統
- Deep Standard : standard ConvNet features with a linear SVM classifier 一般
- Deep Optimal : optimizing the **transferability factors** for each task

遷移模型



Transferring ConvNet Representation



Range of Target Tasks

- The order these target tasks are based on their similarity to the source task of object image classification as defined by ILSVRC12.
- Instance retrieval as the least similar to the source task. Each task in this set has no explicit category information and is solved by explicit matching to exemplar images
- Compositional- how specific objects interact with one another. Play/hold.

需要找出data與問題相關度高的

資料與要做的事情關係度

Decreasing similarity to ImageNet →

解決的問題

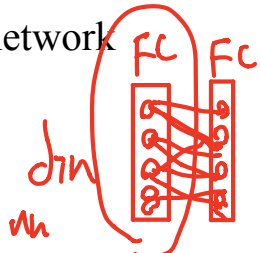
Decreasing similarity to ImageNet ↓	Image Classification	Attribute Detection	Fine-grained Recognition	Compositional	Instance Retrieval
	PASCAL VOC Object	H3D human attributes	Cat & Dog breeds	VOC Human Action	Holiday scenes
	MIT 67 Indoor Scenes	Object attributes	Bird subordinate	Stanford 40 Actions	Paris buildings
	SUN 397 Scene	SUN scene attributes	102 Flowers	Visual Phrases	Sculptures

Network Width 每層filter多少

	Convolutional layers			FC layers
Network	N_T	n_k per layer	output dim	n_h per layer
Tiny (A)	14M	(24, 64, 96, 96, 64)	6x6x64 大小	(4096, 1024, 1000) 多少node
Small (C)	29M	(48, 128, 192, 192, 128)	6x6x128	(4096, 2048, 1000)
Medium(E)	59M	(96, 256, 384, 384, 256)	6x6x256	(4096, 4096, 1000)
Large (F)	138M	(96, 256, 512, 512, 1024, 1024)	5x5x1024	(4096, 4096, 1000)

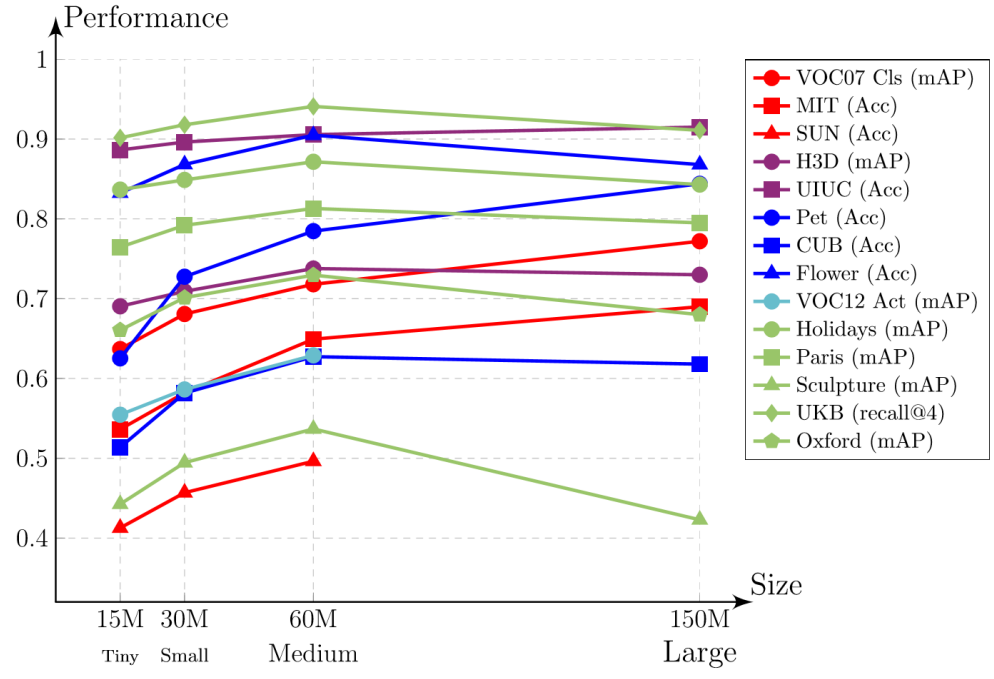
- By keeping the network depth fixed, examine the impact of the network width on different tasks:
  - N_T : the total number of weights parameters in the network
  - n_k : the number of kernels at a convolutional layer
  - n_h : the number of nodes in a fully connected layer.

CNN縮小的維度



Results of Various Network Width

寬度



- 對於model的width設計來說參數量Large時(150M)，只有Target Dataset和Source Dataset很相像時，效果才會提升(**VOC07, MIT, SUN**)。

- 與Source Dataset較不相似時，適合中大型Width Network的(**Medium 59M**)。

  資料與問題不同時可以使用medium就好

- **Tiny和Small**的表現和Large的表現差異不大。因此在移動式裝置或是需要簡化運算量時，Width不是一個很重要的參數，甚至可以設計的小一點，表現更好。

  和large差異不大，small就夠用了

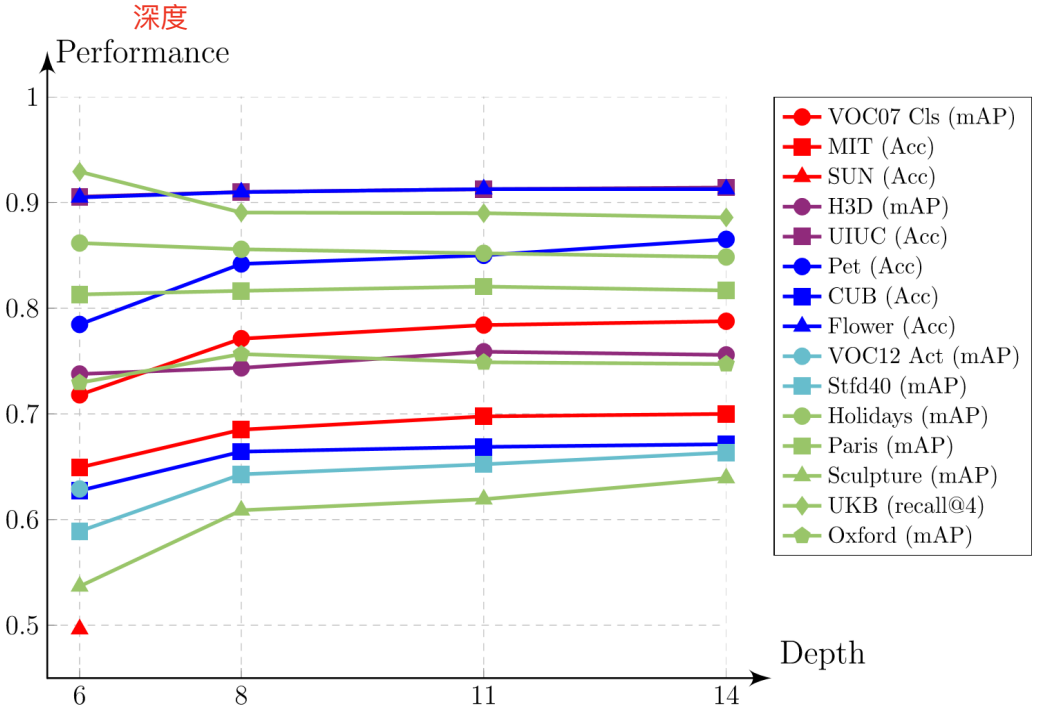
Network Depth

	Convolutional layers			FC layers
Network	N_T	n_k per layer	output dim	n_h per layer
Deep8 (~E)	85M	(1×64, 1×128, 3×256)	8×8×256	(4096, 4096, 1000)
Deep11 (H)	86M	(1×64, 3×128, 4×256)	8×8×256	(4096, 4096, 1000)
Deep13 (I)	86M	(2×64, 4×128, 4×256)	8×8×256	(4096, 4096, 1000)
Deep16 (J)	87M	(2×64, 5×128, 6×256)	8×8×256	(4096, 4096, 1000)

- By keeping the number of network parameters fixed, examine the impact of the network depth on different tasks:
  - N_T : the total number of weights parameters in the network
  - n_k : the number of kernels at a convolutional layer
  - n_h : the number of nodes in a fully connected layer.

總參數量固定
深一點是否有差異

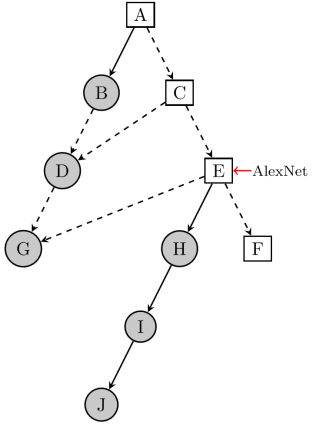
Results of Network Depth



- 不論和Source Dataset相似與否，Network的Depth影響全部Target Dataset的效果。
- 因此Network的Depth是設計的越深越好!

Width versus Depth

Convolutional layers				FC layers
Network	N_T	n_k per layer	output dim	n_h per layer
Deep Tiny(B)	85M	(13×64)	6×6×64	(4096, 1024, 1000)
Deep Small (D)	86M	(13×128)	6×6×128	(4096, 2048, 1000)
Deep Medium(G)	86M	(13×256)	6×6×256	(4096, 4096, 1000)

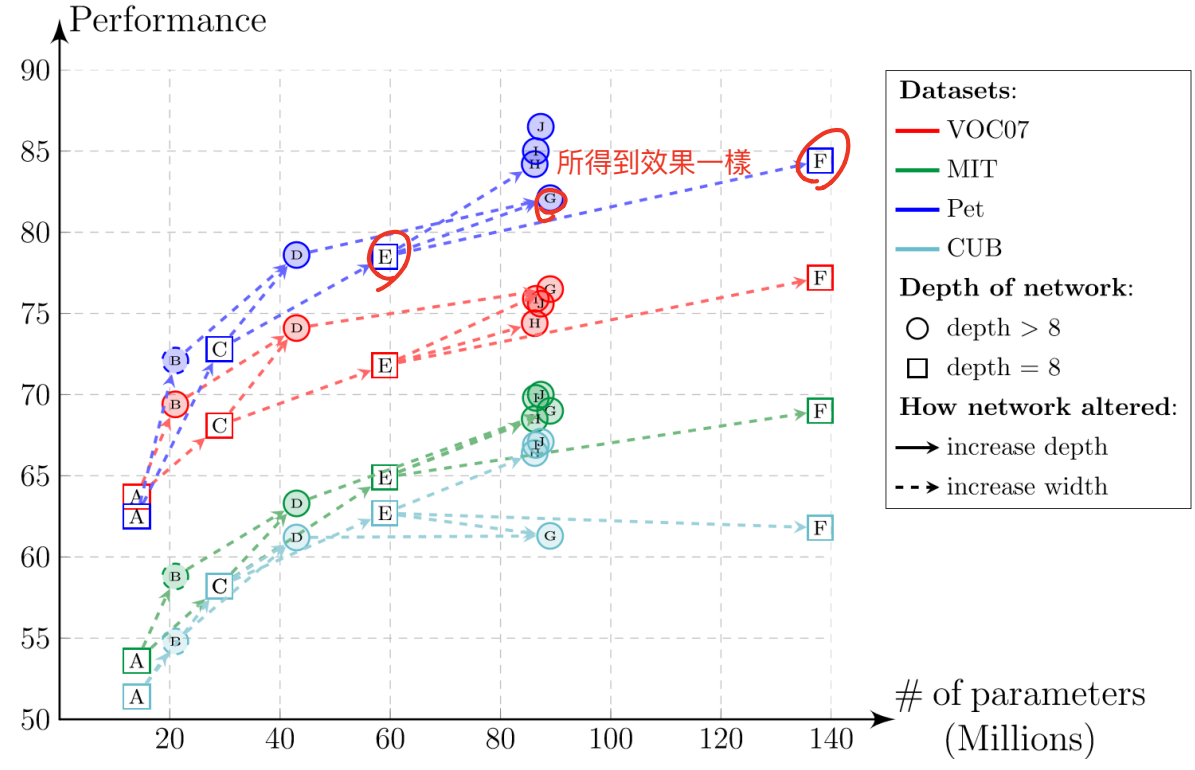


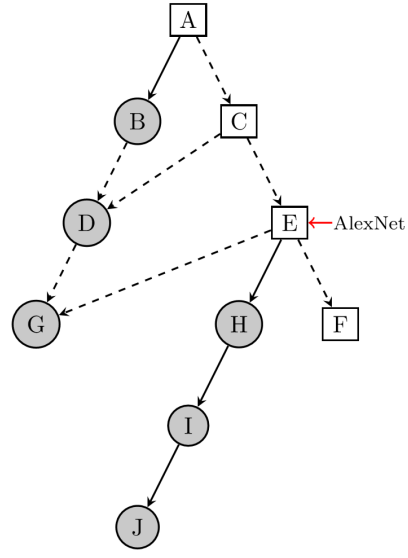
How network altered:

→ increase depth

- - → increase width

Results of Width versus Depth

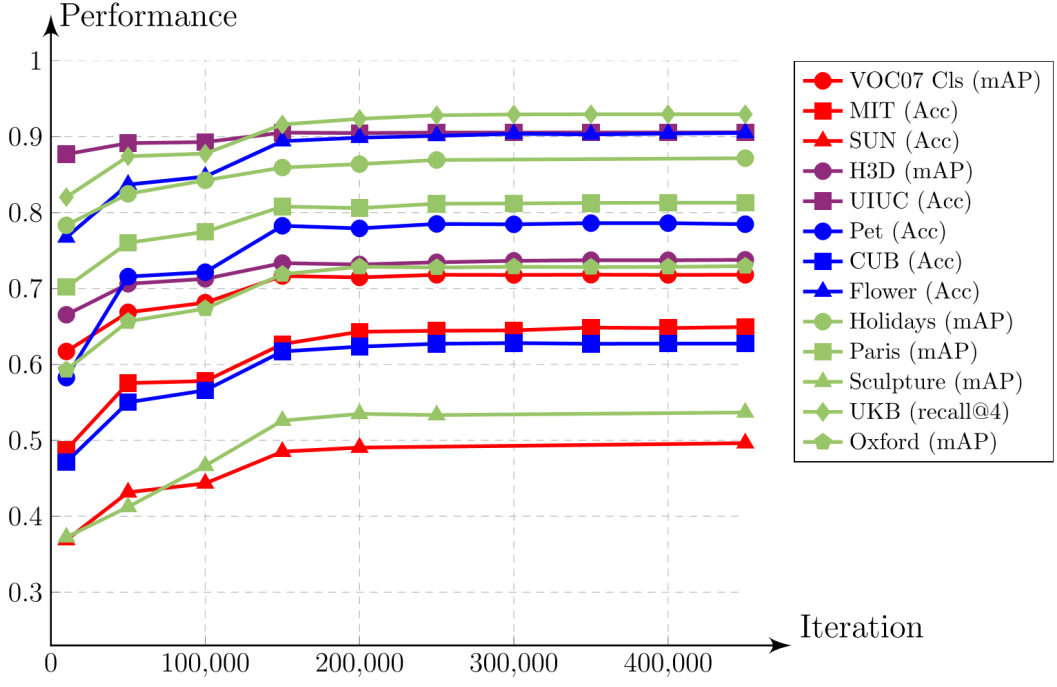




- 觀察A-B與A-C，E-F與E-H，改變Depth得到的效果提升比改變Width來的明顯且重要。

Early Stopping

提早結束



訓練次數大概在

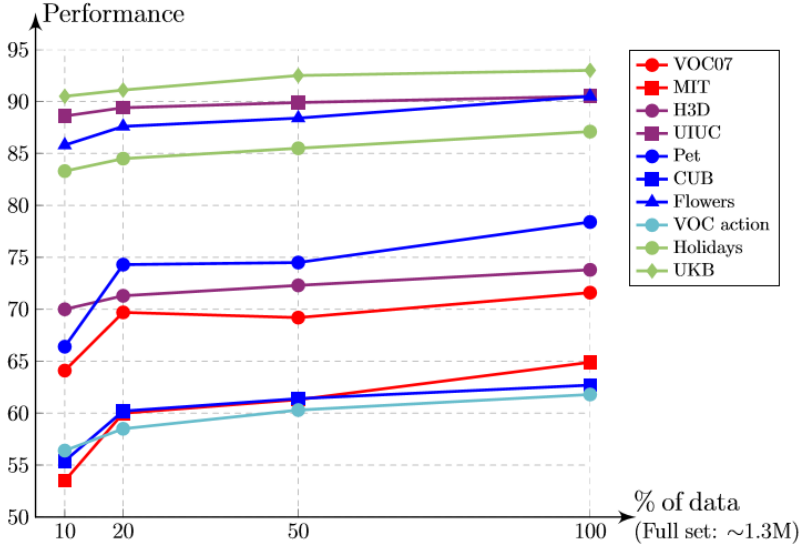
- 大約在200000的iteration(約40 epoch)就無法再有效的提升效果。效果差不多
- 因此不需要提早停止，因為絕大部分的training的epoch都大約在這個區間。
- 從這個例子並未看到overfitting，因次並不代表一定不需要提早停止。

Data Density & Data Diversity

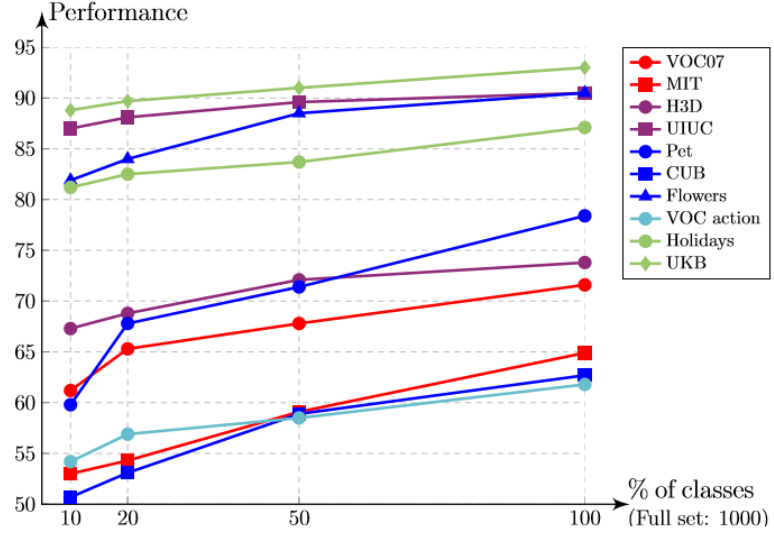
- Data Density – 單一class中資料量變多
- Data Diversity – class的數量變多

1. 圖左單一class的data數量越多越好
2. 圖右dataset的class種類數量越多越好

類別數多分類準確度會提高



Increase datasets 10%, 20%, and 50% of the 1.3 million images in ILSVRC12.



Increased the number of classes from 100 to 1000, but kept the number of images the same as in ILSVRC 2012.

Source Task

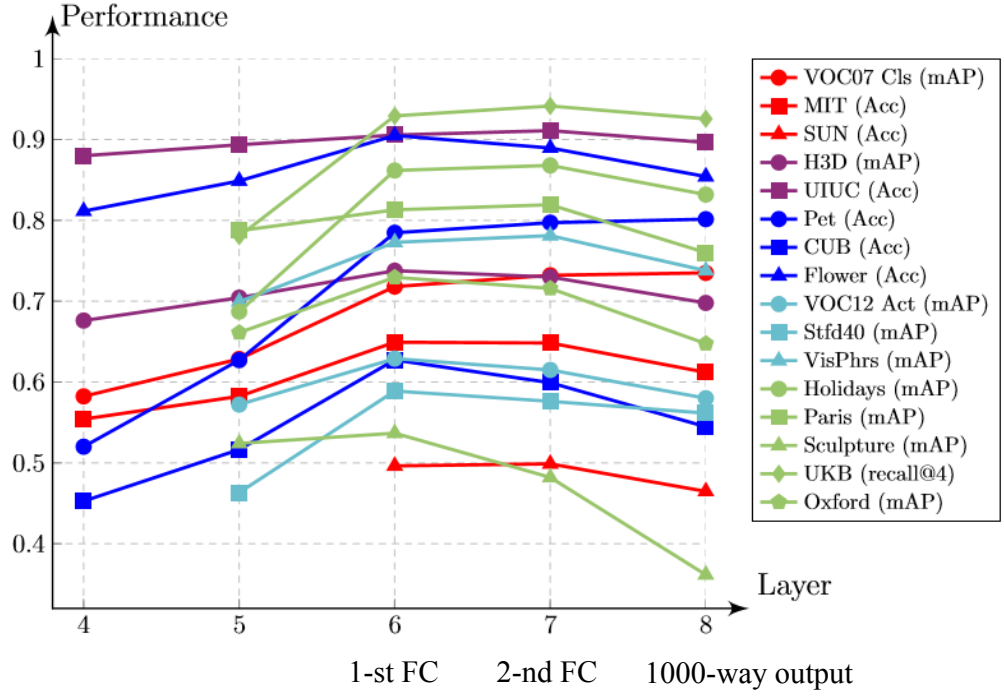
	Image Classification			Attribute Detection		Fine-grained Recognition			Compositional	Instance Retrieval		
Source task	VOC07	MIT	SUN	H3D	UIUC	Pet	CUB	Flower	Stanf. Act40	Oxf.	Scul.	UKB
ImageNet	71.6	64.9	49.6	73.8	**90.4**	**78.4**	**62.7**	**90.5**	58.9	71.2	52.0	93.0
Places	68.5	69.3	55.7	68.0	88.8	49.9	42.2	82.4	53.0	70.0	44.2	88.7
Hybrid	72.7	69.6	56.0	72.6	90.2	72.4	58.3	89.4	58.2	**72.3**	52.3	92.2
Concat	**73.8**	**70.8**	**56.2**	**74.2**	**90.4**	75.6	60.3	90.2	**59.6**	72.1	**54.0**	**93.2**

- ImageNet: 1.3M張影像 (Object)
- Places: 3.5M張影像 (Scene)
- Hybrid: 4.8M張影像(ImageNet+Places)
- Concat: 將ImageNet和Places分別訓練完後的model，取得兩組feature concatenate作為特徵

兩個data set特徵取出再串起來

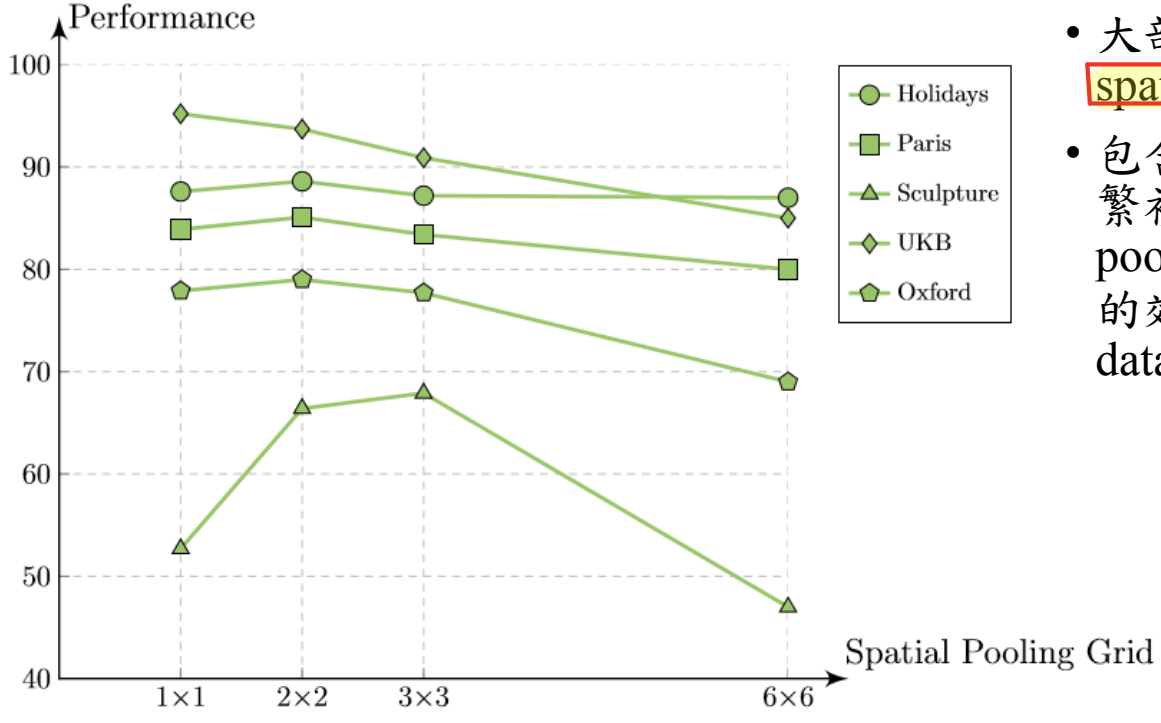
1. Hybrid特徵對於相似於Source Dataset的有較好的效果。對於不相似的dataset，因為Source Dataset的Label多樣性ImageNet的Label大於Places，增加類別數量是好事
2. Places雖然Dataset的影像數量是ImageNet的數量近3倍，但是效果都不理想，證實類別量的效果>資料量
3. Hybrid的結果不總是最好，可以歸咎於Places數量多，訓練產生了bias。當使用Concat設定，除了Fine-grained，效果皆得到提升。對Fine-grained而言，Places的資料並不相關，徒增加資料維度。

Network Layer as Feature Representation

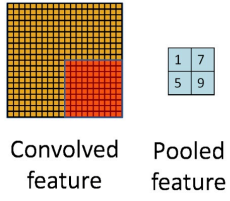


- 對Source Dataset最相似的幾個Target Dataset，取最後一層當Feature會最好。
- 對於和Source Dataset 稍微相似的則是取倒數第二層最好。
- 對於大多數其餘的Target Dataset來說，取自第六層的Feature，也就是第一層的FC來說效果會最好
- 建議全部取自第六層也就是第一層FC來當作Feature的效果會是普遍來說最好

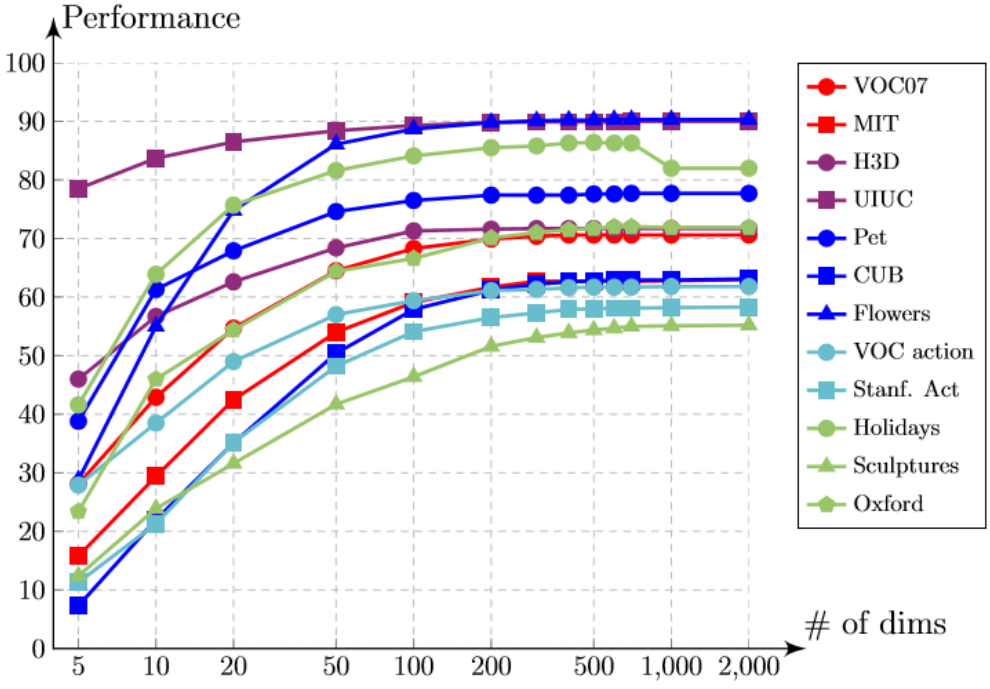
Spatial Pooling



- 大部分的情形，2x2的spatial pooling就夠用了。
- 包含複雜細節或是紋路繁複的圖，需要大的pooling才能得到較好的效果，如Sculpture dataset.



Dimensionality Reduction

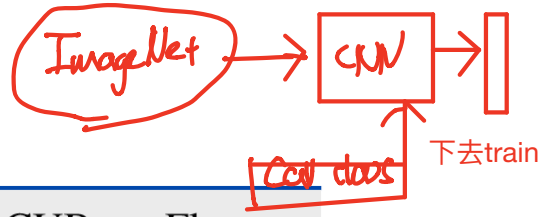


降低維度，訓練速度快

- 做PCA降維可以減少Feature的維度，來增加運算速度
- 幾乎所有的Target Dataset都在500維之後效果就沒太多提升
- 以原始4096維的Feature來說，降到500維會是平均表現維持較佳的維度

Fine-tuning

給一個預訓練模型使用別人訓練過模型



	Representation	MIT	CUB	Flower
Original FC7 feature	Medium FC7	65.9	62.9	90.4
FT: Find-tune feature 微調	Medium FT	**66.3**	**66.4**	**91.4**

與pretrain不同

勝！

- Fine-tuning is done by initializing a network with weights optimized for ILSVRC12 and then updating the network's weights using the target task training set.
- The learning rate used for fine-tuning is typically set to be less than the initial learning rate used to optimize the ConvNet for ILSVRC12.
  - This ensures the features learnt from the larger dataset are not forgotten.

Increasing Training Data 增加訓練資料

- To measure the PASCAL VOC 2007 object detection, fine-tune the AlexNet network using samples from the Oxford Pet and Caltech-UCSD birds datasets.
- Even though there already exists a large number of samples for those classes in ImageNet (more than 100,000 dogs), adding around 3000 dogs from the Oxford Pet dataset improves performance significantly.

Representation	bird	cat	dog
ConvNet [14]	38.5	51.4	46.0
ConvNet-FT VOC [14]	50.0	60.7	56.1
ConvNet-FT VOC+CUB+Pet	**51.3**	**63.0**	**57.2**

Different Classifiers

Classifier	VOC07	MIT	Pet
Linear SVM	72.8	63.7	79.6
Logistic regression	71.7	63.6	79.5
Neural Network ReLU	-	61.3	81.9
Neural Network exponentiated	-	63.0	80.4
Perceptron (no ReLU, no dropout)	-	59.6	78.2

- 幾乎各個分類器在不同情況下得到的效果都不太一樣

Optimized Accuracy

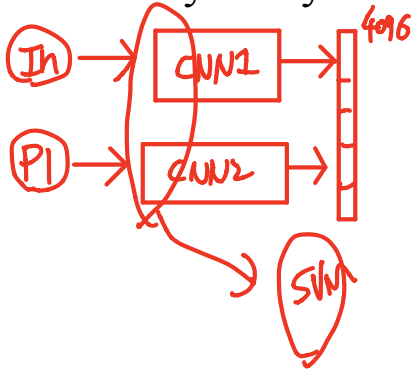
	Image Classification			Attribute Detection			Fine-grained Recognition			Compositional			Instance Retrieval				
	VOC07	MIT	SUN	SunAtt	UIUC	H3D	Pet	CUB	Flower	VOCa.	Act40	Phrase	Holid.	UKB	Oxf.	Paris	Scul.
non-ConvNet	[41] 71.1	[27] 68.5	[47] 37.5	[32] 87.5	[45] 90.2	[53] 69.1	[31] 59.2	[12] 62.7	[21] 90.2	[30] 69.6	[49] 45.7	[36] 41.5	[43] 82.2	[54] 89.4	[43] **81.7**	[43] 78.2	[4] 45.4
Deep Standard	71.8	64.9	49.6	91.4	90.6	73.8	78.5	62.8	90.5	69.2	58.9	77.3	86.2	93.0	73.0	81.3	53.7
Deep Optimized [d]	**80.7**	**71.3**	**56.0**	**92.5**	**91.5**	**74.6**	**88.1**	**67.1**	**91.3**	**74.3**	**66.4**	**82.3**	**90.0**	**96.3**	79.0	**85.1**	**67.9**
Err. Reduction	32%	18%	13%	13%	10%	4%	45%	12%	8%	17%	18%	22%	28%	47%	22%	20%	31%
Source Task	ImgNet	Hybrid	Hybrid	Hybrid	ImgNet	ImgNet	ImgNet	ImgNet	ImgNet	ImgNet	ImgNet	ImgNet	Hybrid	ImgNet	ImgNet	ImgNet	ImgNet
Network Width	Medium	Medium	Medium	Medium	Large	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Medium	Medium
Network Depth	16	8	8	8	8	16	16	16	16	16	16	16	8	8	16	16	16
Rep. Layer	last	last	last	last	2nd last	2nd last	2nd last	3rd last	3rd last	3rd last	3rd last	3rd last	4th last	4th last	4th last	4th last	4th last
PCA	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓	✓	✓
Pooling	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	1×1	1×1	2×2	2×2	3×3

How to Design and Optimize your Network?

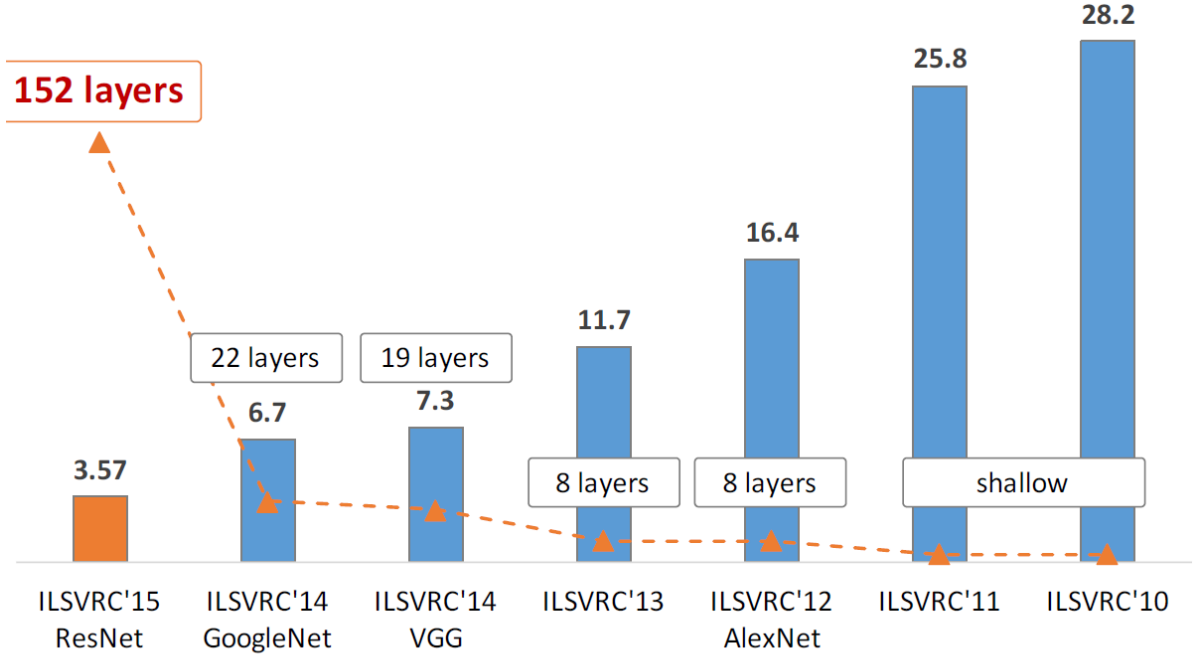
Factor	Target task: Source task ImageNet	...	FineGrained recognition	...	Instance retrieval
Early stopping	接近越深越好 Don't do it				
Network depth	As deep as possible				
Network width	Wider				Moderately wide →
Diversity/Density	More classes better than more images per class				
Fine-tuning	Yes, more improvement with more labelled data				
Dim. reduction	Original dim				Reduced dim →
Rep. layer	Later layers				Earlier layers →

Summary

- Factors of transferability
  - Range of Target Tasks –the relationships of pre-train/target model
  - Network Width versus Network Depth
  - Data Density versus Data Diversity
  - Feature Representation – which layer? hybrid? concatenate?
  - Spatial Pooling
  - Dimension Reduction
  - Early Stopping

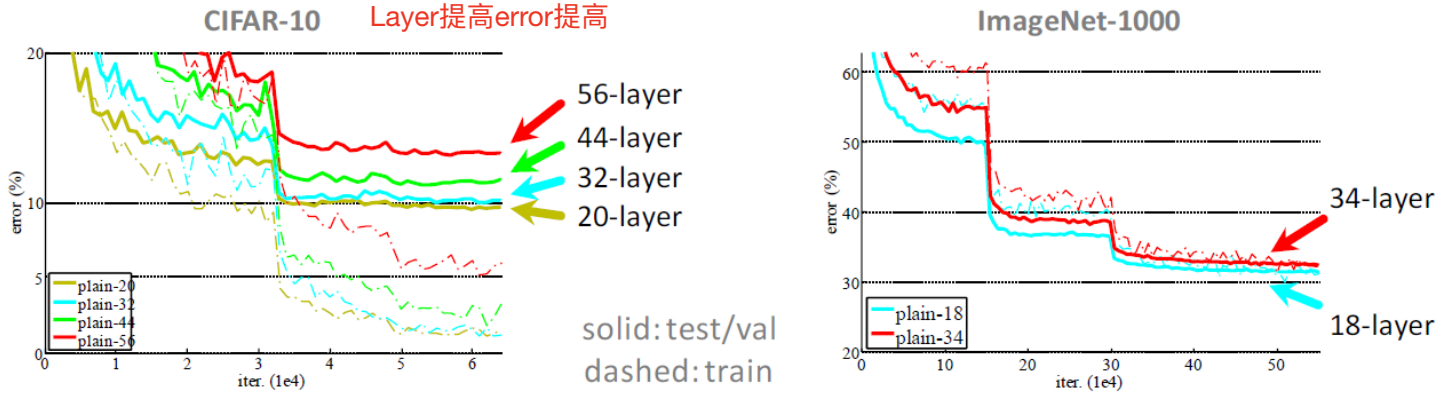


The Revolution of Depth Increasing

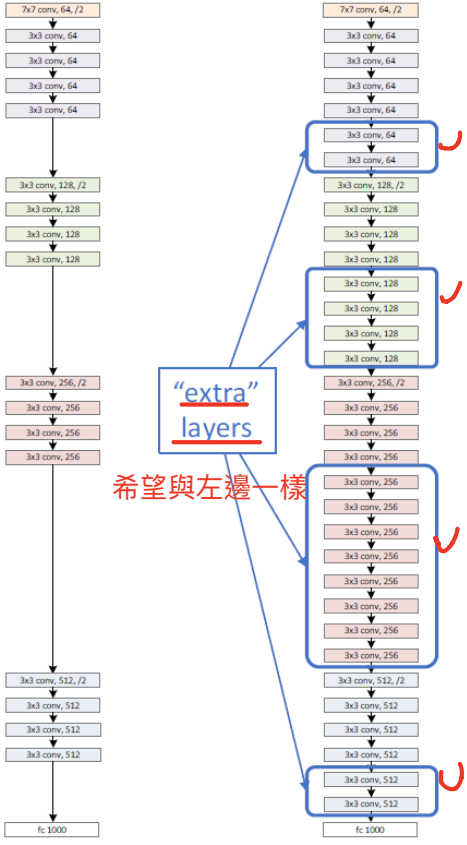


Simply stacking layers?

- *Plain* nets: stacking 3x3 conv layers…
- 56-layer net has **higher training error** and test error than 20-layer net
- "Overly deep" plain nets have **higher training error**
- A general phenomenon, observed in many datasets



a shallower model (18 layers)



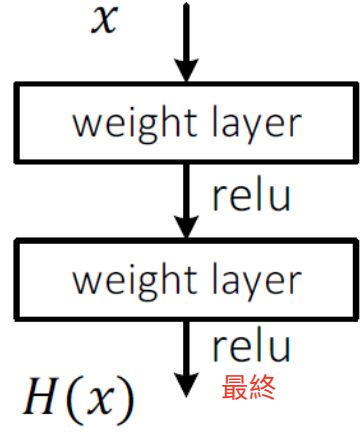
a deeper counterpart (34 layers)

- Richer solution space
- A deeper model should not have **higher training error** 多layer與左邊差不多
- A solution *by construction*: 需要學習更多
  - original layers: copied from a learned shallower model
  - extra layers: set as identity
  - at least the same training error
    較深不一定叫多train err
- Optimization difficulties: solvers cannot find the solution when going deeper…

Deep Residual Learning

- Plaint net

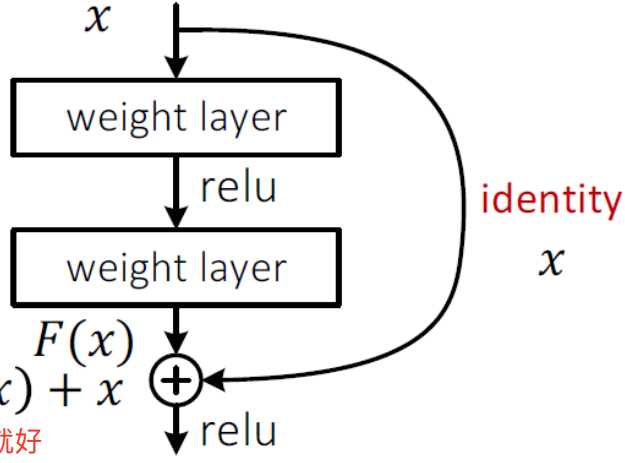
any two stacked layers



- H x is any desired mapping,
- Hope the 2 weight layers fit $H(x)$

模型學習減輕很多

- Residual net



- Hope the 2 weight layers fit $F(x)$
- If identity were optimal, easy to set weights as 0
- If optimal mapping is closer to identity, easier to find small fluctuations

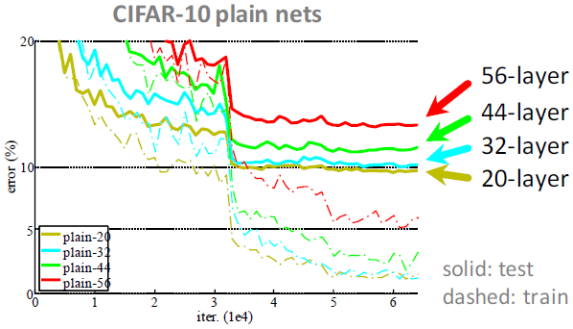
Network Design

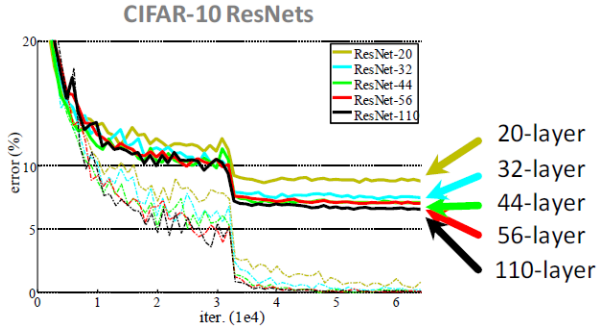
- Keep it simple
- Basic design (VGG-style)
  - all 3x3 conv (almost)
  - spatial size /2 => # filters x2 (~same complexity per layer)
  - Simple design, just deep!

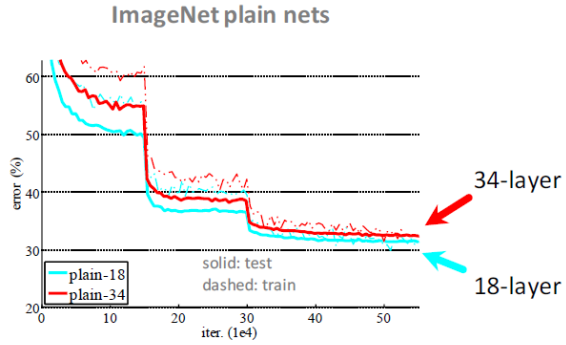
可以讓學習負擔降低很多

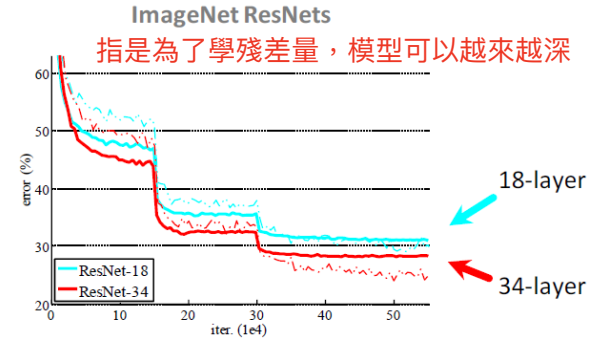
Error Measure

- Deeper ResNets have **lower training error**, and also lower test error

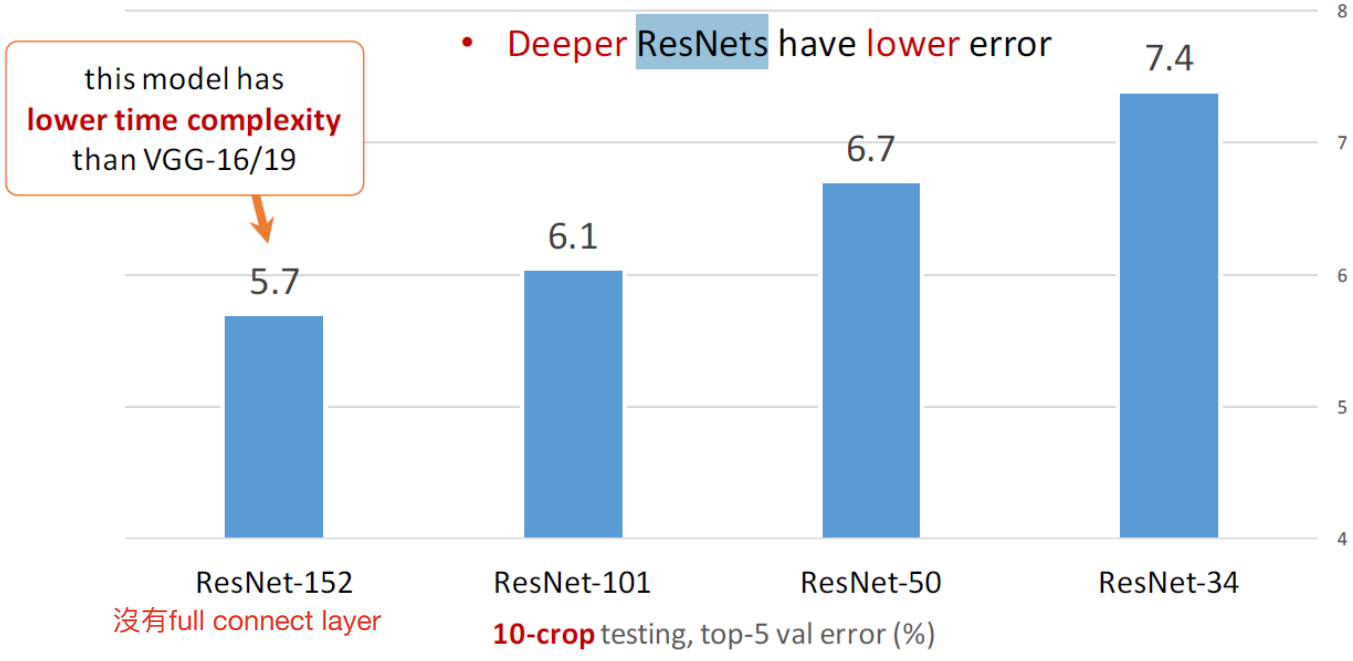






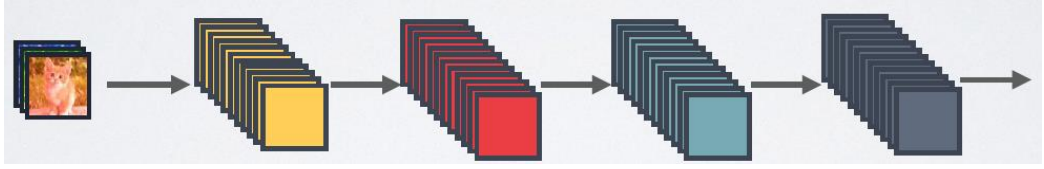


ResNets Go Deeper…



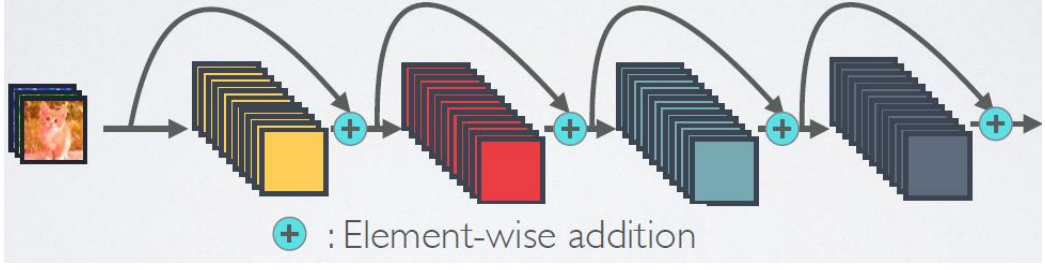
Densely Connected Convolutional Networks

- Network connection
  - Standard connection



  - ResNet

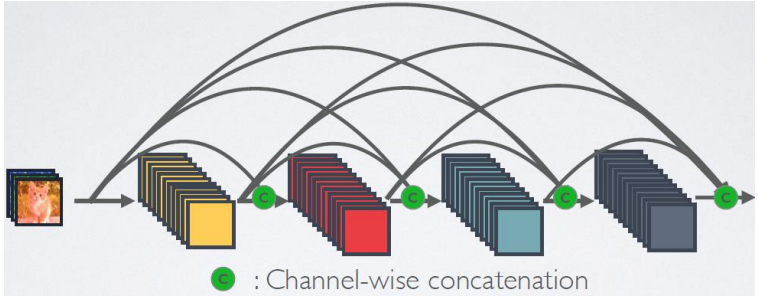
連結線一條

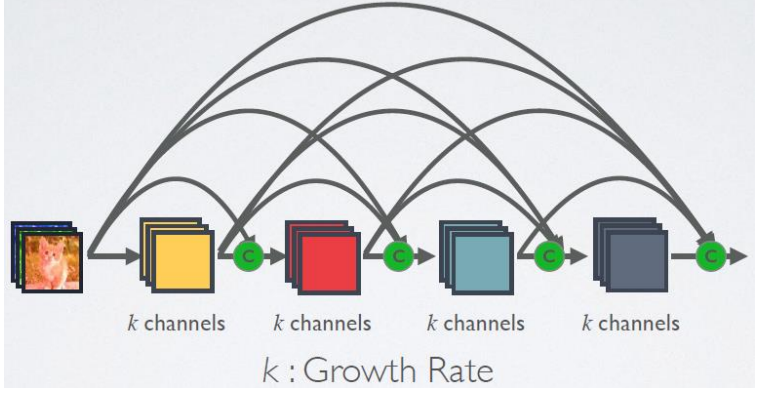


Densely Connected Convolutional Networks

- DenseNet

  連結線很密

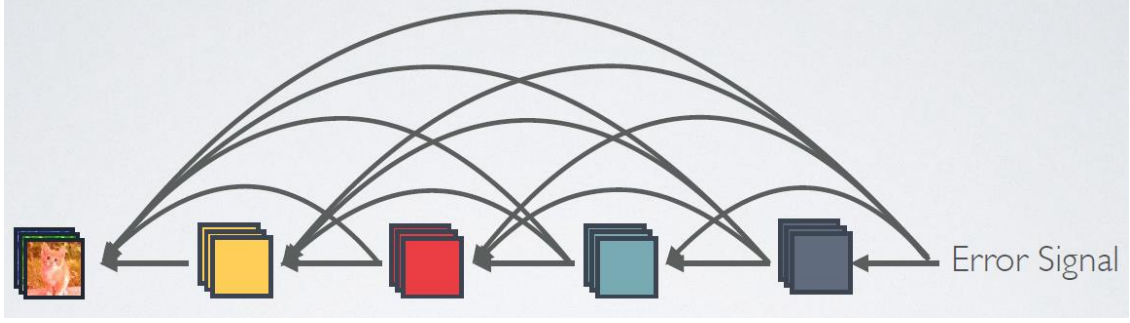




Advantages of DenseNet

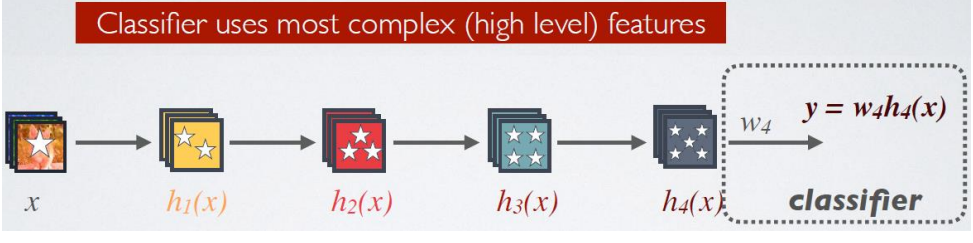
- Strong gradient flow
  - Alleviate the vanishing-gradient problem
- Strengthen feature propagation.

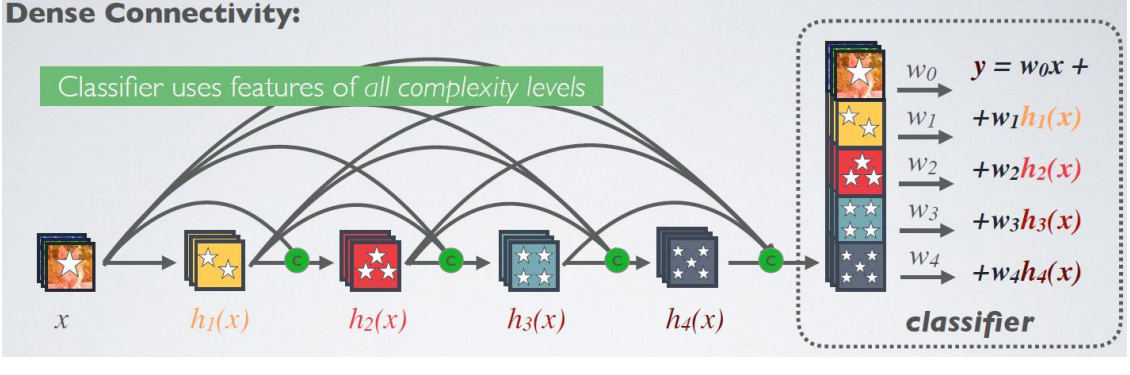
殘差往後加
低階特徵往後加，包含中階與高階特增
強化特徵傳遞功能



Advantages of DenseNet

- Encourage feature reuse





Reference & Slides Courtesy

- Deep Residual Learning for Image Recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun, CVPR 2016.
- ICML 2016 Tutorial – Deep Residual Network, Kaiming He
- Densely Connected Convolutional Networks, Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, CVPR 2017.
- Factors of Transferability for a Generic ConvNet Representation, Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, Stefan Carlsson, 2015 TPAMI.