經濟部工業局

AI產業實戰應用人才淬煉計畫

人工智慧及資安應用實務訓練課程

# 卷積神經網路 (CNN)

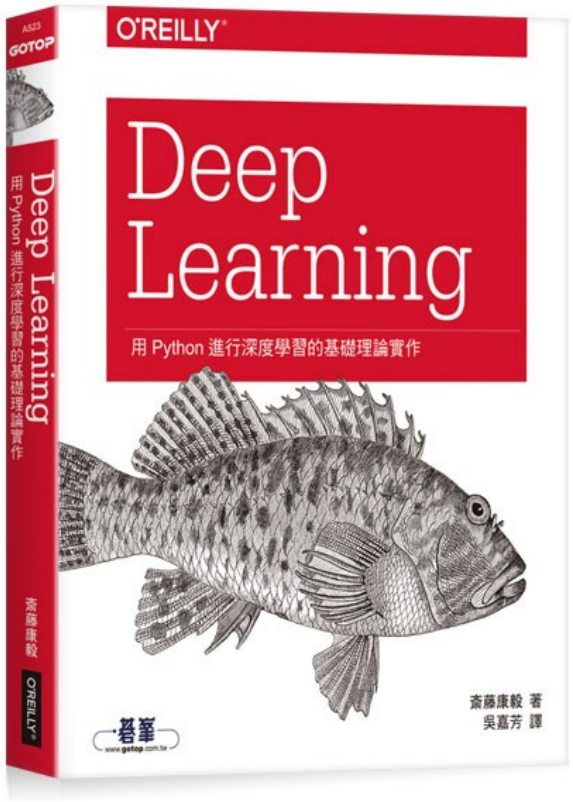
Convolutional Neural Network

# Agenda

- 卷積神經網路CNN
- CNN的應用
- **CNN歷史進展**
- **Transfer Learning遷移學習**
- CNN in security

# 卷積神經網路CNN

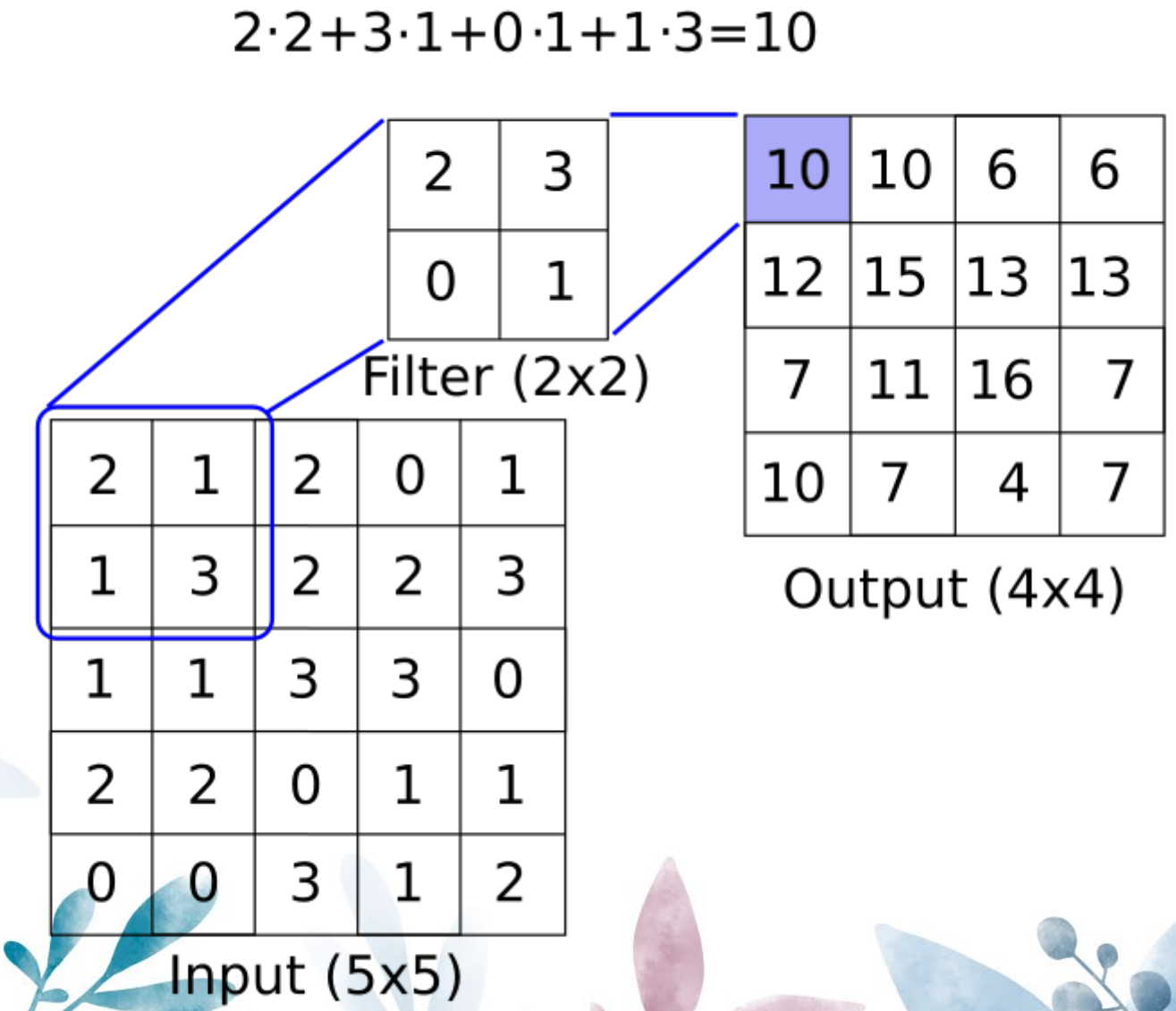
Deep Learning：用Python進行深度學習的基礎理論實作
作者： 斎藤康毅 譯者： 吳嘉芳
出版社：歐萊禮 出版日期：2017/08/17



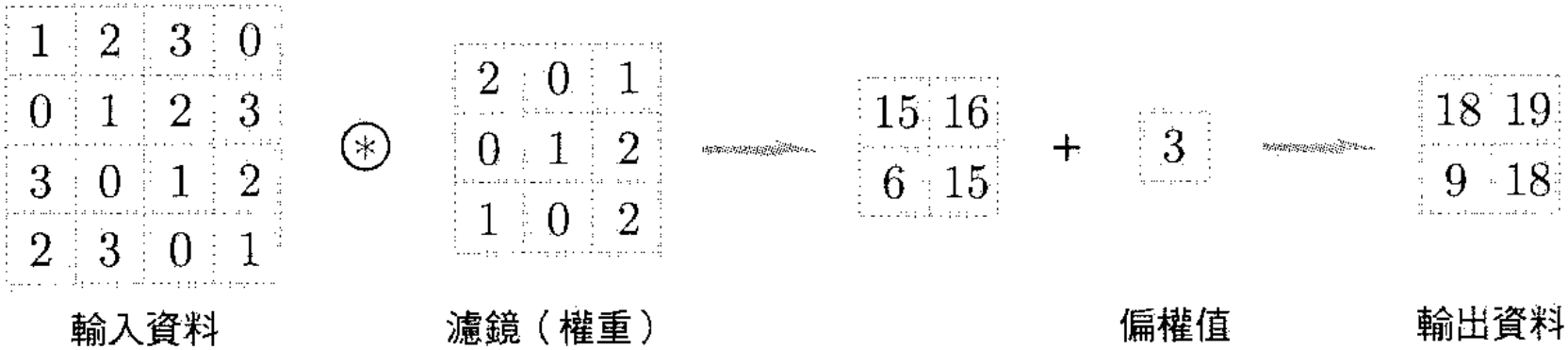

第一章 Python入門
第二章 感知器
第三章 神經網路
第四章 神經網路的學習
**第五章 誤差反向傳播法**
第六章 與學習有關的技巧
**第七章 卷積神經網路**
第八章 深度學習
附錄A Softmax-with-Loss層的計算圖
參考文獻

https://github.com/oreilly-japan/deep-learning-from-scratch

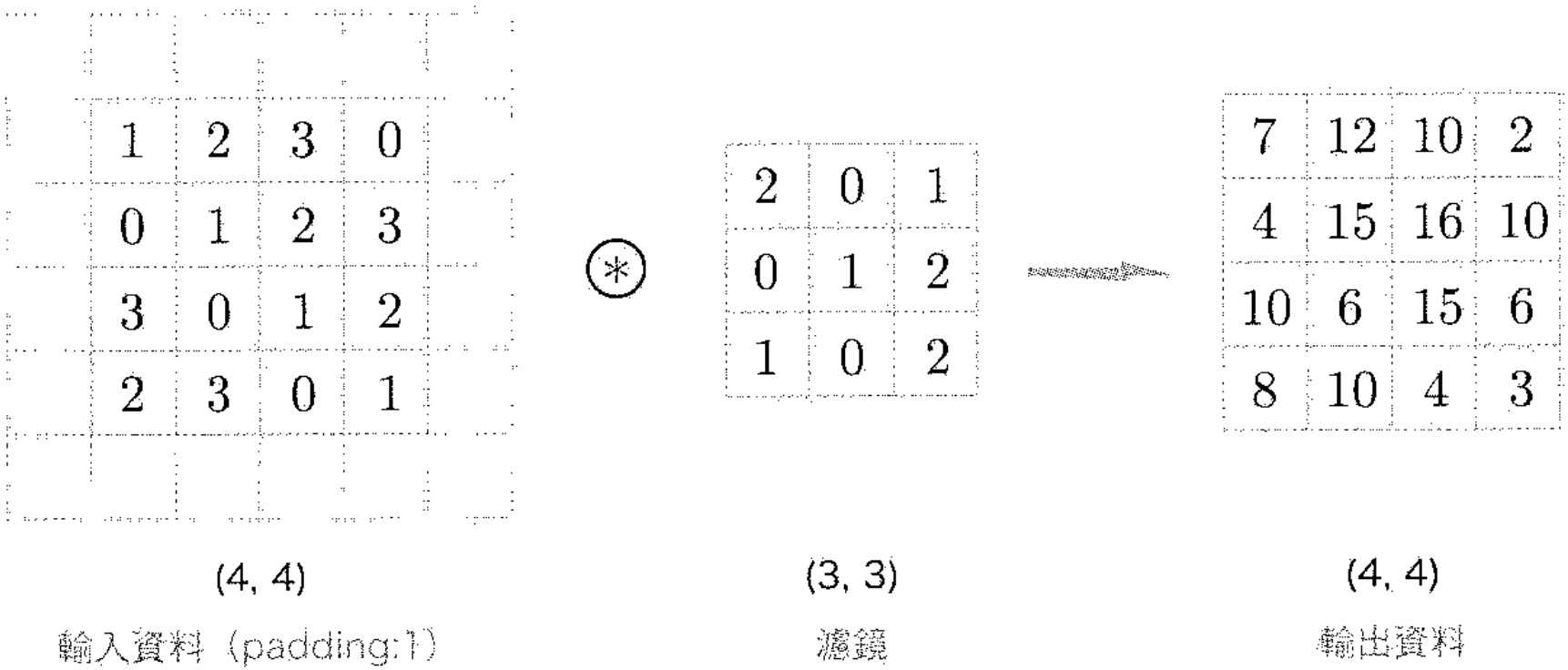
# Convolution運算



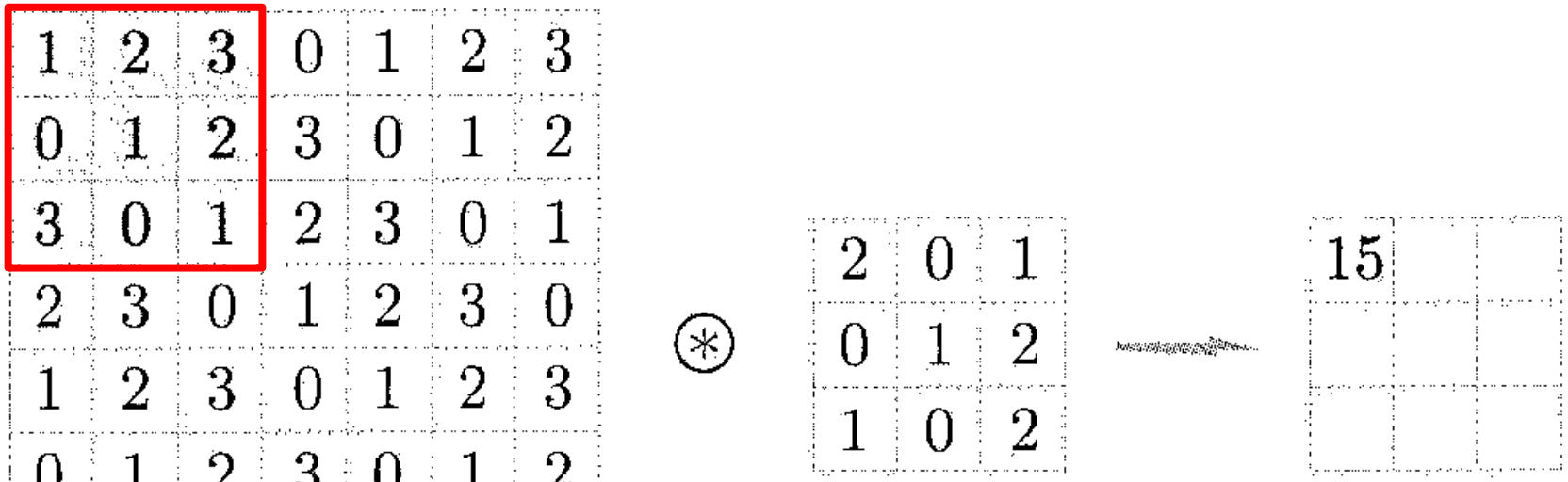
# Convolution運算



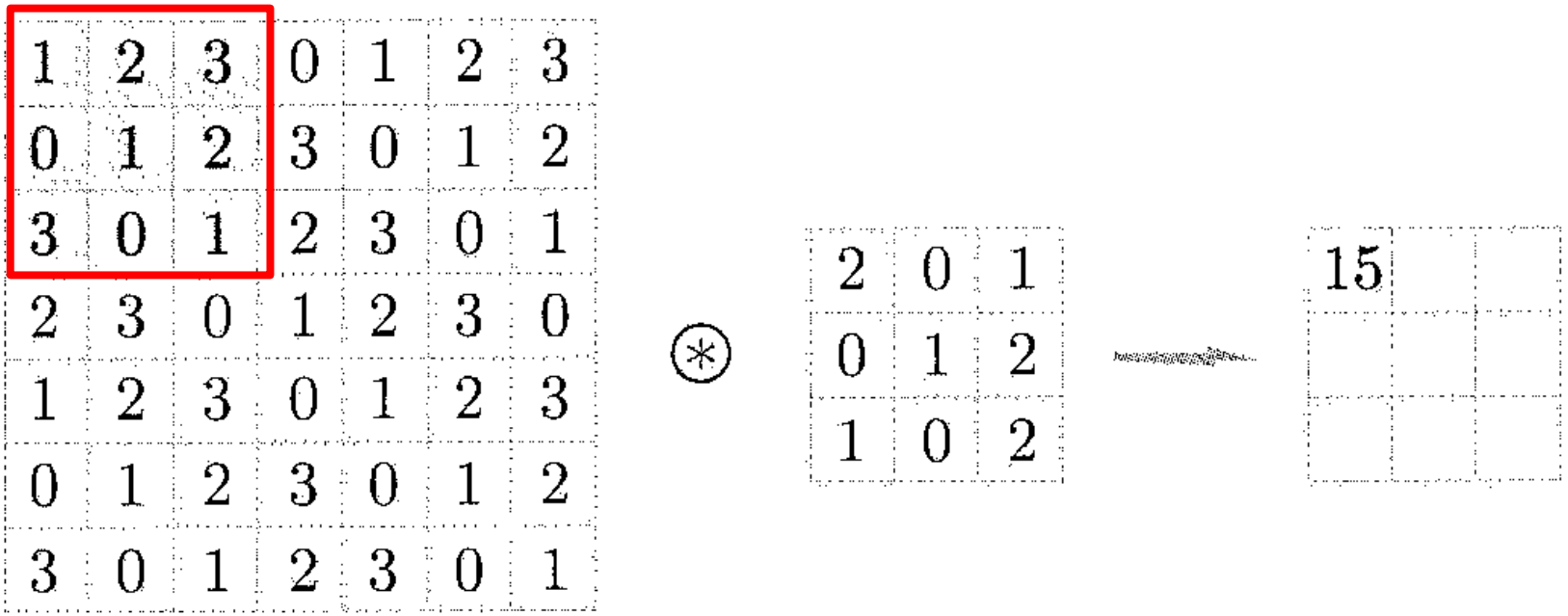
# Convolution運算:padding(填補)



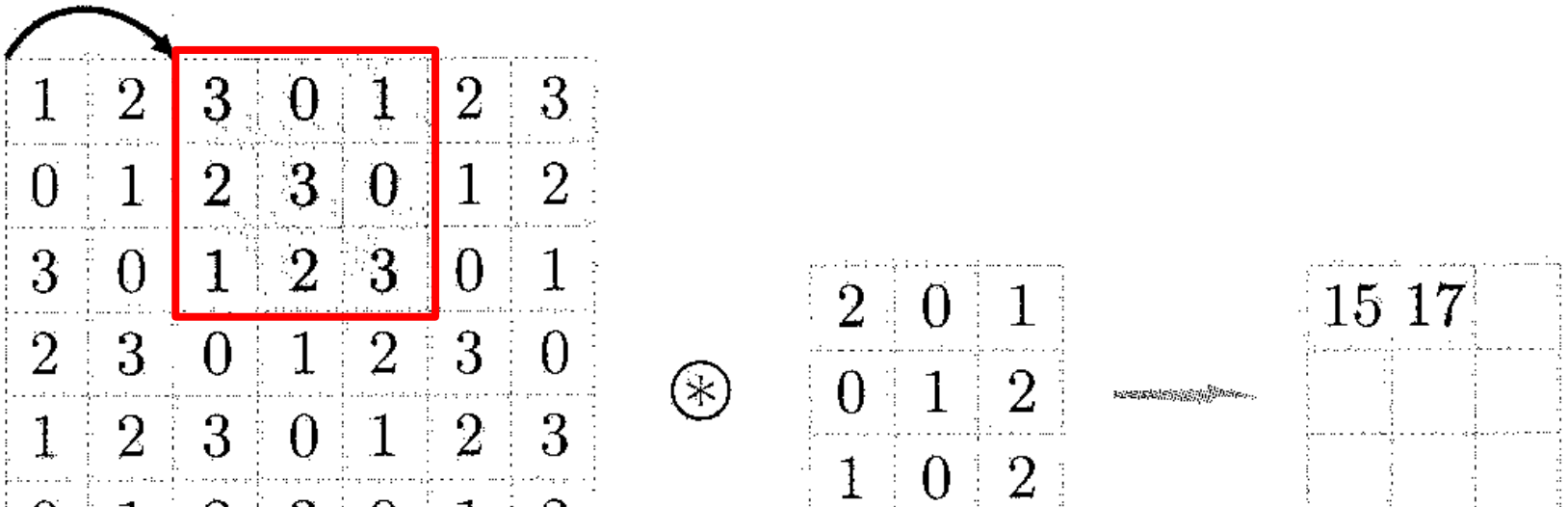
# Convolution運算:stride(步幅)



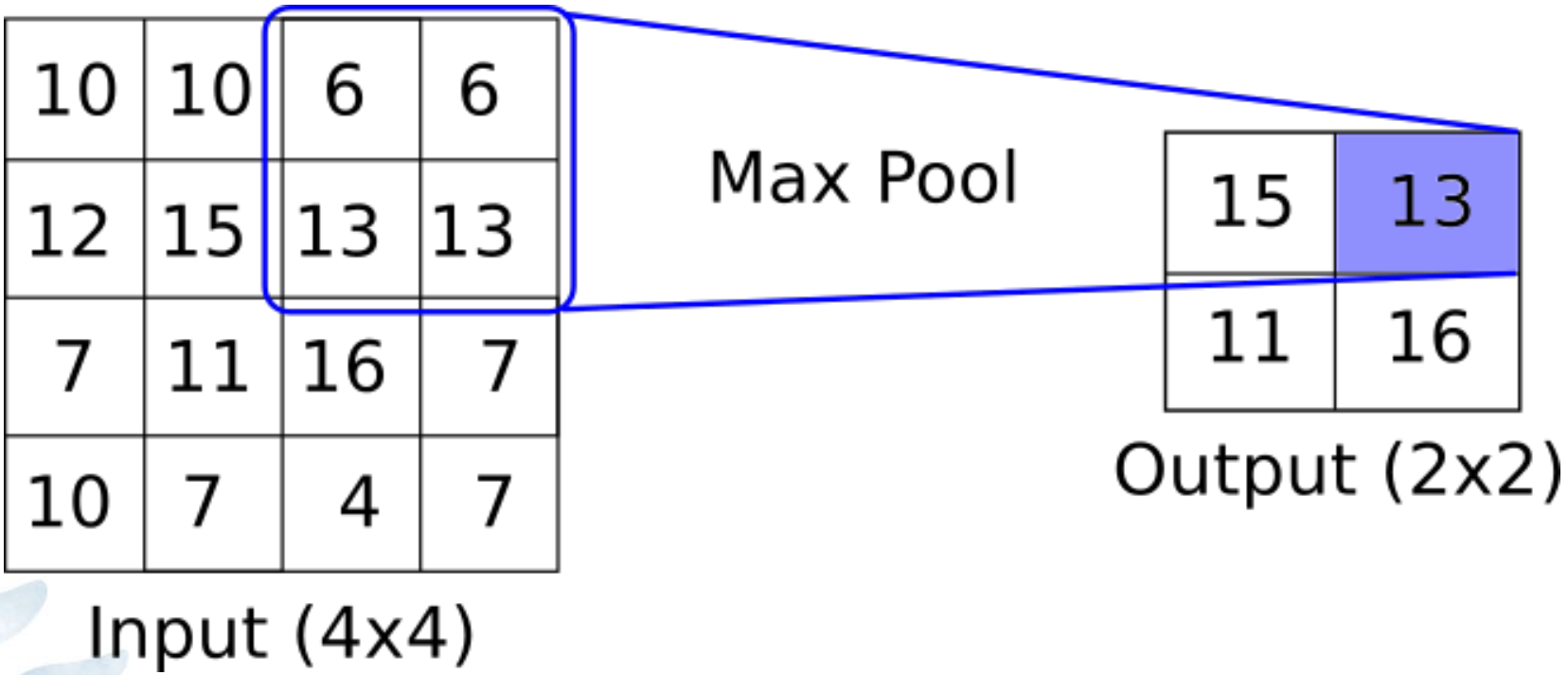
# Convolution運算:stride(步幅)




步幅：2



# MaxPooling運算



https://www.tensorflow.org/api_docs/python/tf/nn/max_pool

```
tf.nn.max_pool(
    value,
    ksize,
    strides,
    padding,
    data_format='NHWC',
    name=None
)
```

padding: A string, either 'VALID' or 'SAME'

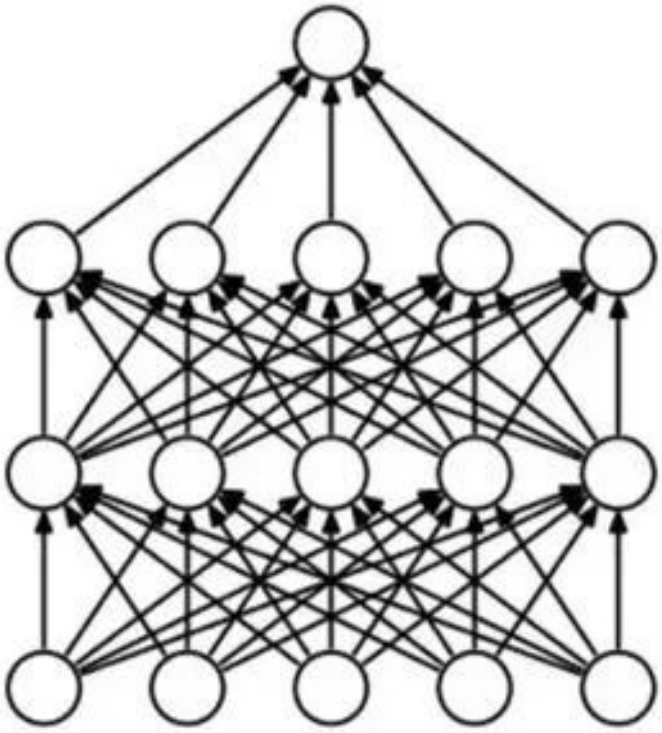
data_format: A string. 'NHWC', 'NCHW' and 'NCHW_VECT_C' are supported

# Dropout

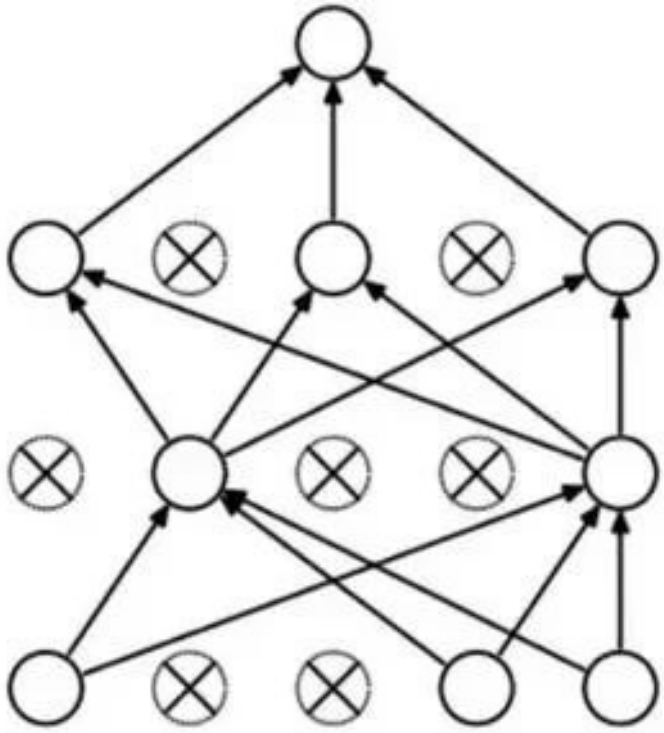
Dropout技術最早於2012年Hinton文章《ImageNet Classification with Deep Convolutional Neural Networks》中提出，在這篇文章中，Dropout技術確實提升了模型的性能，一般是添加到卷積神經網絡模型的全連接層中，使用深度學習工具箱實現起來很容易。

原文網址：https://kknews.cc/tech/rb48g6v.html

https://zh.wikipedia.org/wiki/Dropout



(a) Standard Neural Net



(b) After applying dropout.

http://jmlr.org/papers/v15/srivastava14a.html

# DropBlock: A regularization method for convolutional networks

**DropBlock: A regularization method for convolutional networks**

**Golnaz Ghiasi**
Google Brain

**Tsung-Yi Lin**
Google Brain

**Quoc V. Le**
Google Brain

14原文網址：
https://kknews.cc/tech/jebvrap.html

# https://arxiv.org/abs/1810.12890



**Computer Science > Computer Vision and Pattern Recognition**

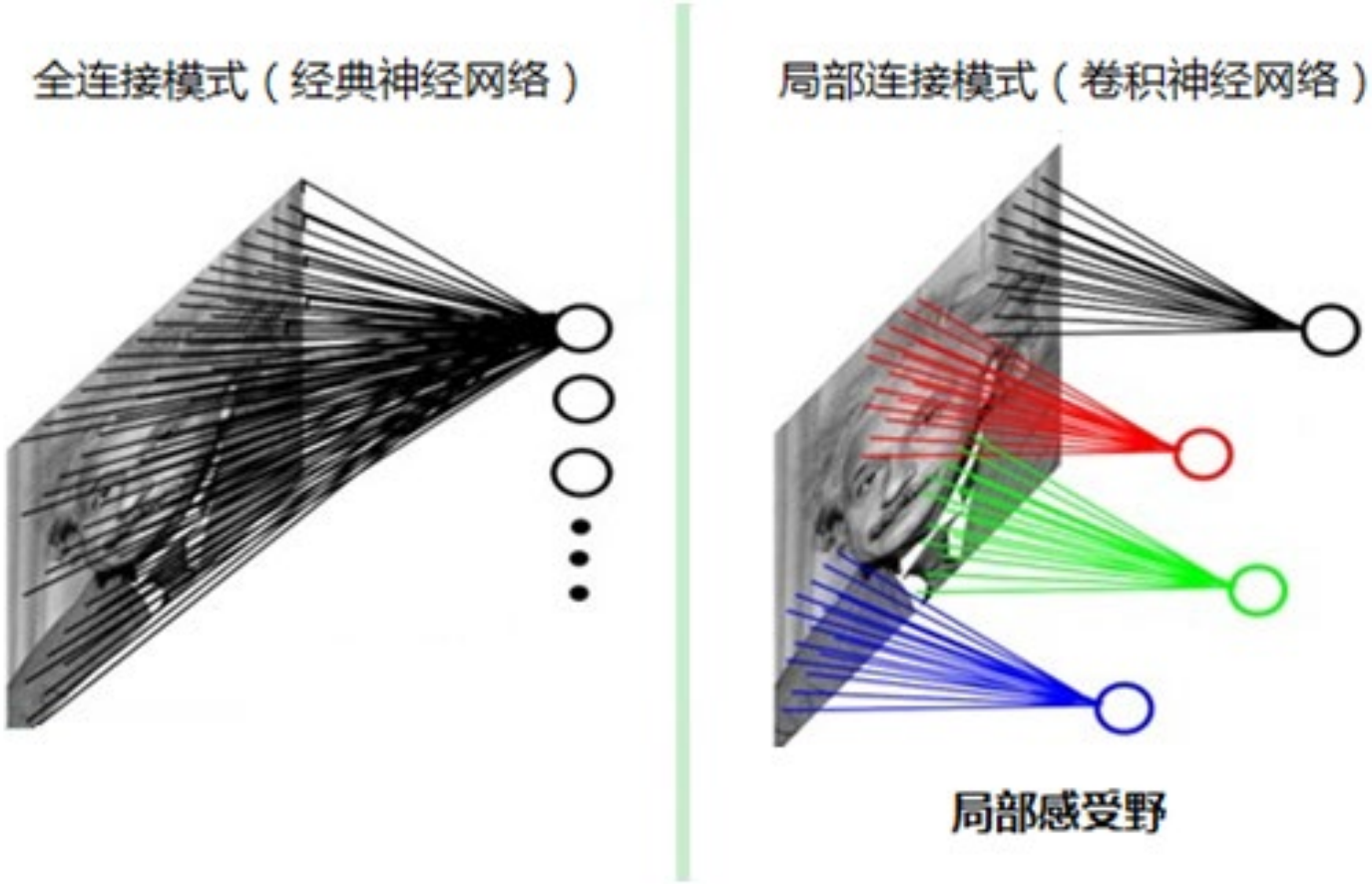
*[Submitted on 30 Oct 2018]*

## DropBlock: A regularization method for convolutional networks

Golnaz Ghiasi, Tsung-Yi Lin, Quoc V. Le

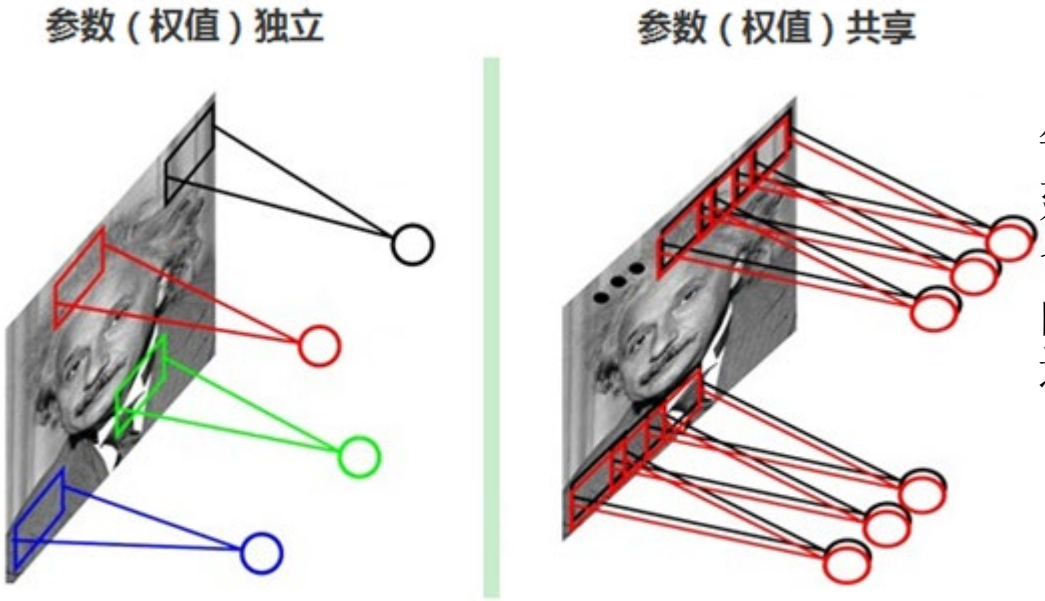
Deep neural networks often work well when they are over-parameterized and trained with a massive amount of noise and regularization, such as weight decay and dropout. Although dropout is widely used as a regularization technique for fully connected layers, it is often less effective for convolutional layers. This lack of success of dropout for convolutional layers is perhaps due to the fact that activation units in convolutional layers are spatially correlated so information can still flow through convolutional networks despite dropout. Thus a structured form of dropout is needed to regularize convolutional networks. In this paper, we introduce DropBlock, a form of structured dropout, where units in a contiguous region of a feature map are dropped together. We found that applying DropbBlock in skip connections in addition to the convolution layers increases the accuracy. Also, gradually increasing number of dropped units during training leads to better accuracy and more robust to hyperparameter choices. Extensive experiments show that DropBlock works better than dropout in regularizing convolutional networks. On ImageNet classification, ResNet-50 architecture with DropBlock achieves $78.13\%$ accuracy, which is more than $1.6\%$ improvement on the baseline. On COCO detection, DropBlock improves Average Precision of RetinaNet from $36.8\%$ to $38.4\%$.

# CNN的幾個特點：
# 局部感知、參數共用、池化




https://my.oschina.net/u/876354/blog/1632862

# CNN的幾個特點：
# 局部感知、參數共用、池化



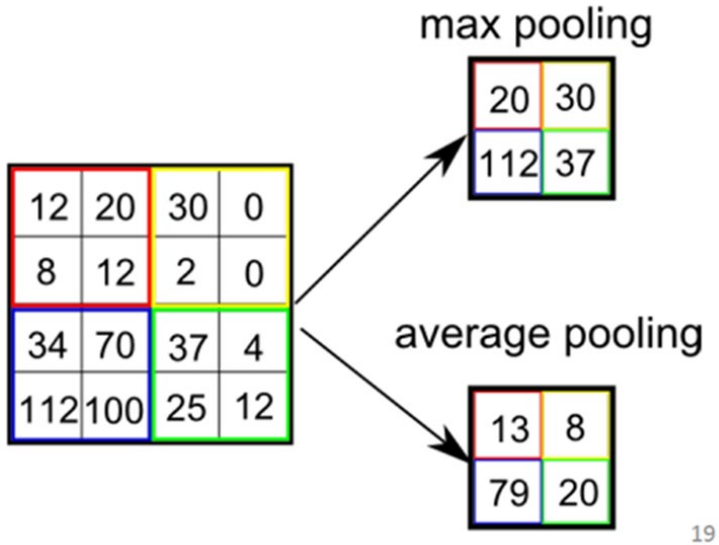

每張自然圖像（人物、山水、建築等）都有其固有特性，也就是說，圖像其中一部分的統計特性與其它部分是接近的。

這也意味著這一部分學習的特徵也能用在另一部分上，能使用同樣的學習特徵。

因此，在局部連接中隱藏層的每一個神經元連接的局部圖像的權值參數（例如5×5），將這些權值參數共用給其它剩下的神經元使用，那麼此時不管隱藏層有多少個神經元，需要訓練的參數就是這個局部圖像的許可權參數（例如5×5），也就是卷積核的大小，這樣大大減少了訓練參數

# CNN的幾個特點：
# 局部感知、參數共用、池化

隨著模型網路不斷加深，卷積核越來越多，要訓練的參數還是很多，而且直接拿卷積核提取的特徵直接訓練也容易出現**過擬合**的現象。

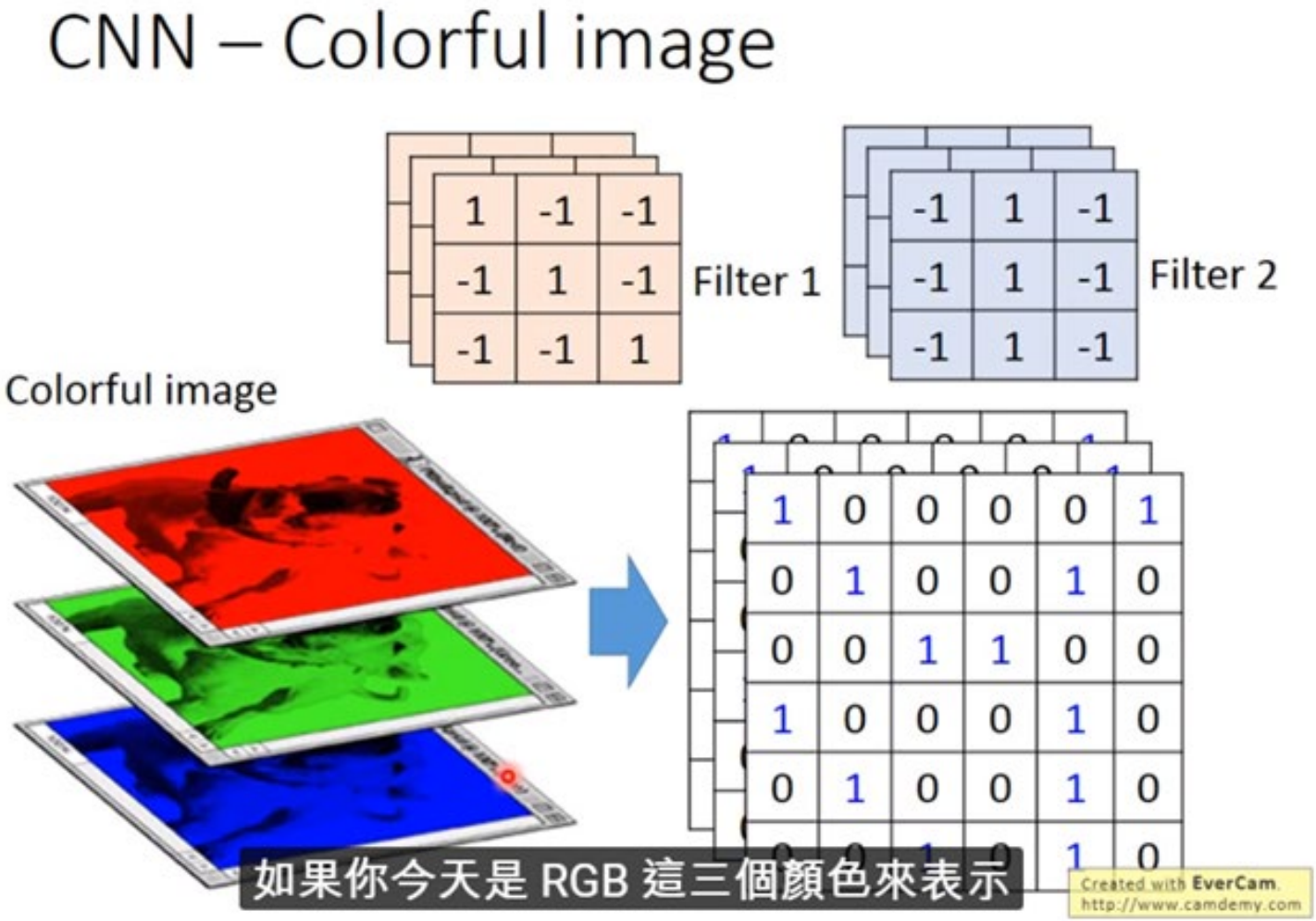




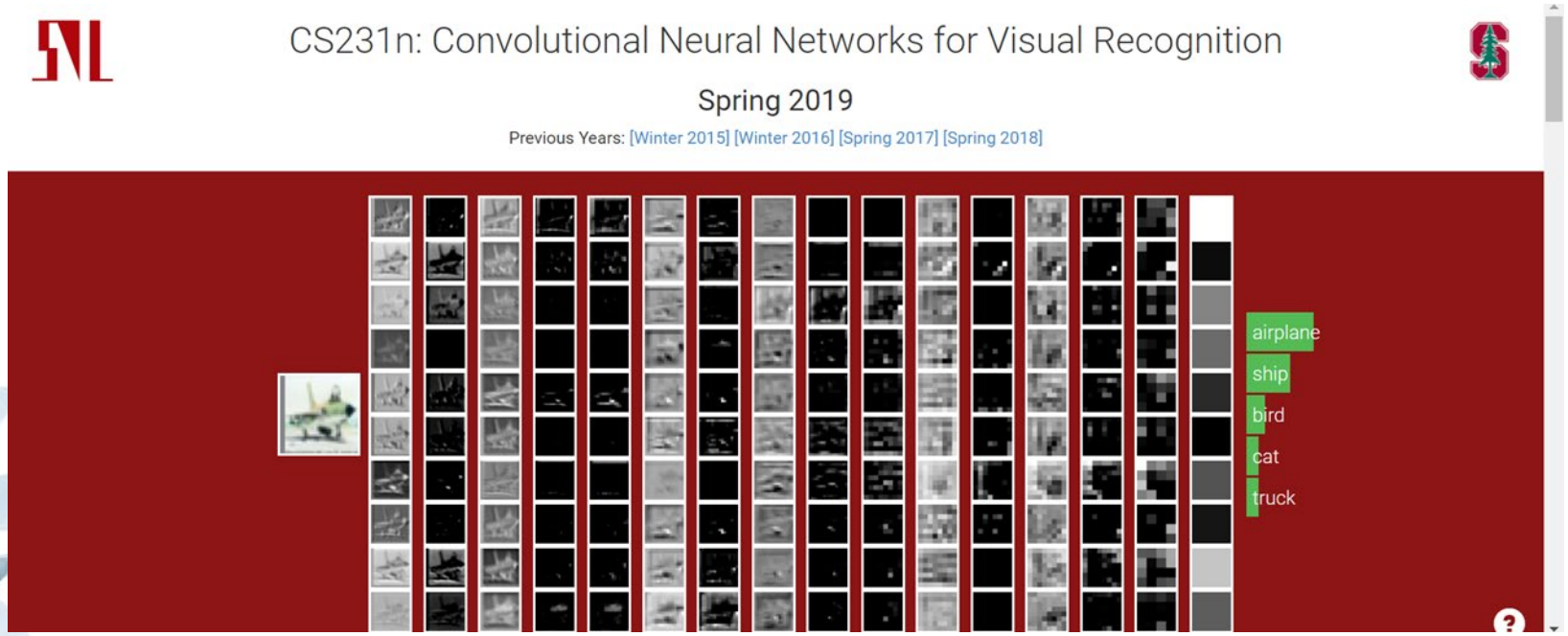

對圖像使用卷積提取特徵是因為圖像具有一種“靜態性”的屬性，因此，一個很自然的想法就是對不同位置區域提取出有**代表性的特徵（進行聚合統計，例如最大值、平均值等）**，這種聚合的操作就叫做池化，

池化的過程通常也被稱為特徵映射的過程（特徵降維）

ML Lecture 10: Convolutional Neural Network
https://www.youtube.com/watch?v=FrKWiRv254g



# http://cs231n.stanford.edu/



# CNN 與 電腦視覺

# Image processing

automatic processing, manipulation, analysis, and interpretation of images using algorithms and codes on a computer.




imageio

scikit-image
image processing in python


Python讀取視訊的兩種方法（imageio和cv2)
https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/357894/

# 使用矩陣儲存圖片

**Black & White**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

1 bit per pixel.

Values 0 – 1

Height x Width

**Grayscale**

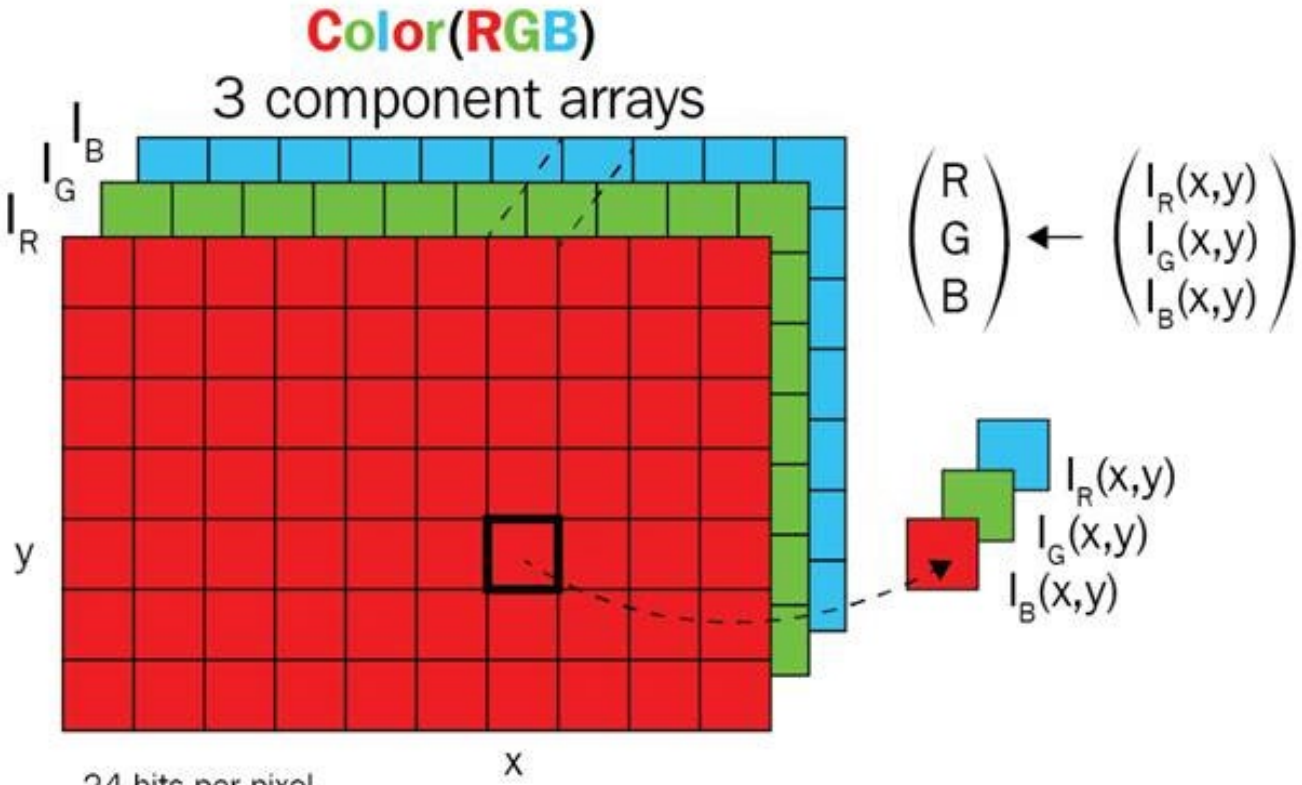
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 8 | 120 | 221 | 189 | 145 | 212 | 33 | 244 |
| 149 | 20 | 30 | 72 | 205 | 177 | 208 | 173 |
| 47 | 19 | 3 | 249 | 102 | 229 | 161 | 122 |
| 0 | 70 | 198 | 197 | 28 | 147 | 107 | 91 |
| 243 | 40 | 131 | 54 | 150 | 18 | 11 | 158 |
| 123 | 66 | 137 | 10 | 139 | 111 | 241 | 183 |
| 239 | 217 | 100 | 153 | 131 | 236 | 230 | 113 |
| 237 | 166 | 62 | 169 | 129 | 32 | 141 | 185 |

8 bits per pixel.

Values 0 – 255

Height x Width

# 使用矩陣儲存圖片




24 bits per pixel.

RB Red/Green/Blue

Height x Width X 3:

Python
Imaging
Library

# PIL/Pillow

https://pillow.readthedocs.io/en/stable/

安裝 **sudo pip install pillow**

Python Imaging Library

https://pillow.readthedocs.io/en/stable/

## 安裝 sudo pip install pillow

- PIL/Pillow 是 Python強大的圖像處理庫，功能包括：基本圖像操作（創建縮略圖、幾何變換、圖像裁剪、圖像分離與合併、粘貼圖片）圖像存儲、圖像顯示、格式轉換、截屏操作、圖像繪製功能、圖像濾鏡功能以及其他常用方法
- 他支援許多不同圖像檔案格式的open打開，操作和保存。
- PIL功能非常強大，但API卻非常簡單易用。
- 由於PIL僅支持到Python 2.7，加上年久失修，於是一群志願者在PIL的基礎上創建了相容的版本，名字叫Pillow，支持最新Python 3.x，又加入了許多新特性
- 適用於Windows，Mac OS X和Linux。

https://pillow.readthedocs.io/en/stable/handbook/tutorial.html

# 載入圖片

!wget https://raw.githubusercontent.com/XXXXXX/Tensorflow2/master/images/monalisa.jpg

```
from PIL import Image

im = Image.open("monalisa.jpg")

print(im.format, im.size, im.mode)
```

https://pillow.readthedocs.io/en/stable/handbook/tutorial.html

scikit-image
image processing in python

https://scikit-image.org/docs/stable/user_guide

# Scikit-image skimage

https://scikit-image.org/

https://scikit-image.org/docs/dev/auto_examples/

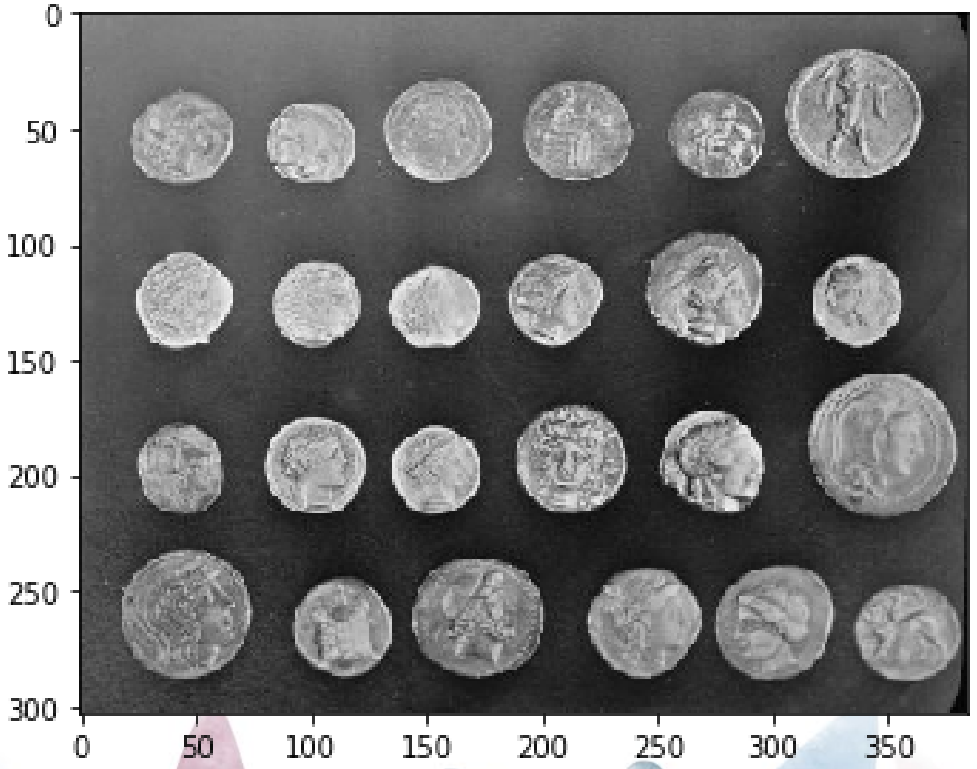
# 載入圖片

```
!wget
https://raw.githubusercontent.com/XXXXXX/Tensorflow2/master/images/monalisa.jpg
```

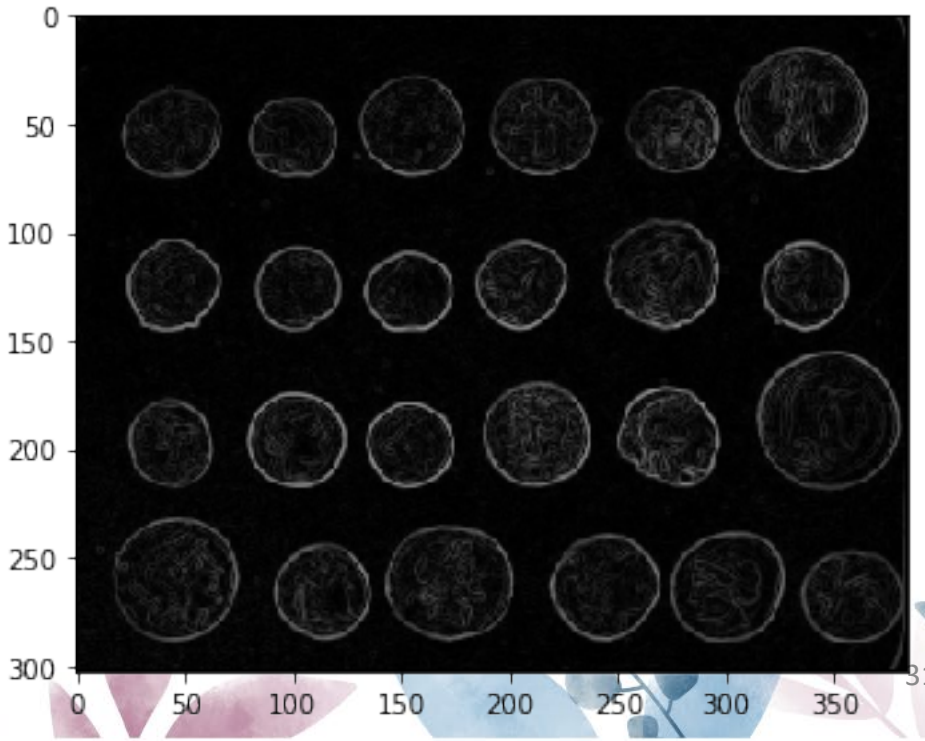
# 顯示圖片

```
from skimage import io

io.imshow('monalisa.jpg')
io.show()
```

```
from skimage import data, io

image = data.coins()

io.imshow(image)
io.show()
```



```
from skimage import data, io, filters

image = data.coins()
# ... or any other NumPy array!

edges = filters.sobel(image)

io.imshow(edges)
io.show()
```




# imageio

# imageio

https://imageio.readthedocs.io/en/stable/examples.html

**!pip install imageio**

產生**GIF** 图片

```
import imageio
im = imageio.imread('imageio:astronaut.png')
im.shape  # im is a numpy array
imageio.imwrite('astronaut-gray.jpg', im[:, :, 0])
```

https://imageio.readthedocs.io/en/stable/standardimages.html

Imageio provides a number of standard images.
These include classic 2D images, as well as animated and volumetric images.

The image names can be loaded by using a special URI, e.g. `imread('imageio:astronaut.png')`. The images are automatically downloaded (and cached in your appdata directory).

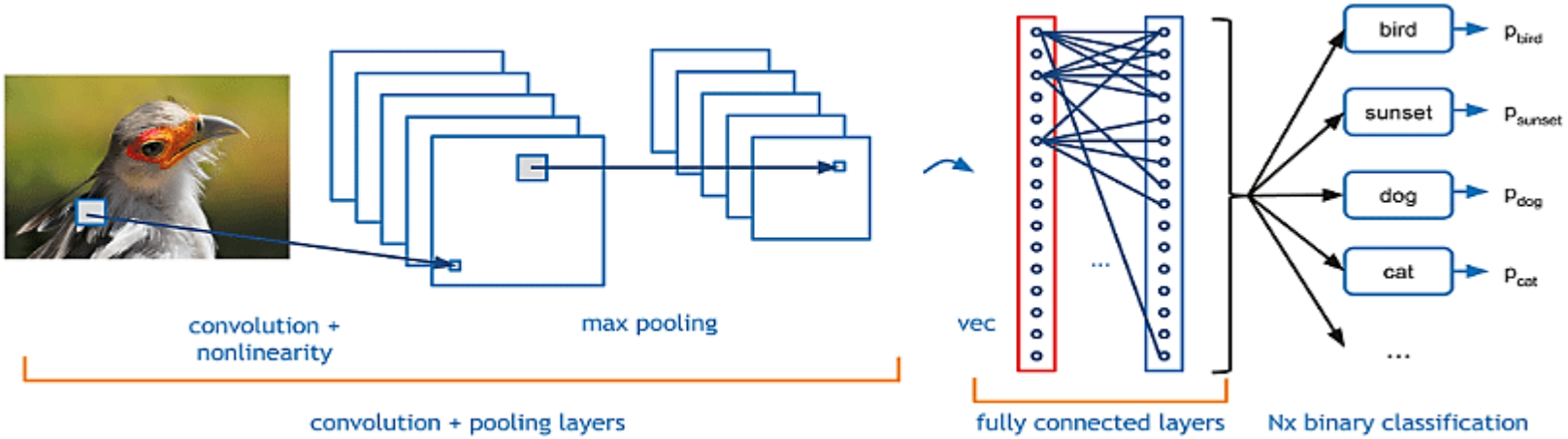
- chelsea.bsdf: The chelsea.png in a BSDF file(for testing)
- newtonscradle.gif: Animated GIF of a newton's cradle
- cockatoo.mp4: Video file of a cockatoo
- stent.npz: Volumetric image showing a stented abdominal aorta
- astronaut.png: Image of the astronaut Eileen Collins
- camera.png: Classic grayscale image of a photographer
- checkerboard.png: Black and white image of a chekerboard
- chelsea.png: Image of Stefan's cat
- clock.png: Photo of a clock with motion blur (Stefan van der Walt)
- coffee.png: Image of a cup of coffee (Rachel Michetti)
- coins.png: Image showing greek coins from Pompeii
- horse.png: Image showing the silhouette of a horse (Andreas Preuss)
- hubble_deep_field.png: Photograph taken by Hubble telescope (NASA)
- immunohistochemistry.png: Immunohistochemical (IHC) staining
- moon.png: Image showing a portion of the surface of the moon
- page.png: A scanned page of text
- text.png: A photograph of handdrawn text
- wikkie.png: Image of Almar's cat
- chelsea.zip: The chelsea.png in a zipfile (for testing)

# 執行看看有甚麼結果??

```
import numpy as np
import imageio
from PIL import Image

img=imageio.imread('monalisa.jpg')
high,width,ichannel=img.shape
print(type(img))
print(img.shape)

imageio.imwrite('leopard_i1.jpg',img)
imageio.imwrite('leopard_i2.jpg',np.float32(img/10))
 # automatic brightness adjust
imageio.imwrite('leopard_i3.jpg',np.uint8(img/10))
```

# 圖像分類Image Classification

給定單一類別標籤的圖像，要求為一組未曾見過的測試圖像預測圖像類別並衡量預測的準確性。

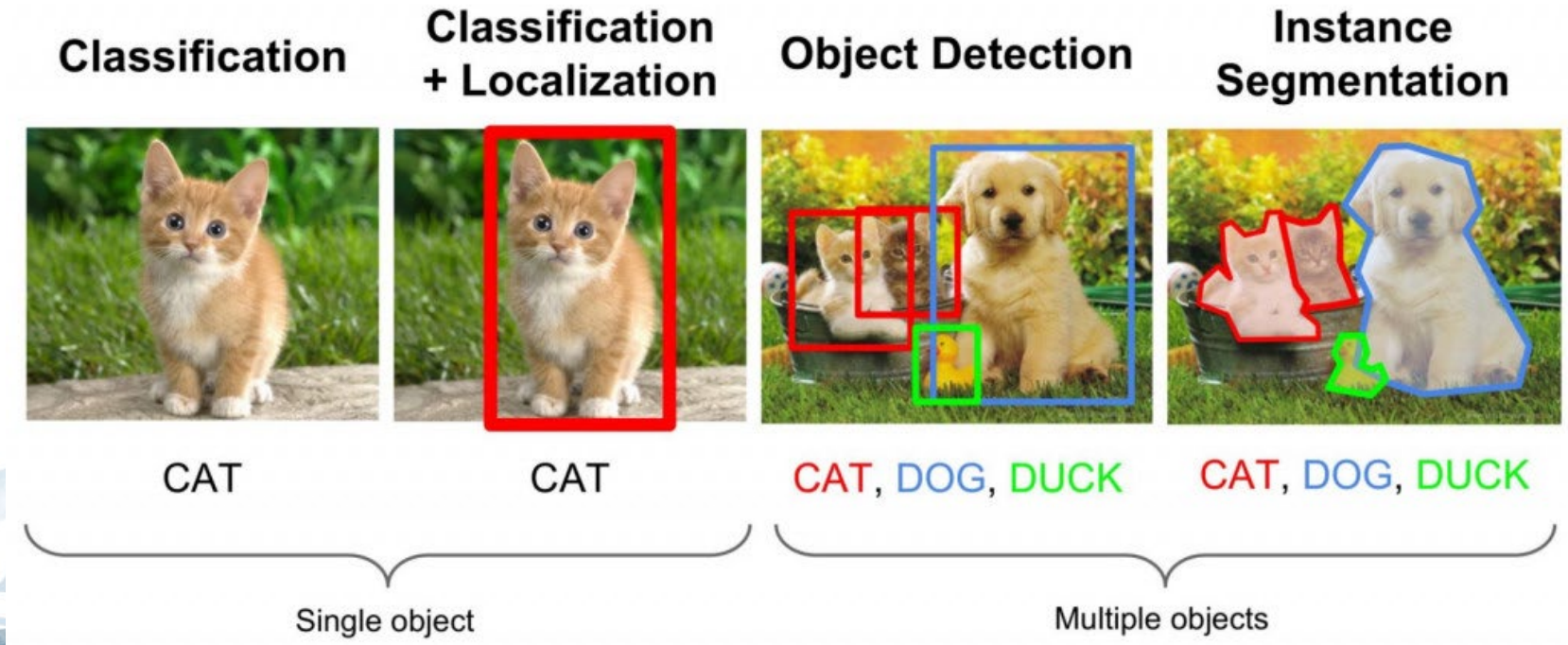



圖片來源：https://www.researchgate.net/figure/Deep-Convolutional-Neural-Network-CNN-for-classification-image-from-31_fig7_318277197

https://ithelp.ithome.com.tw/articles/10204738

# 目標偵測object detection

找出圖像或視頻中的感興趣目標，
同時實現輸出檢測目標的位置和類別




圖片來源：https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852

https://ithelp.ithome.com.tw/articles/10204738

Object Detection

# 常用資料集

PASCAL VOC 包含20個類別。
通常是用VOC07和VOC12的trainval並集作為訓練，
用VOC07的測試集作為測試。

MS COCO COCO比VOC更困難。
COCO包含80k訓練圖像、40k驗證圖像、和20k沒有公開標記的測試圖像(test-dev)，80個類別，平均每張圖7.2個目標。
通常是用80k訓練和35k驗證圖像的並集作為訓練，其餘5k圖像作為驗證，20k測試圖像用於線上測試。

# Awesome Object Detection

https://github.com/amusi/awesome-object-detection

https://handong1587.github.io/deep_learning/2015/10/09/object-detection.html

R-CNN
Fast R-CNN
Faster R-CNN
Mask R-CNN
Light-Head R-CNN
Cascade R-CNN
SPP-Net
YOLO
YOLOv2
YOLOv3
YOLT

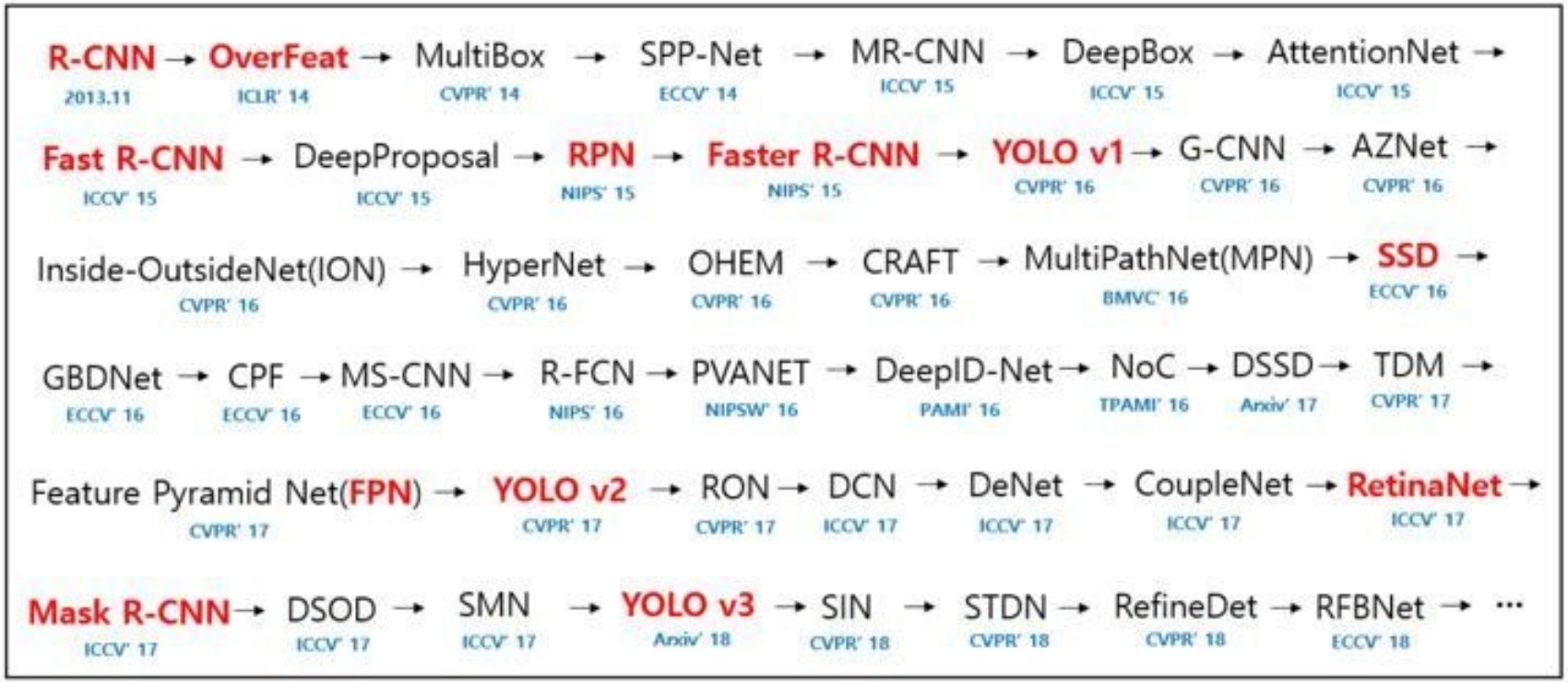
SSD
DSSD
FSSD
ESSD
MDSSD
Pelee
Fire SSD
R-FCN
FPN
DSOD

RetinaNet
MegDet
RefineNet
DetNet
SSOD
CornerNet
M2Det
3D Object Detection
ZSD（Zero-Shot Object Detection）
OSD（One-Shot object Detection）
Weakly Supervised Object Detection

2013-----RCNN----Rich feature hierarchies for accurate object detection and semantic segmentation
2013----Deep Neural Networks for Object Detection
2014----SPPnet----Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition-sppnet
2015----Fast R-CNN
2016----Faster R-CNN-Towards Real-Time Object Detection with Region Proposal Networks
2016----Inside-Outside Net_Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks
2016----R-FCN-Object Detection via Region-based Fully Convolutional Networks
2016----SSD-Single Shot MultiBox Detector
2016----YOLO9000-better,faster,stronger
2016----You Only Look Once-Unified, Real-Time Object Detection
2017----A-Fast-RCNN_Hard Positive Generation via Adversary for Object Detection
2017----Deformable Convolutional Networks
2017----DSOD_ Learning Deeply Supervised Object Detectors from Scratch
2017----Focal Loss for Dense Object Detection
2017----Mask R-CNN
2017----Speed_Accuracy trade-offs for modern convolutional object detectors
2017----Light-Head R-CNN_In Defense of Two-Stage Object Detector
2018----CornerNet：Detecting Objects as Paired Keypoints

# 目標偵測領域重要論文

paper list from 2014 to now(2018)



Recent Advances in Deep Learning for Object Detection
intro: From 2013 (OverFeat) to 2019 (DetNAS)
arXiv: https://arxiv.org/abs/1908.03673

A Survey of Deep Learning-based Object Detection
intro：From Fast R-CNN to NAS-FPN
arXiv：https://arxiv.org/abs/1907.09408

Object Detection in 20 Years: A Survey
arXiv：https://arxiv.org/abs/1905.05055

《Recent Advances in Object Detection in the Age of Deep Convolutional Neural Networks》
arXiv: https://arxiv.org/abs/1809.03193

《Deep Learning for Generic Object Detection: A Survey》
arXiv: https://arxiv.org/abs/1809.02165

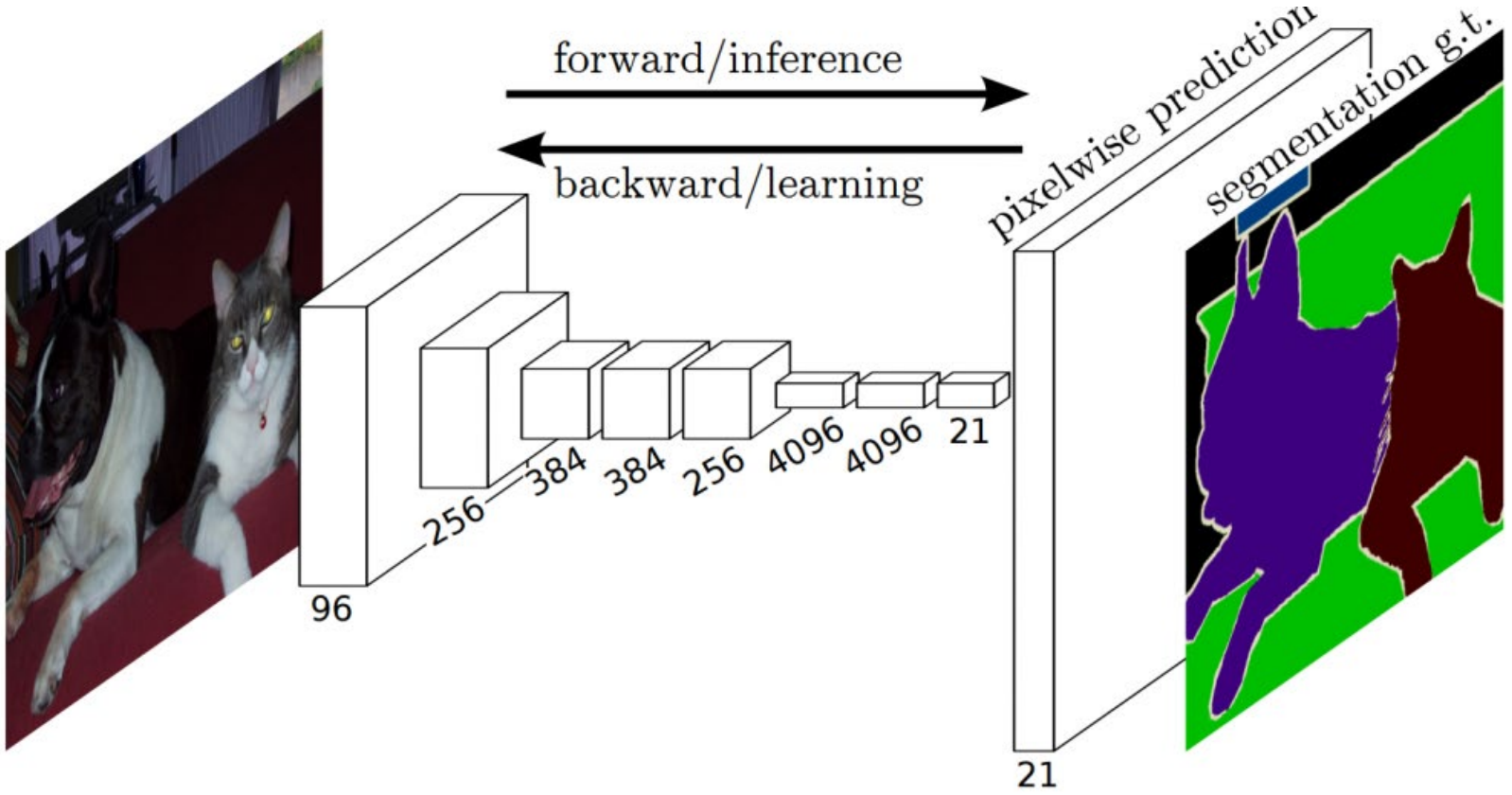
# Semantic Segmentation


圖片來源：https://medium.com/@keremturgutlu/semantic-segmentation-u-net-part-1-d8d6f6005066

https://ithelp.ithome.com.tw/articles/10204738

# Fully Convolutional Networks (FCN)

end-to-end 的 CNN 架構，所有的 layer 都是卷積層，無任何全連接層（fully connected layers），可適應任意尺寸的輸入，且減少參數的數量和計算時間




圖片來源：http://deeplearning.net/tutorial/fcn_2D_segm.html

# Instance Segmentation

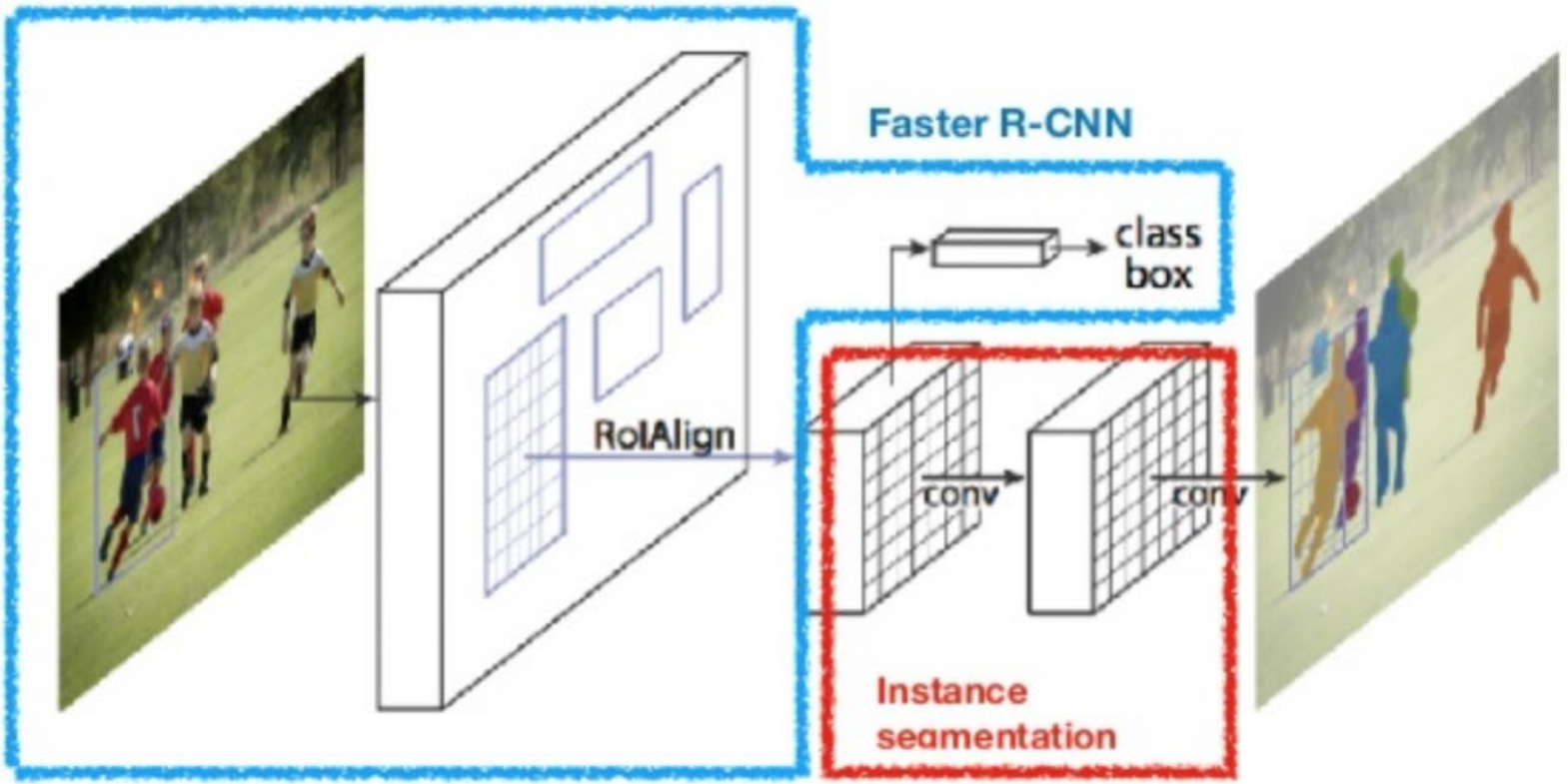
將不同類別的 instance 做切割。
Instance Segmentation 比分類任務更為複雜，例如景點有多個重疊的物體和不同的背景，不僅要對這些不同的物體進行分類，還要確定它們之間的邊界、差異和關係。


圖片來源：https://medium.com/@keremturgutlu/semantic-segmentation-u-net-part-1-d8d6f6005066

https://ithelp.ithome.com.tw/articles/10204738

# Mask R-CNN

將 Faster R-CNN 擴展到 pixel級的圖像分割，運用 RoIAlign（Region of Interests Align）來讓遮罩位置更準確




https://ithelp.ithome.com.tw/articles/10204738

# 全景分割(2018)
# Panoptic Segmentation

圖像中的每個圖元點都必須被分配給一個語義標籤和一個實例id

https://mp.weixin.qq.com/s?__biz=MzA3NDIyMjM1NA==&mid=2649033506&idx=1&sn=507d1e00cb348ebfc3e8cba6616891a4&chksm=8712b55fb0653c4912a4872eed4ce39d2c0a312ab143af0acd98e2cfa882004f716c8ad045f3&token=1045771457&lang=zh_CN#rd

# 全景分割資料庫

Cityscapes：https://www.cityscapes-dataset.com/

ADE20k：http://groups.csail.mit.edu/vision/datasets/ADE20K/

Mapillary Vistas：
https://blog.mapillary.com/product/2017/05/03/mapillary-vistas-dataset.html

## Basic classification: Classify images of clothing

https://www.tensorflow.org/tutorials/keras/classification

**Convolutional Neural Network (CNN)**

https://www.tensorflow.org/tutorials/images/cnn

training a simple Convolutional Neural Network (CNN) to classify CIFAR images.

## Image classification

how to classify cats or dogs from images.

It builds an image classifier using a tf.keras.Sequential model and load data using tf.keras.preprocessing.image.ImageDataGenerator

https://www.tensorflow.org/tutorials/images/classification

# Transfer Learning

**Transfer learning with TensorFlow Hub**

https://www.tensorflow.org/tutorials/images/transfer_learning_with_hub

**Transfer learning with a pretrained ConvNet**

Feature Extraction vs Fine-Tuning

https://www.tensorflow.org/tutorials/images/transfer_learning

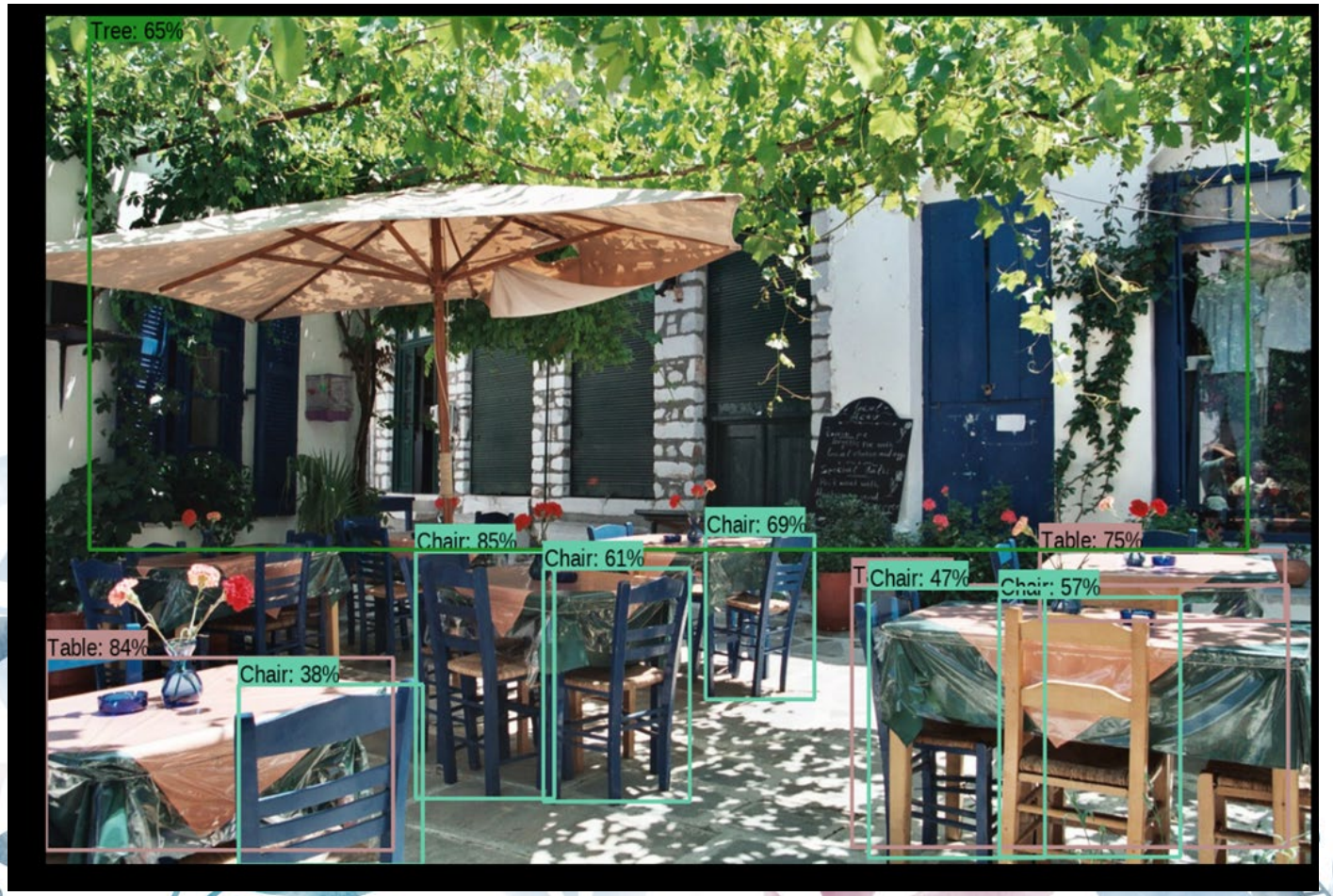
**Data augmentation**

Manual image manipulations and augmentation using tf.image

https://www.tensorflow.org/tutorials/images/data_augmentation

# Object Detection

https://github.com/tensorflow/hub/blob/master/examples/colab/object_detection.ipynb



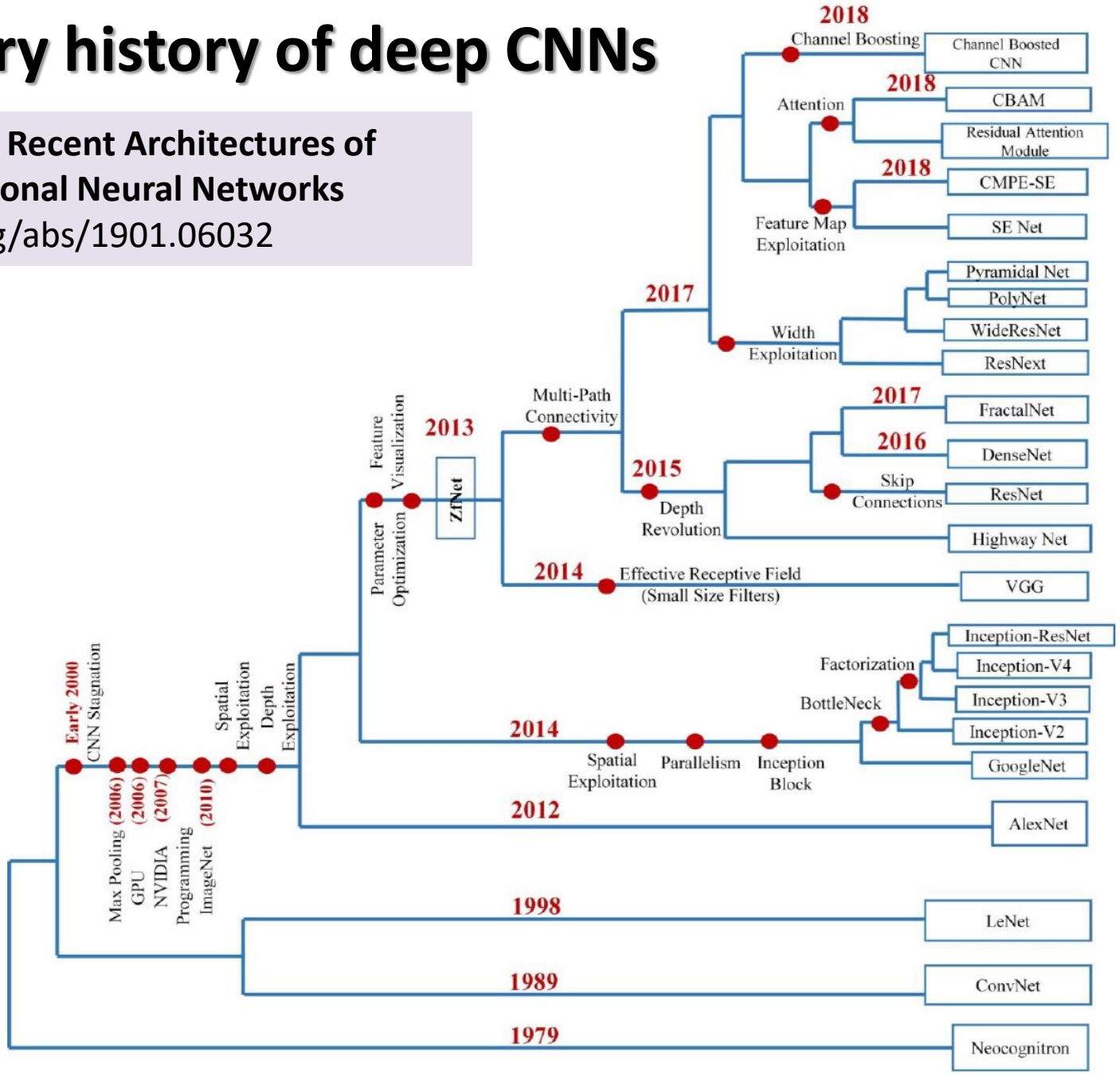
# Image segmentation using a modified U-Net

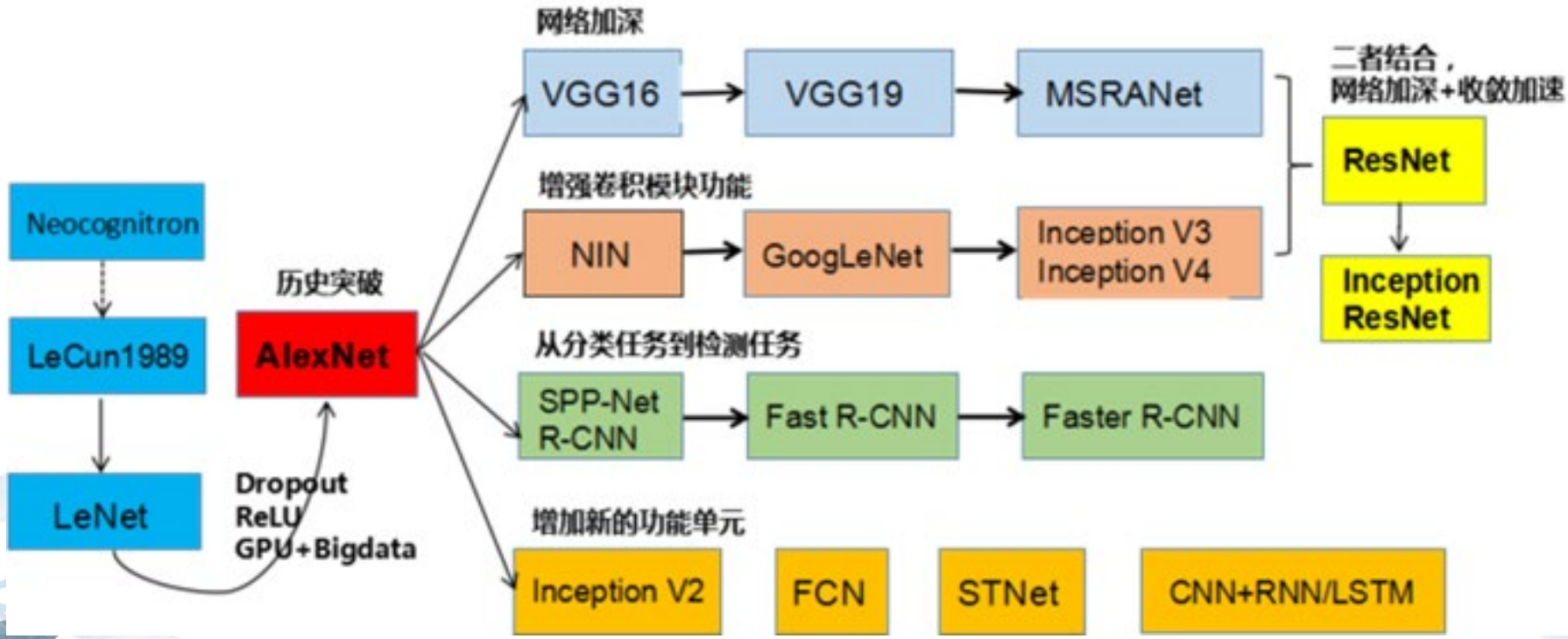
https://www.tensorflow.org/tutorials/images/segmentation

# CNN歷史進展

# Evolutionary history of deep CNNs

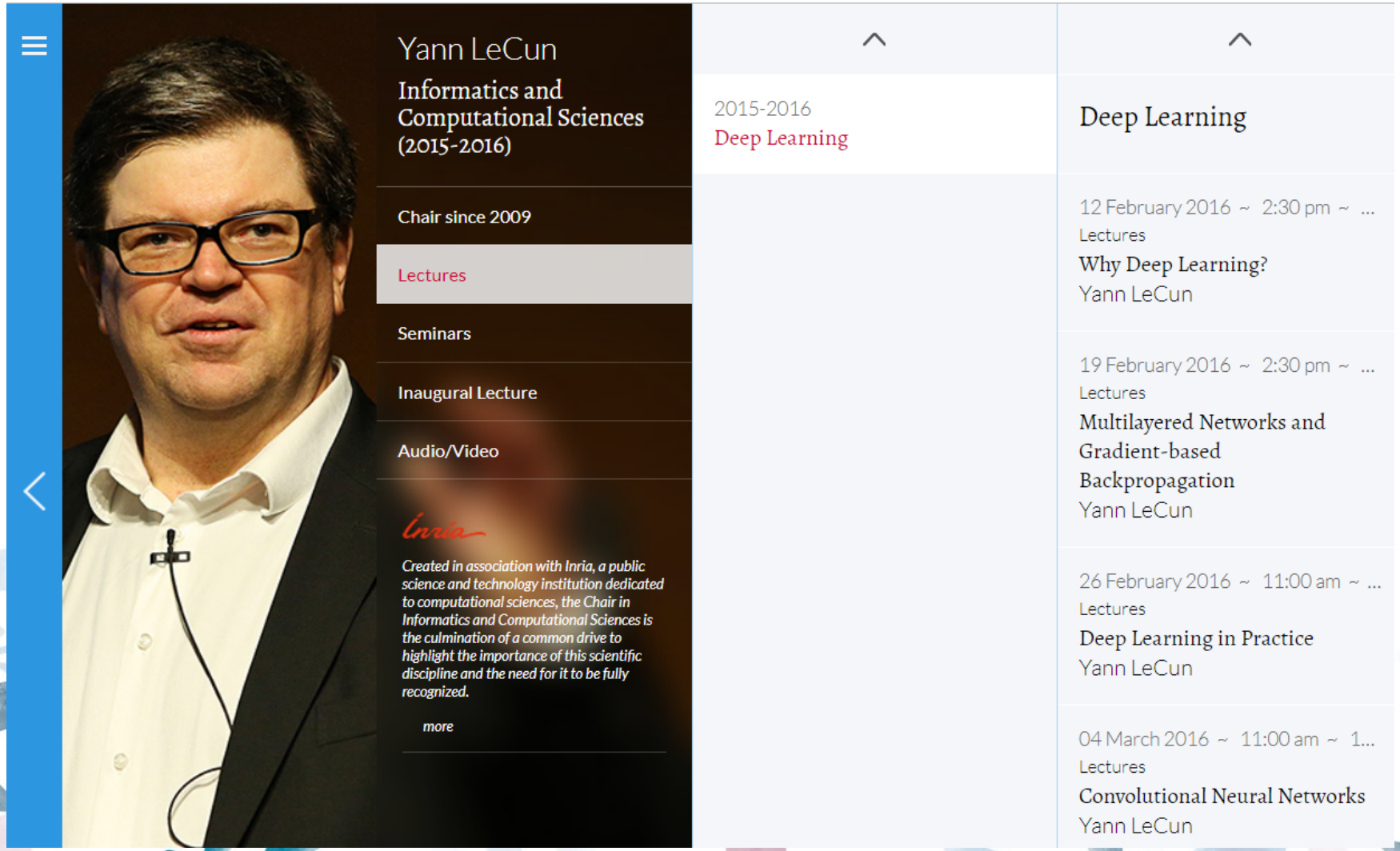
**A Survey of the Recent Architectures of Deep Convolutional Neural Networks**
https://arxiv.org/abs/1901.06032



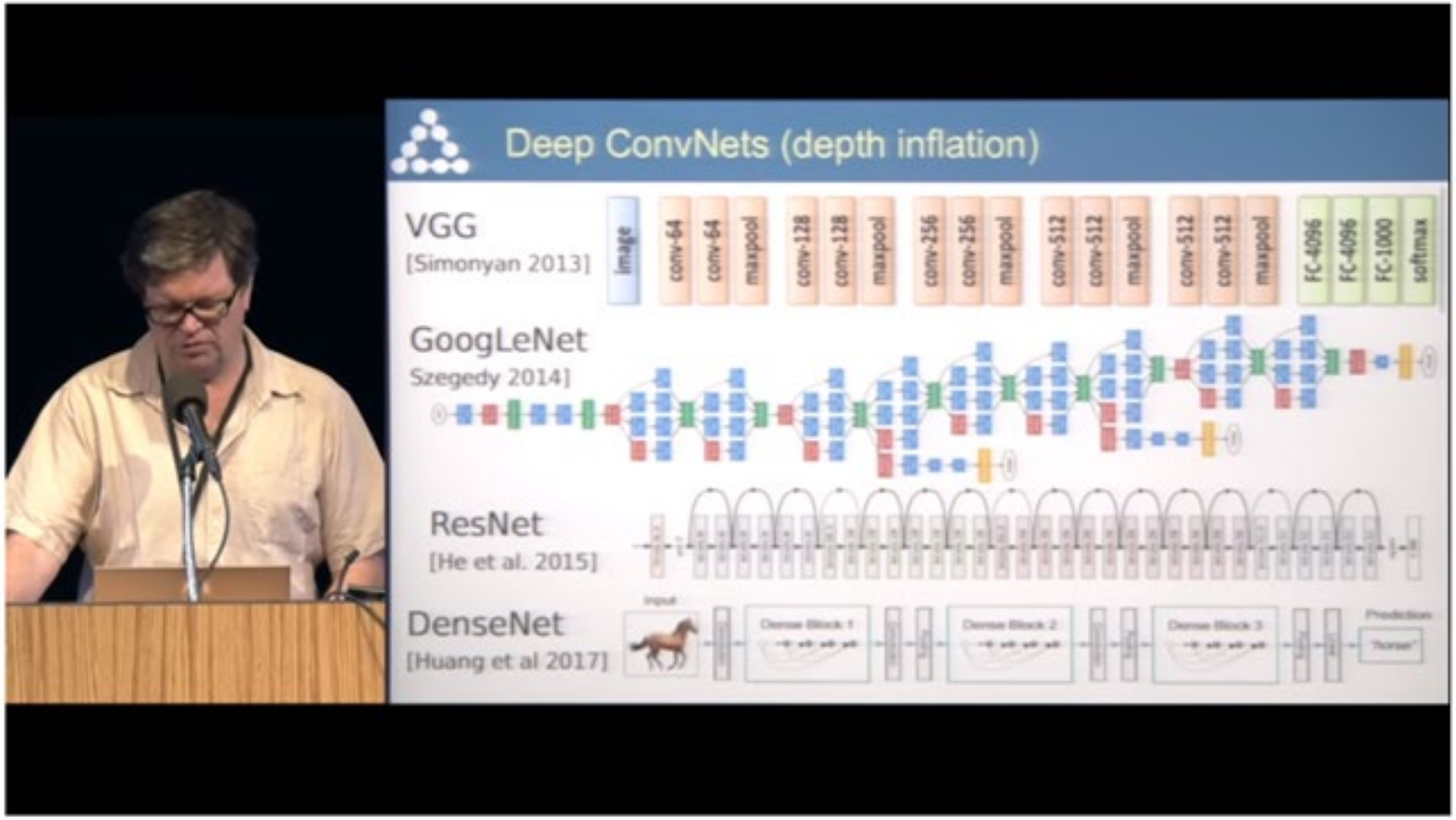



https://medium.com/@rgnuj122/lenet%E6%89%8B%E6%8A%8A%E6%89%8B%E5%AF%A6%E4%BD%9C%E6%95%99%E5%AD%B8-1c2f149c822f

https://www.college-de-france.fr/site/en-yann-lecun/course-2015-2016.htm



https://www.youtube.com/watch?v=cWzi38-vDbE

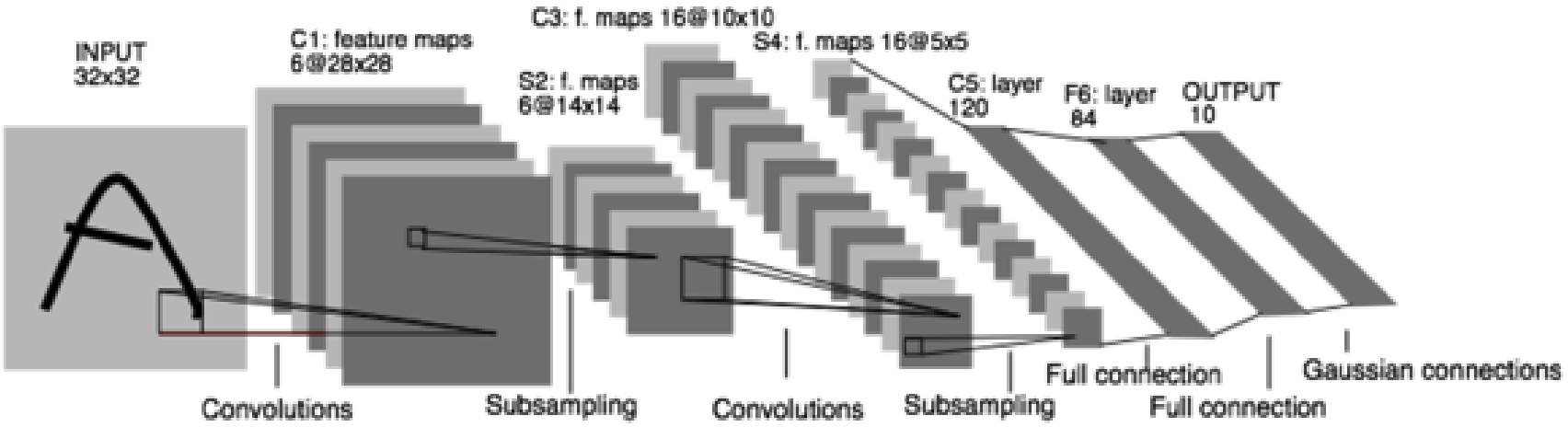



Yann LeCun - How does the brain learn so much so quickly? (CCN 2017)

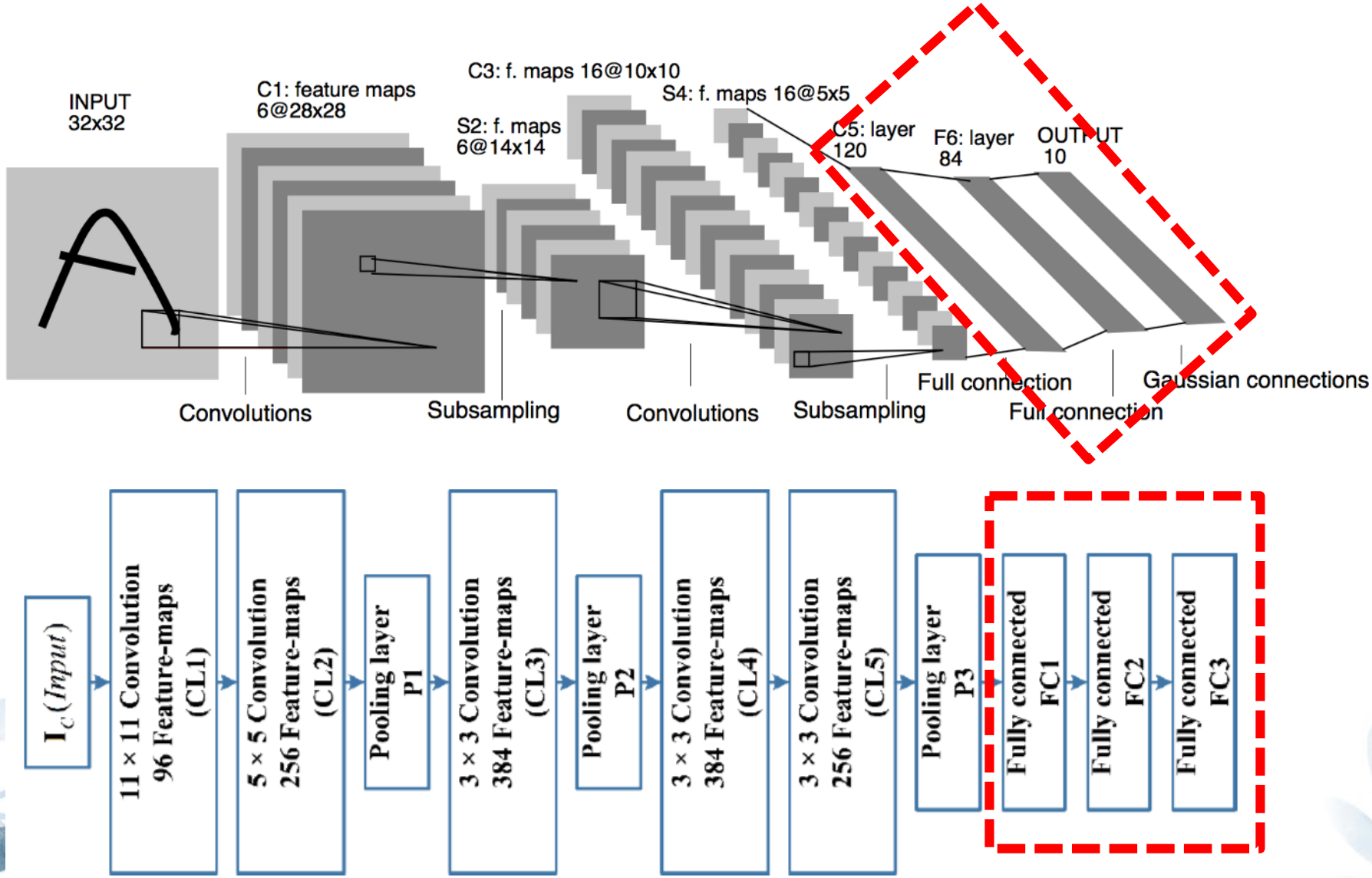
觀看次數：28,776次 喜歡 不喜歡 分享 儲存 ...

Presented at Cognitive Computational Neuroscience (CCN) 2017 (http://www.ccneuro.org) held September 6-8, 2017.

# Convolutional Networks: 1989




LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [ LeNet ]




**A Survey of the Recent Architectures of Deep Convolutional Neural Networks**
https://arxiv.org/abs/1901.06032

# ImageNet

**(since 2009)**

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was founded in 2010

How we teach computers to understand pictures | Fei Fei Li

https://www.youtube.com/watch?v=40riCqvRoMs

ImageNet is the modern object recognition benchmark dataset. It was introduced in 2009, and has led to amazing progress in object recognition since then.



How we teach computers to understand pictures | Fei Fei Li

# Godfather of Deep Learning

重要里程碑

1986 :the backpropagation algorithm for training multi-layer neural networks

https://www.cs.toronto.edu/~hinton/

Hinton, G. E., Osindero, S. and Teh, Y. (2006)
A fast learning algorithm for **deep** belief nets.
Neural Computation, 18, pp 1527-1554

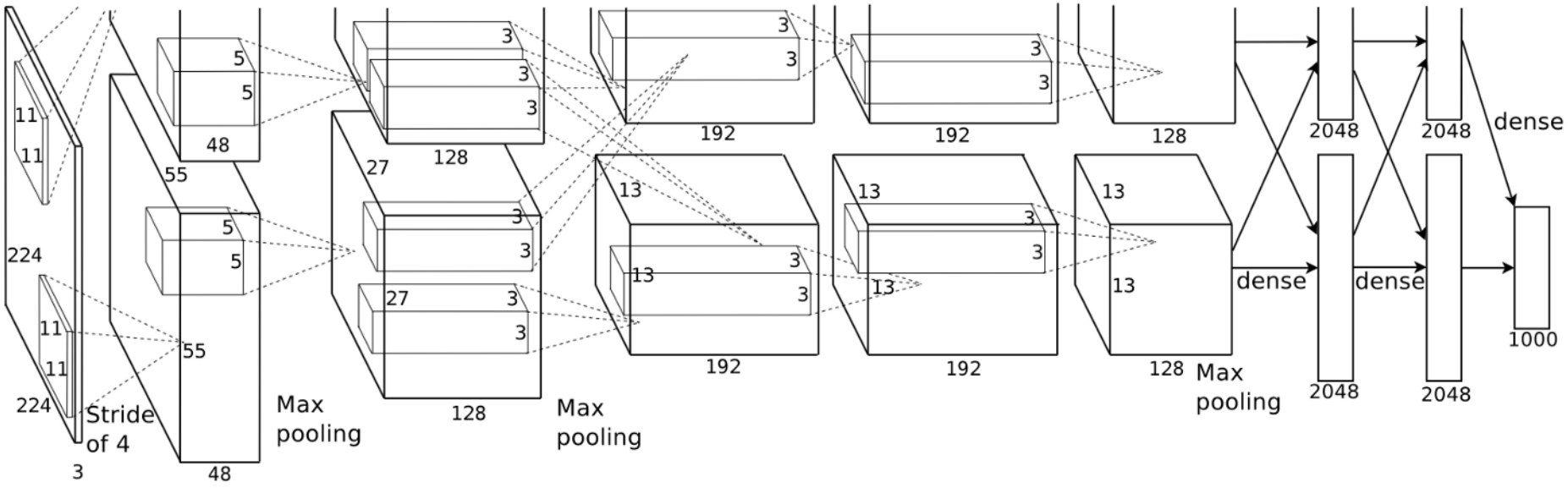
2012 AlexNet ( Alex Krizhevsky)啟動AI/NN電腦視覺熱潮

2018 Turing Prize (Yoshua Bengio and Yann LeCun )

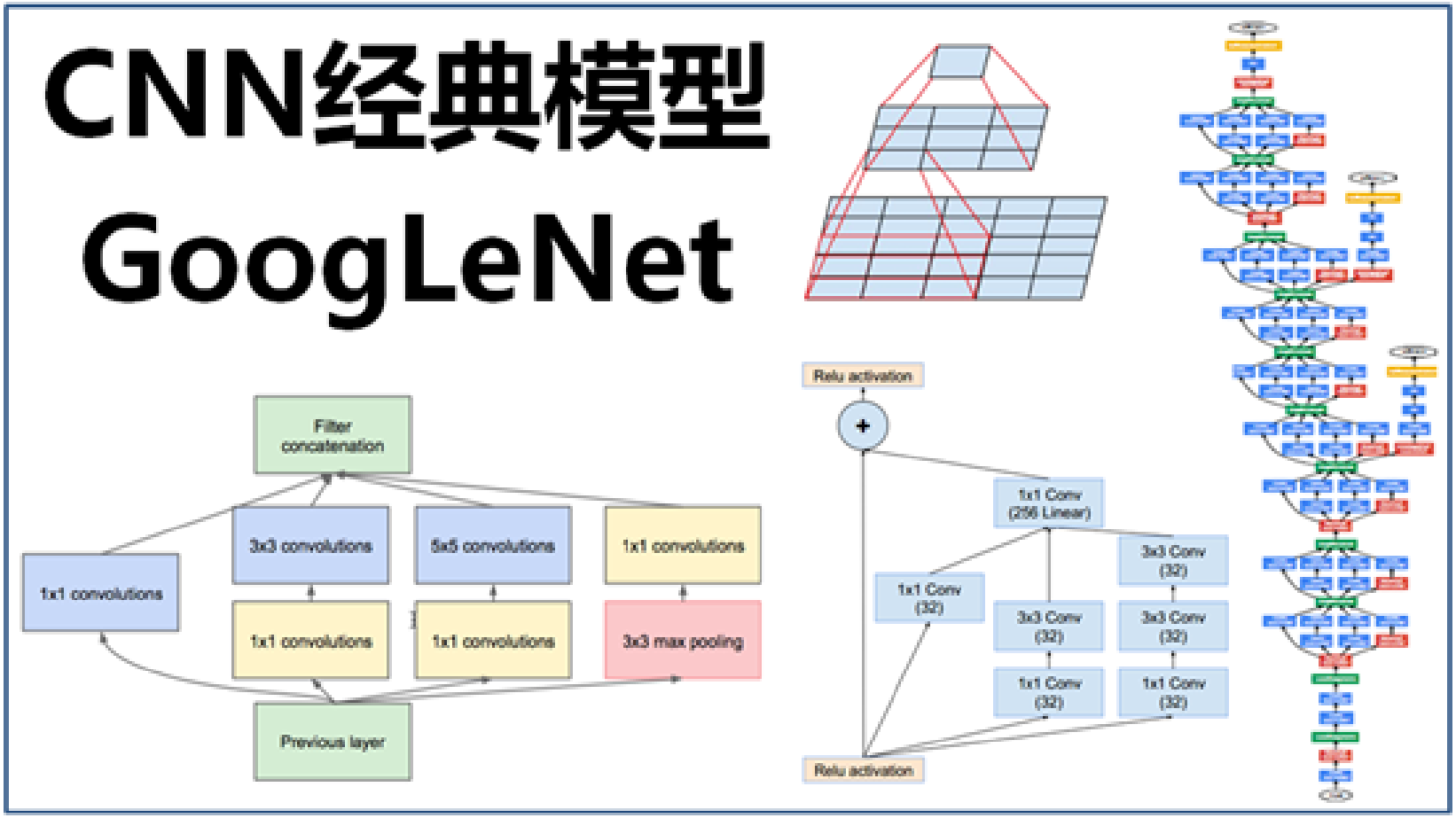
2012年在coursera開設課程的簡報
https://www.cs.toronto.edu/~hinton/coursera_slides.html

# AlexNet(2012)::大突破

- Hinton學生Alex Krizhevsky於2012年提出並拿下ILSVRC”12的冠軍讓CNN重返榮耀
- 其將top-5 error減少至15.3% ， outperform同年度第二名26.2%
- LeNet5的加強版
- 主要的新技術與應用::將ReLU, Dropout, LRN加到model中
- 用GPU來加快training效率
- data augmentation增加訓練資料集



# GoogLeNet(2014)




https://www.itread01.com/content/1544969366.html

# GoogLeNet architecture

In 2014, ILSVRC, Google published its own network known as GoogLeNet. Its performance is a little better than VGGNet; GoogLeNet's performance is 6.7% compared to VGGNet's performance of 7.3%.

The main attractive feature of GoogLeNet is that it runs very fast due to the introduction of a new concept called **inception module**, thus reducing the number of parameters to only 5 million; that's 12 times less than AlexNet.

It has lower memory use and lower power use too.

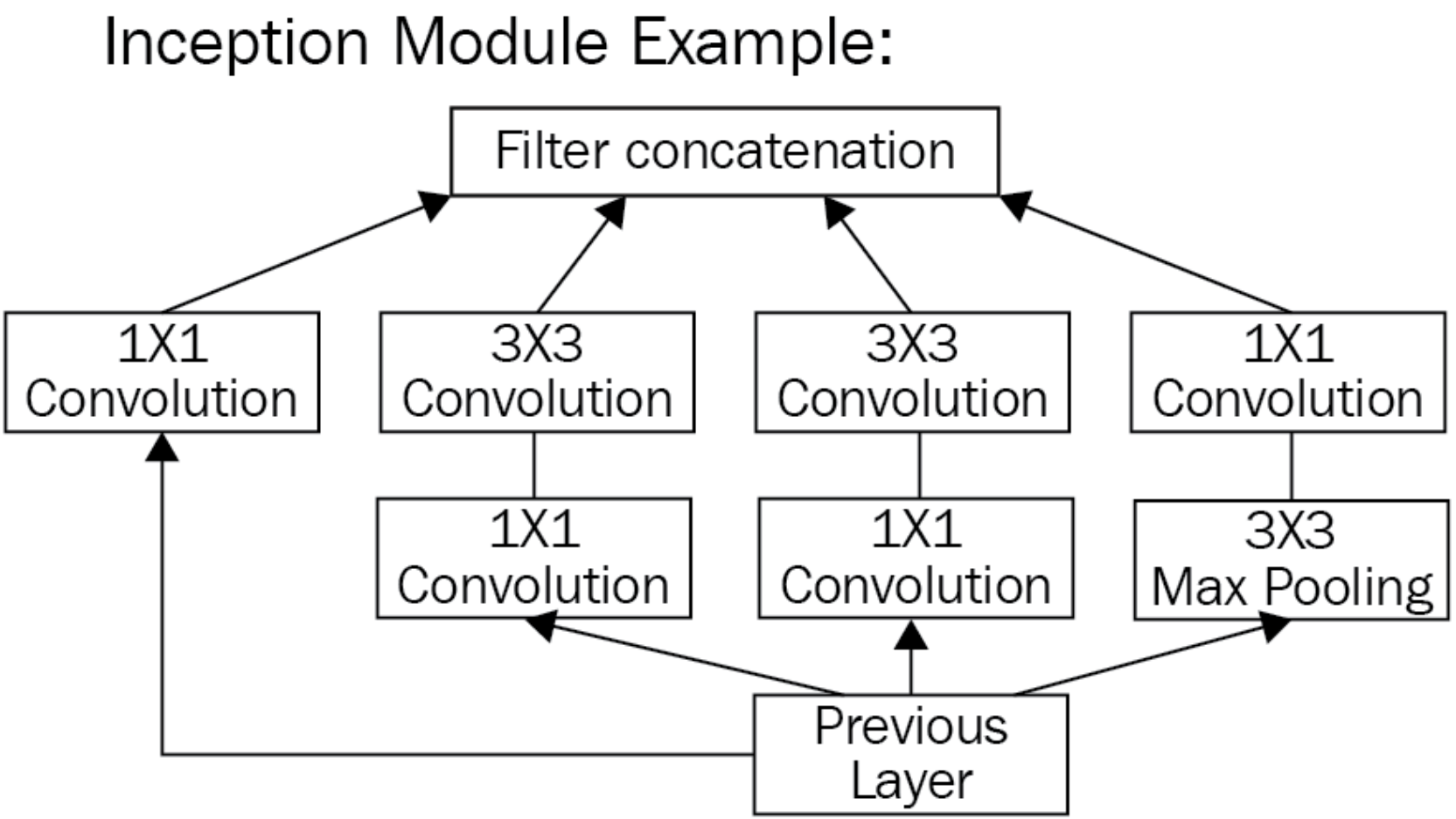
It has 22 layers, so it is a very deep network. Adding more layers increases the number of parameters and it is likely that the network overfits. There will be more computation, because a linear increase in filters results in a quadratic increase in computation. So, the designers use the inception module and GAP. The fully connected layer at the end of the network is replaced with a GAP layer because fully connected layers are generally prone to overfitting. GAP has no parameters to learn or optimize.

Inception V1

Inception V2

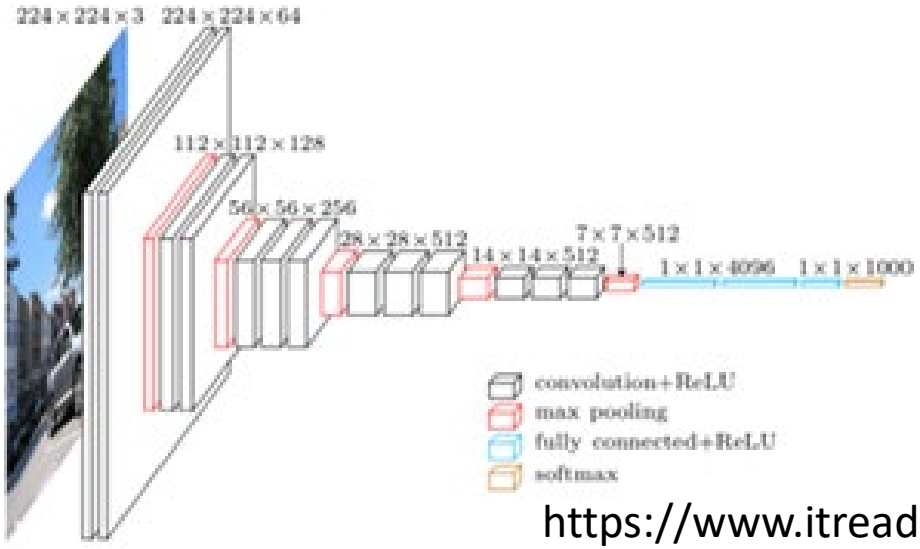
Inception V3

Inception V4



# CNN经典模型

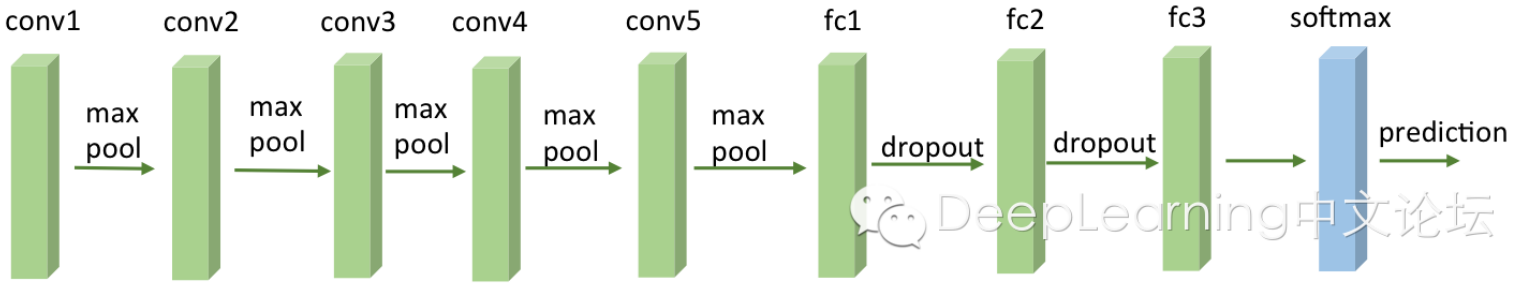
# VGGNet




| 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 | | | | | |

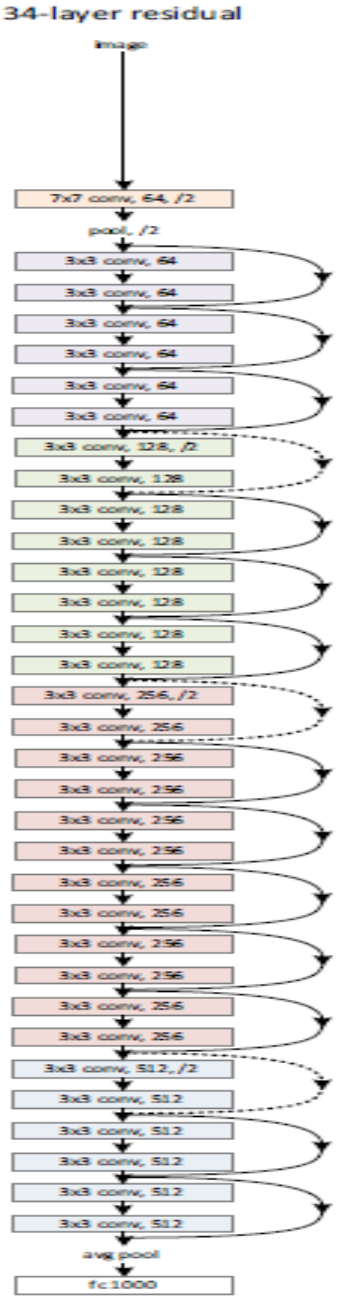
https://www.itread01.com/content/1541992744.html

each conv includes 3 convolutional layers



# ResNet(2015)
# Deep Residual Lea

最深的model
採用的152層



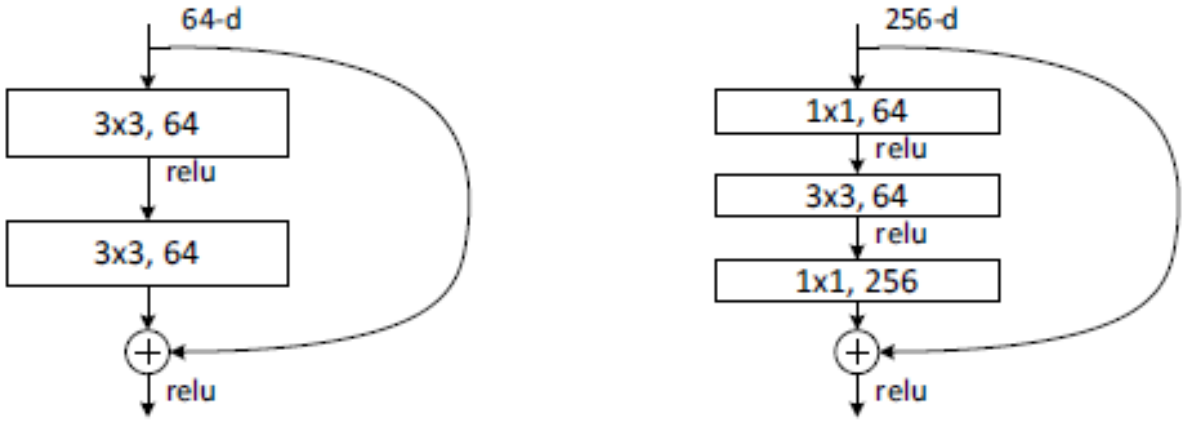
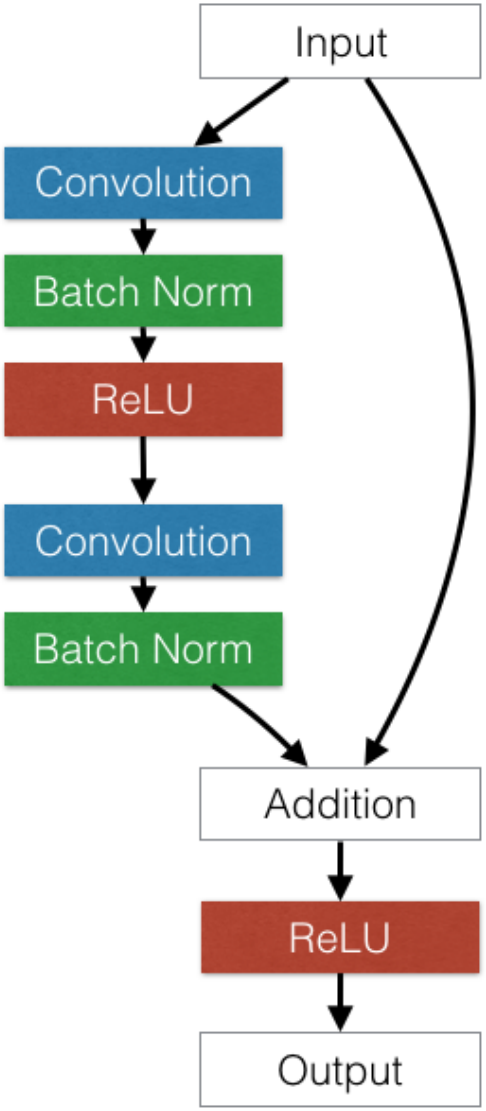
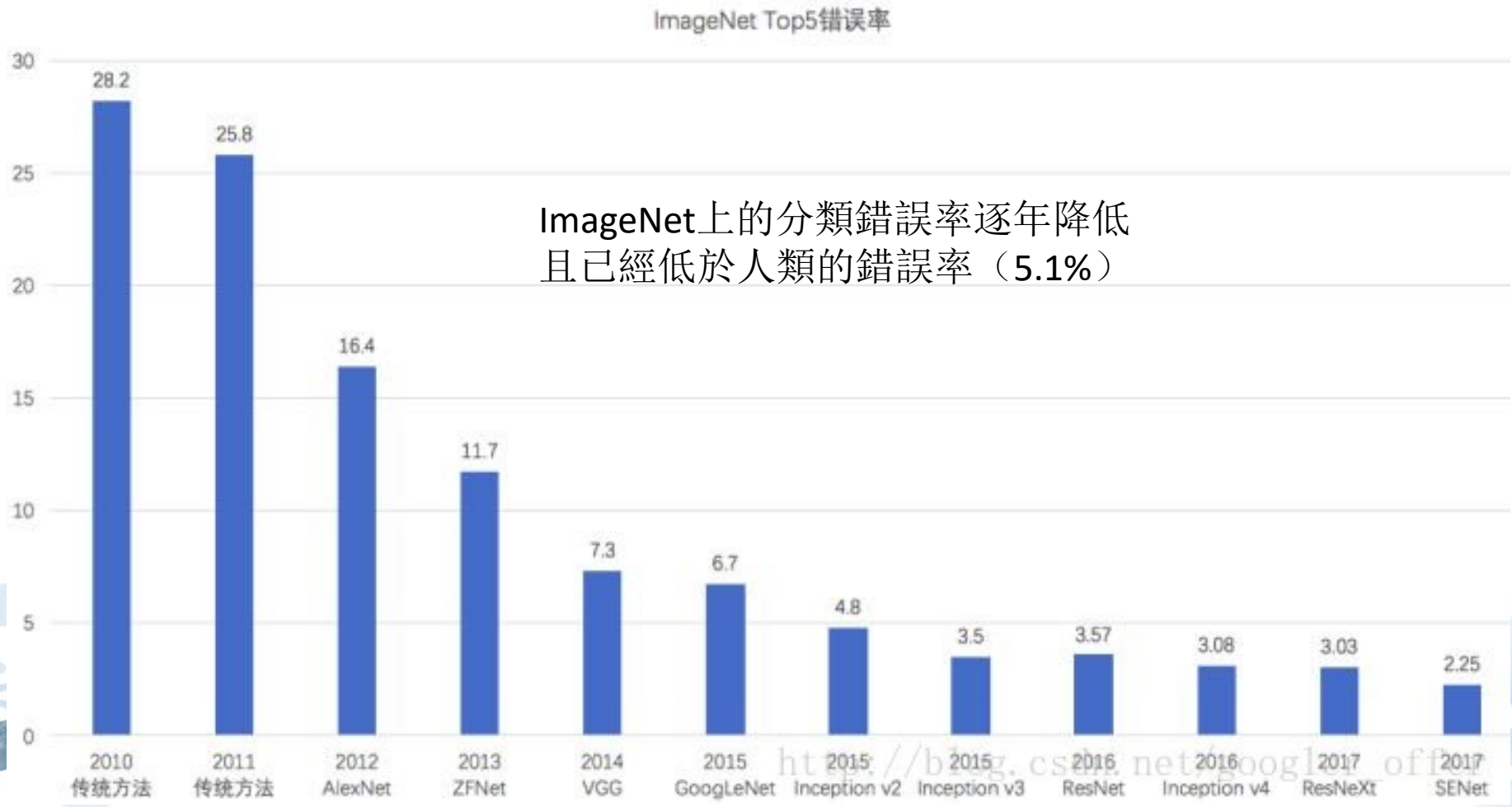



Figure 5. A deeper residual function $\mathcal{F}$ for ImageNet. Left: a building block (on 56×56 feature maps) as in Fig. 3 for ResNet-34. Right: a "bottleneck" building block for ResNet-50/101/152.

https://www.itread01.com/content/1541992744.html




In the 2015 ImageNet ILSVRC competition, the winner was ResNet from Microsoft, with an error rate of 3.57%. ResNet is a kind of VGG in the sense that the same structure is repeated again and again to make the network deeper. Unlike VGGNet, it has different depth variations, such as 34, 50, 101, and 152 layers. It has a whopping 152 layers compared to AlexNet 8, VGGNet's 19 layers, and GoogLeNet's 22 layers. The ResNet architecture is a stack of residual blocks. The main idea is to skip layers by adding connections to the neural network. Every residual block has 3 x 3 convolution layers. After the last conv layer, a GAP layer is added. There is only one fully connected layer to classify 1,000 classes. It has different depth varieties, such as 34, 50, 101, or 152 layers for the ImageNet dataset. For a deeper network, say more than 50 layers, it uses the bottleneck features concept to improve efficiency. No dropout is used in this network.




ImageNet上的分類錯誤率逐年降低
且已經低於人類的錯誤率（5.1%）

# SENet::Squeeze-and-Excitation Networks(2017-2019)

https://arxiv.org/abs/1709.01507 **Caffe實作** https://github.com/hujie-frank/SENet

The central building block of convolutional neural networks (CNNs) is the **convolution operator**, which enables networks to construct informative features by fusing both spatial and channel-wise information within local receptive fields at each layer.

A broad range of prior research has investigated the spatial component of this relationship, seeking to strengthen the representational power of a CNN by enhancing the quality of spatial encodings throughout its feature hierarchy.

In this work, we focus instead on the channel relationship and propose a novel architectural unit, which we term **the "Squeeze-and-Excitation" (SE) block**, that adaptively recalibrates channel-wise feature responses by explicitly modelling interdependencies between channels. We show that these blocks can be stacked together to form SENet architectures that generalise extremely effectively across different datasets. We further demonstrate that SE blocks bring significant improvements in performance for existing state-of-the-art CNNs at slight additional computational cost. Squeeze-and-Excitation Networks formed the foundation of our ILSVRC 2017 classification submission which won first place and reduced the top-5 error to 2.251%, surpassing the winning entry of 2016 by a relative improvement of ~25%. Models and code are available at this https URL.

**Tensorflow 實作** https://github.com/taki0112/SENet-Tensorflow

**PyTorch 實作** https://github.com/moskomule/senet.pytorch

**Keras 實作**

https://github.com/titu1994/keras-squeeze-excite-network

https://github.com/yoheikikuta/senet-keras

**說明文件**

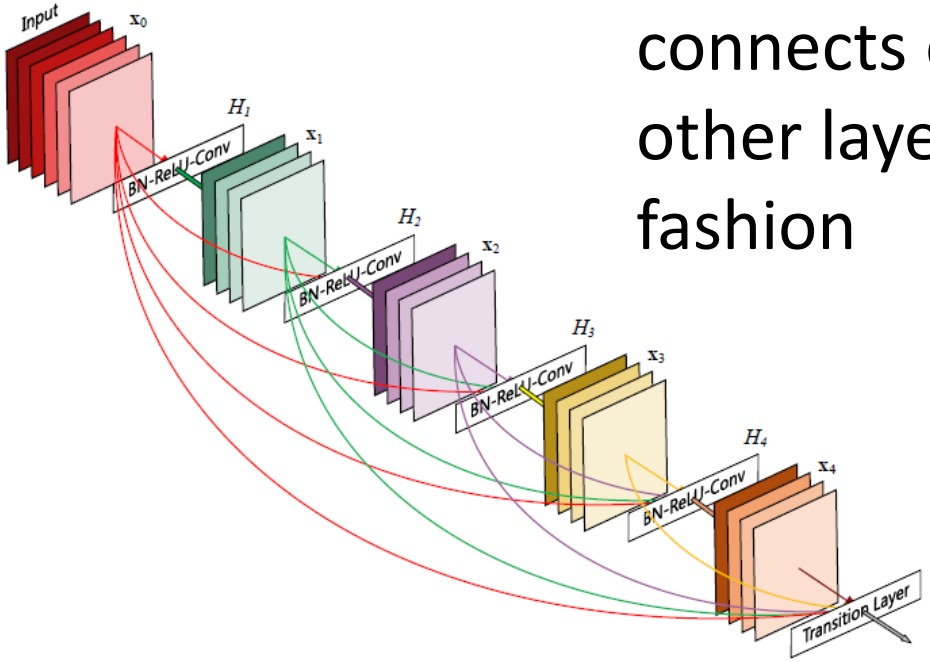
https://blog.csdn.net/u014380165/article/details/78006626

keras瞎搞系列-從LeNet到SENet——卷積神經網路回顧
https://blog.csdn.net/googler_offer/article/details/79478893

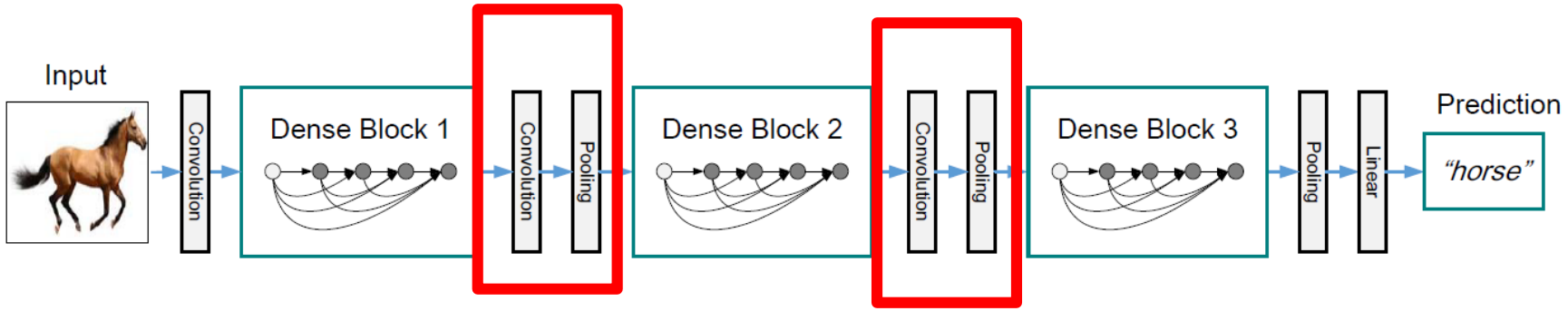
# DenseNet(2018)

Densely Connected Convolutional Networks
Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger
https://arxiv.org/abs/1608.06993

- 相較於ResNet, DenseNet 可用更少的訓練參數取得更佳的準確率
- **CVPR 201 7 最佳論文**
- Keras 提供數個DenseNet 版本供我們使用
  ： DenseNet 121 、DenseNet 169 、DenseNet201




connects each layer to every other layer in a feed-forward fashion




# DenseNet architectures for ImageNet

| Layers | Output Size | DenseNet-121 | DenseNet-169 | DenseNet-201 | DenseNet-264 |
|---|---|---|---|---|---|
| Convolution | 112 × 112 | 7 × 7 conv, stride 2 | | | |
| Pooling | 56 × 56 | 3 × 3 max pool, stride 2 | | | |
| Dense Block (1) | 56 × 56 | [1 × 1 conv, 3 × 3 conv] × 6 | [1 × 1 conv, 3 × 3 conv] × 6 | [1 × 1 conv, 3 × 3 conv] × 6 | [1 × 1 conv, 3 × 3 conv] × 6 |
| Transition Layer (1) | 56 × 56 | 1 × 1 conv | | | |
| | 28 × 28 | 2 × 2 average pool, stride 2 | | | |
| Dense Block (2) | 28 × 28 | [1 × 1 conv, 3 × 3 conv] × 12 | [1 × 1 conv, 3 × 3 conv] × 12 | [1 × 1 conv, 3 × 3 conv] × 12 | [1 × 1 conv, 3 × 3 conv] × 12 |
| Transition Layer (2) | 28 × 28 | 1 × 1 conv | | | |
| | 14 × 14 | 2 × 2 average pool, stride 2 | | | |
| Dense Block (3) | 14 × 14 | [1 × 1 conv, 3 × 3 conv] × 24 | [1 × 1 conv, 3 × 3 conv] × 32 | [1 × 1 conv, 3 × 3 conv] × 48 | [1 × 1 conv, 3 × 3 conv] × 64 |
| Transition Layer (3) | 14 × 14 | 1 × 1 conv | | | |
| | 7 × 7 | 2 × 2 average pool, stride 2 | | | |
| Dense Block (4) | 7 × 7 | [1 × 1 conv, 3 × 3 conv] × 16 | [1 × 1 conv, 3 × 3 conv] × 32 | [1 × 1 conv, 3 × 3 conv] × 32 | [1 × 1 conv, 3 × 3 conv] × 48 |
| Classification Layer | 1 × 1 | 7 × 7 global average pool | | | |
| | | 1000D fully-connected, softmax | | | |

過渡層

→ BN-ReLU-Conv

全域平均池化

each "conv" layer shown in the table corresponds the sequence BN-ReLU-Conv

https://www.tensorflow.org/api_docs/python/tf/keras/applications

**densenet** module: DenseNet models for Keras.

**imagenet_utils** module: Utilities for ImageNet data preprocessing & prediction decoding.

inception_resnet_v2 module: Inception-ResNet V2 model for Keras.
inception_v3 module: Inception V3 model for Keras.

mobilenet module: MobileNet v1 models for Keras.
mobilenet_v2 module: MobileNet v2 models for Keras.

nasnet module: NASNet-A models for Keras.

resnet module: ResNet models for Keras.
resnet50 module: Public API for tf.keras.applications.resnet50 namespace.
resnet_v2 module: ResNet v2 models for Keras.

vgg16 module: VGG16 model for Keras.
vgg19 module: VGG19 model for Keras.

xception module: Xception V1 model for Keras.

```
from tensorflow.keras.applications import Xception
xception = Xception()
xception.summary()
```

https://arxiv.org/abs/1901.06032



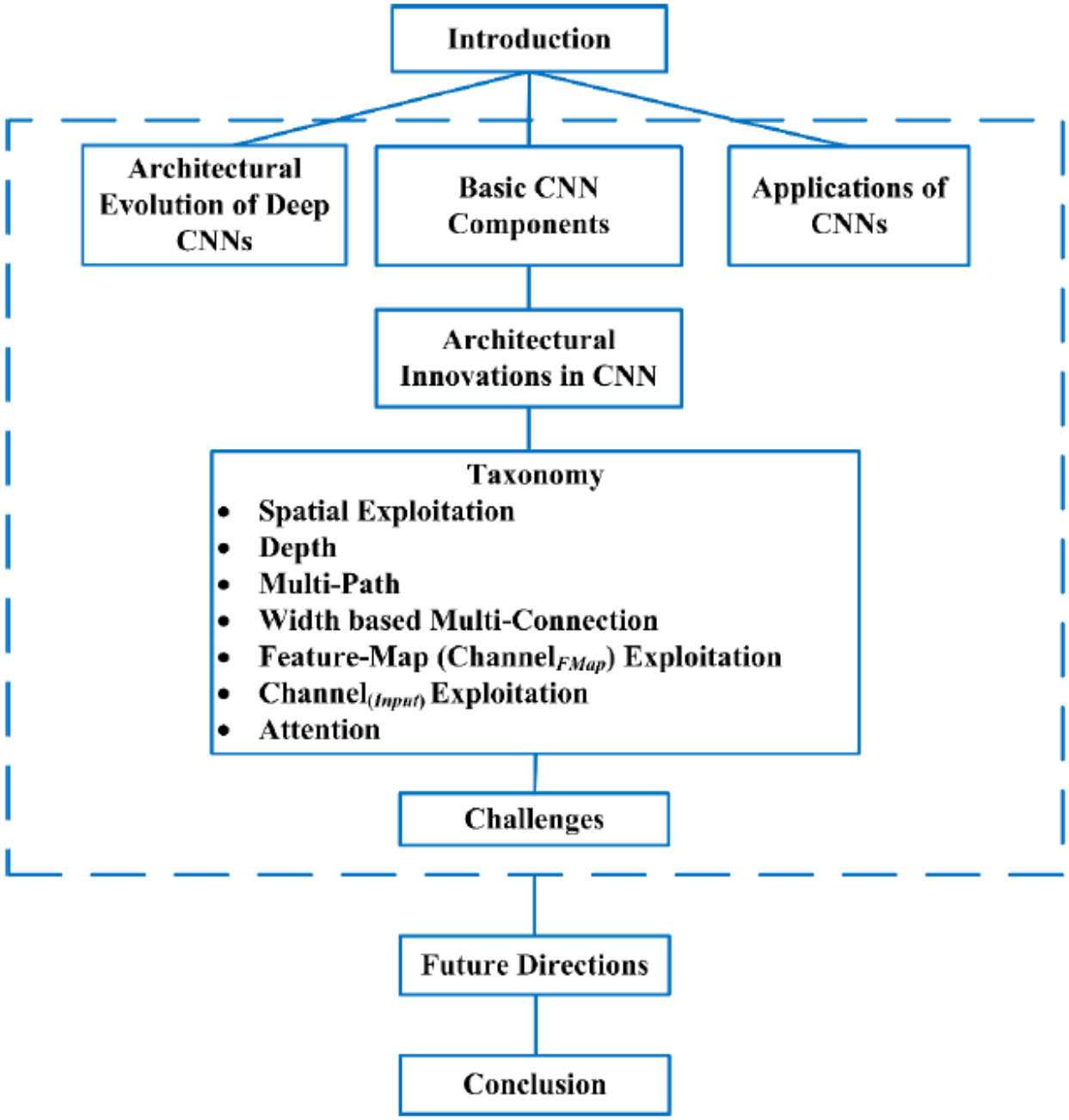
**Computer Science > Computer Vision and Pattern Recognition**

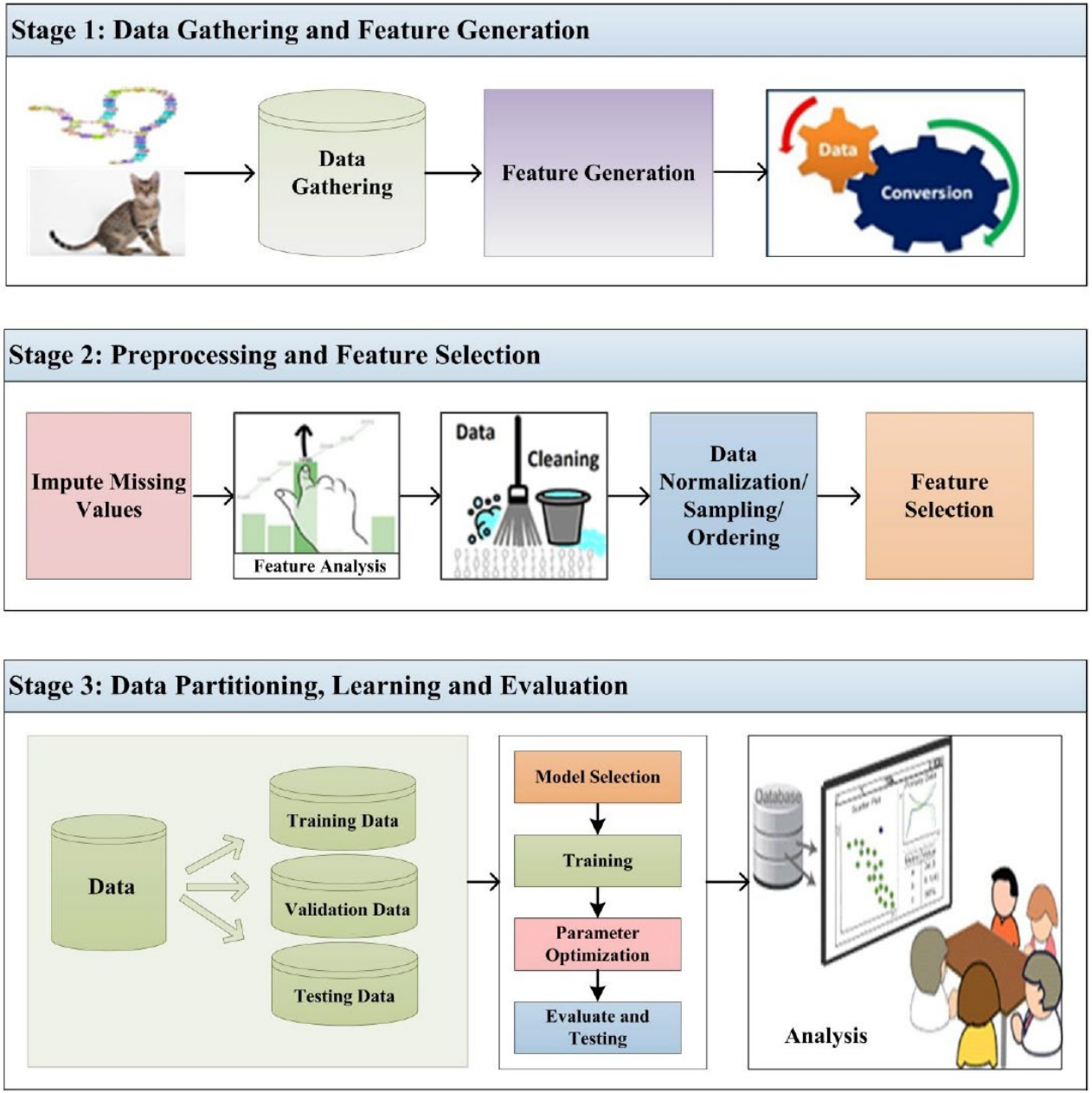
*[Submitted on 17 Jan 2019 (v1), last revised 10 May 2020 (this version, v7)]*

# A Survey of the Recent Architectures of Deep Convolutional Neural Networks

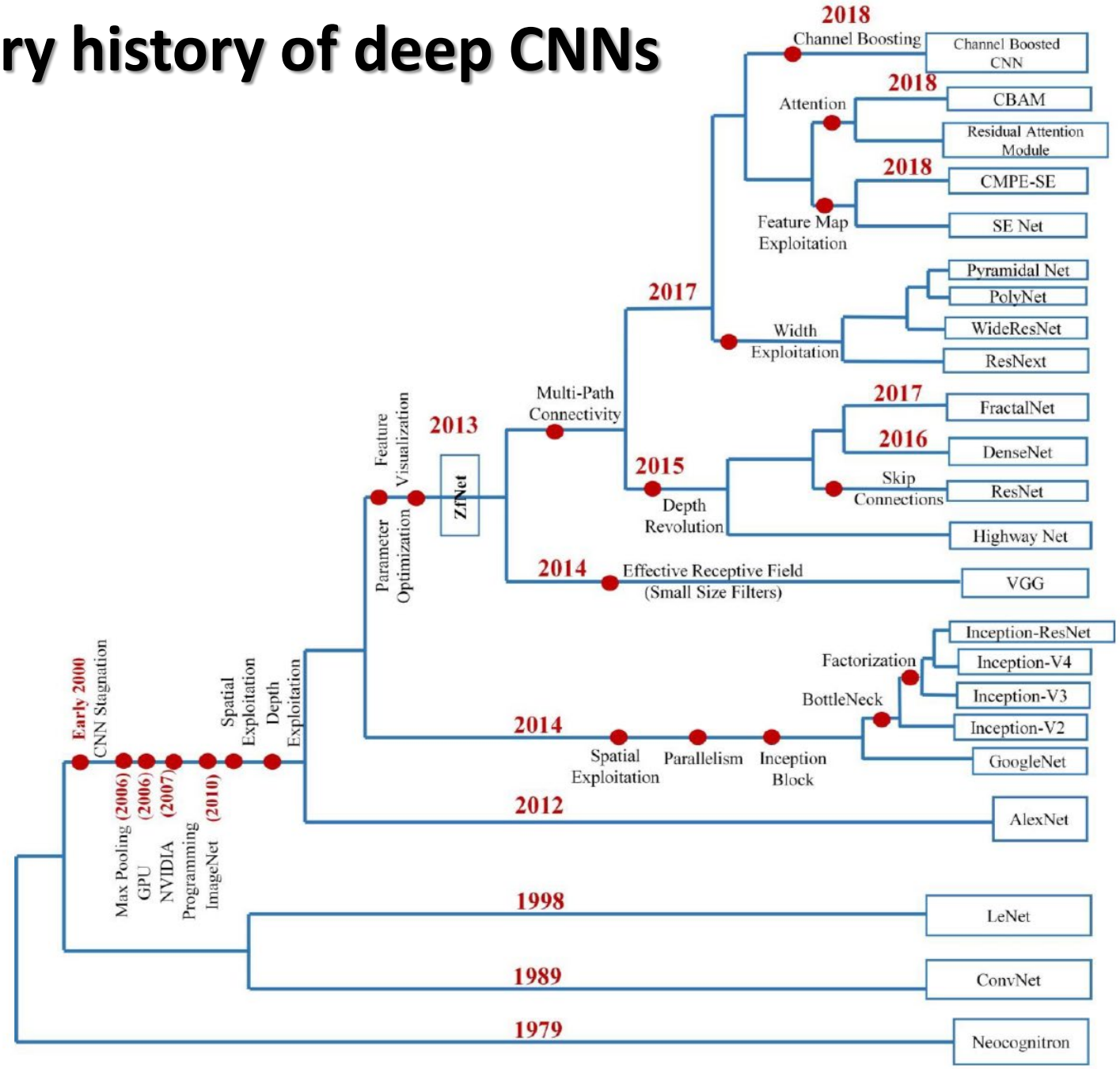
Asifullah Khan, Anabia Sohail, Umme Zahoora, Aqsa Saeed Qureshi

Deep Convolutional Neural Network (CNN) is a special type of Neural Networks, which has shown exemplary performance on several competitions related to Computer Vision and Image Processing. Some of the exciting application areas of CNN include Image Classification and Segmentation, Object Detection, Video Processing, Natural Language Processing, and Speech Recognition. The powerful learning ability of deep CNN is primarily due to the use of multiple feature extraction stages that can automatically learn representations from the data. The availability of a large amount of data and improvement in the hardware technology has accelerated the research in CNNs, and recently interesting deep CNN architectures have been reported. Several inspiring ideas to bring advancements in CNNs have been explored, such as the use of different activation and loss functions, parameter optimization, regularization, and architectural innovations. However, the significant improvement in the representational capacity of the deep CNN is achieved through architectural innovations. Notably, the ideas of exploiting spatial and channel information, depth and width of architecture, and multi-path information processing have gained substantial attention. Similarly, the idea of using a block of layers as a structural unit is also gaining popularity. This survey thus focuses on the intrinsic taxonomy present in the recently reported deep CNN architectures and, consequently, classifies the recent innovations in CNN architectures into seven different categories. These seven categories are based on spatial exploitation, depth, multi-path, width, feature-map exploitation, channel boosting, and attention. Additionally, the elementary understanding of CNN components, current challenges, and applications of CNN are also provided.

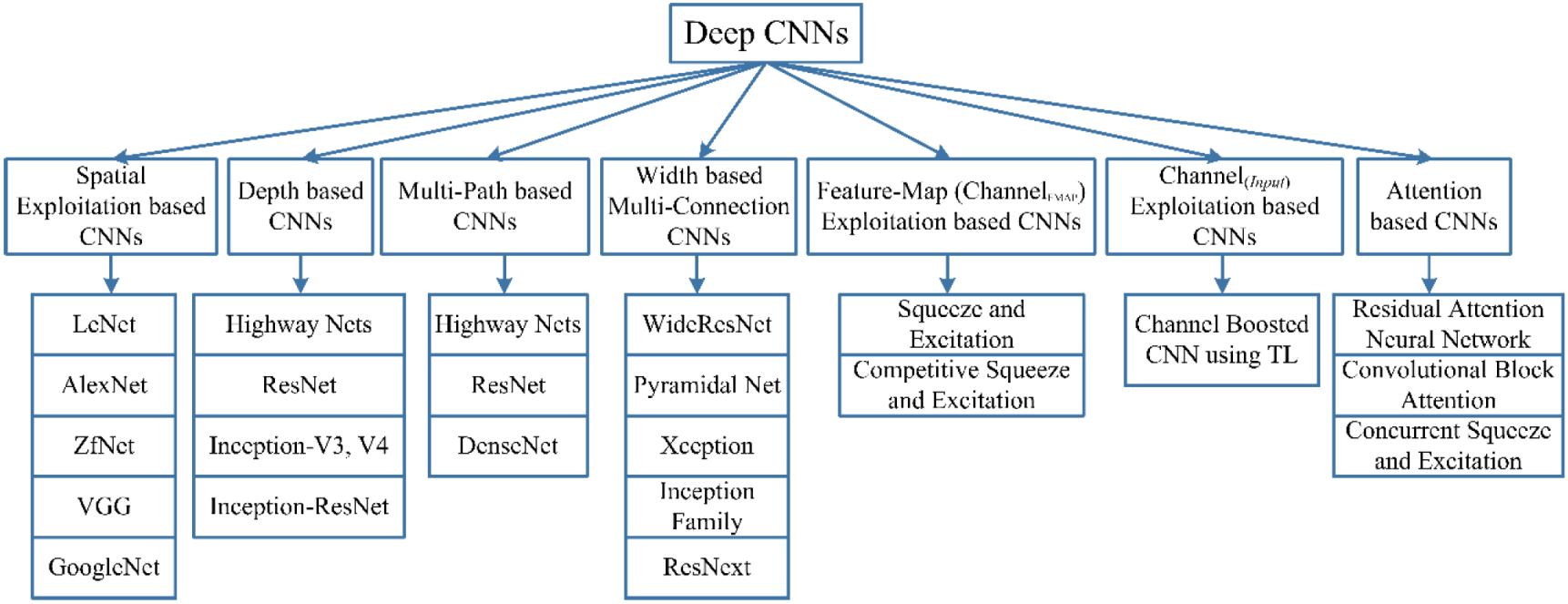

Introduction
Architectural Evolution of Deep CNNs
Basic CNN Components
Applications of CNNs
Architectural Innovations in CNN
Taxonomy
Spatial Exploitation
Depth
Multi-Path
Width based Multi-Connection
Feature-Map (Channel$_{FMap}$) Exploitation
Channel$_{(Input)}$ Exploitation
Attention
Challenges
Future Directions
Conclusion


Stage 1: Data Gathering and Feature Generation
Data Gathering
Feature Generation
Data
Conversion
Stage 2: Preprocessing and Feature Selection
Impute Missing Values
Feature Analysis
Data
Cleaning
Data Normalization/ Sampling/ Ordering
Feature Selection
Stage 3: Data Partitioning, Learning and Evaluation
Data
Training Data
Validation Data
Testing Data
Model Selection
Training
Parameter Optimization
Evaluate and Testing
Database
Analysis

# Evolutionary history of deep CNNs



# Taxonomy of deep CNN architectures



# 作業
# Cifar10 圖像分類

# Cifar10

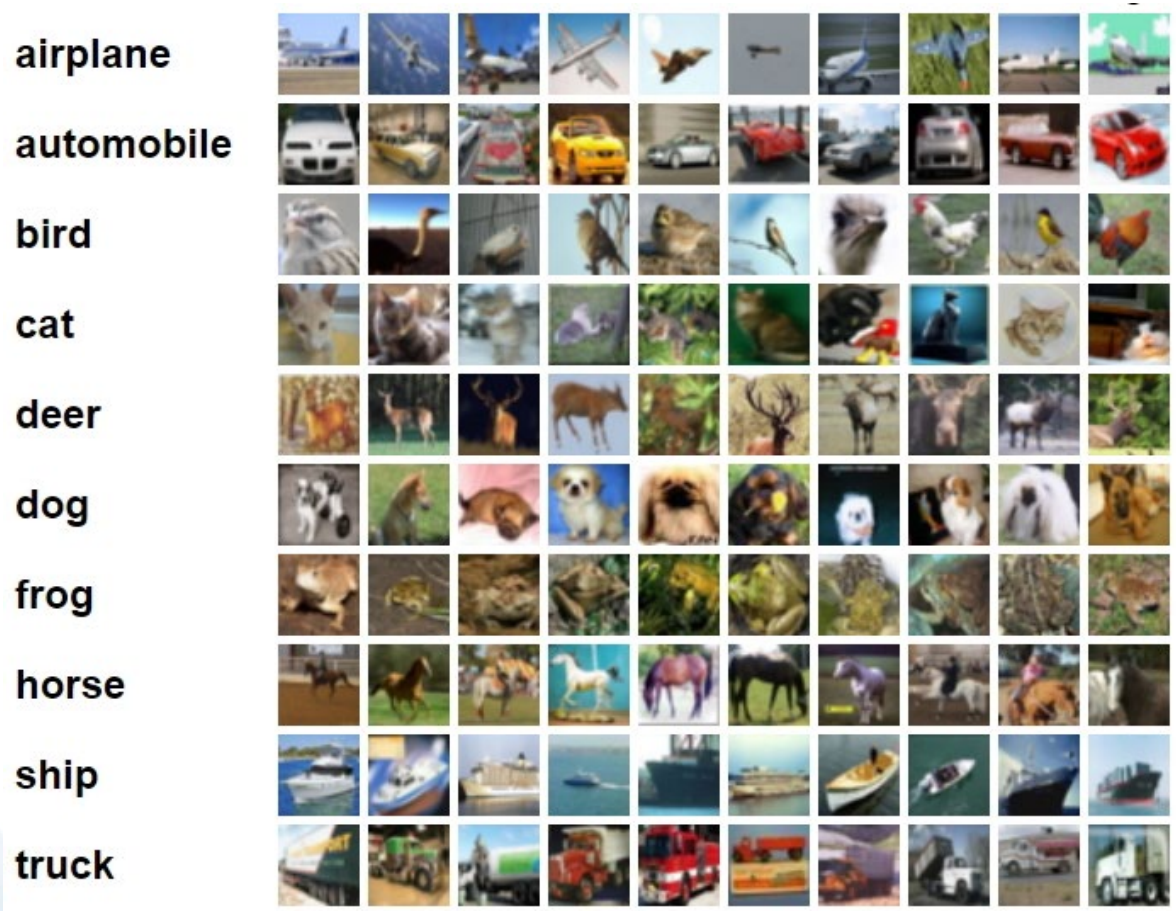
Canadian Institute For Advanced Research
加拿大高等研究院

https://en.wikipedia.org/wiki/CIFAR-10

Cifar-10 的所有圖片被分為 10 個類別 (以 0~9 數字作為 Label 之編碼) :

0 : airplain (飛機)
1 : automobile (汽車)
2 : bird (鳥)
3 : cat (貓)
4 : deer (鹿)
5 : dog (狗)
6 : frog (青蛙)
7 : horse (馬)
8 : ship (船)
9 : truck (卡車)

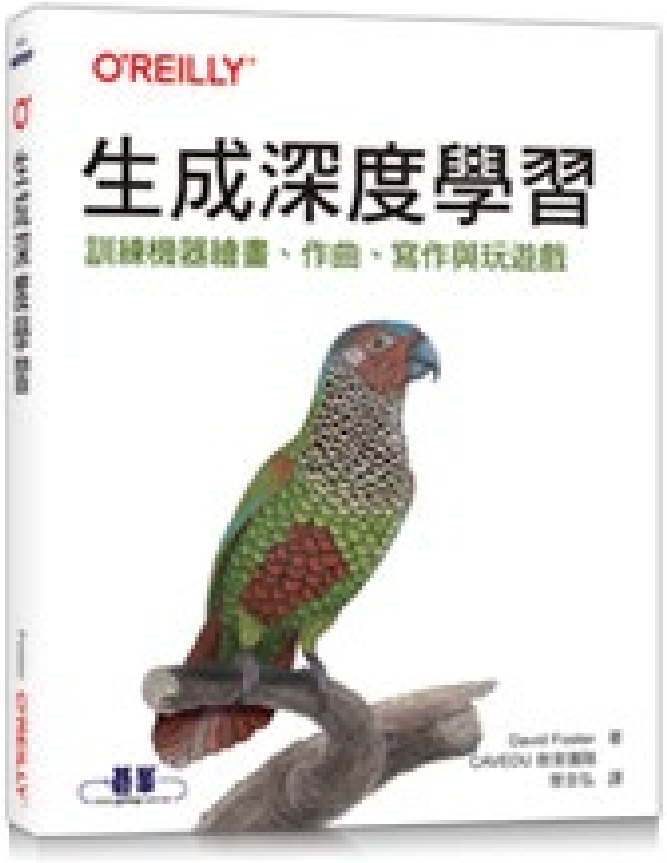
The CIFAR-10 dataset contains 60,000 32x32 color images in 10 different classes.
There are 6,000 images of each class.



# 參考解答

生成深度學習｜訓練機器繪畫、作曲、寫作與玩遊戲
(Generative Deep Learning)
David Foster 著 CAVEDU 教育團隊 曾吉弘 譯
歐萊禮 2020-05-26

**Released June 2019**




2. Deep Learning
Structured and Unstructured Data
Deep Neural Networks
Keras and TensorFlow

Your First Deep Neural Network

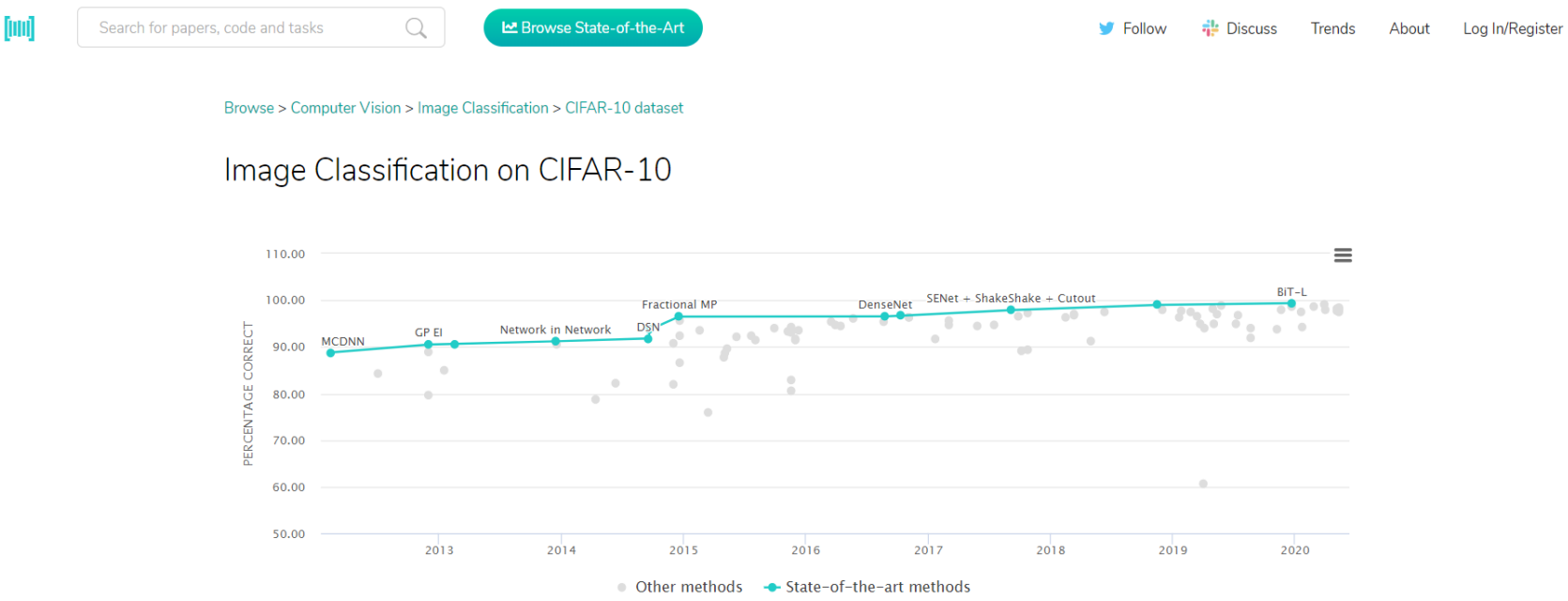
Improving the Model
- Convolutional Layers
- Batch Normalization
- Dropout Layers
- Putting It All Together

https://www.oreilly.com/library/view/generative-deep-learning/9781492041931/

```
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.layers import Input, Flatten, Dense, Conv2D
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical

from tensorflow.keras.datasets import cifar10

(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```

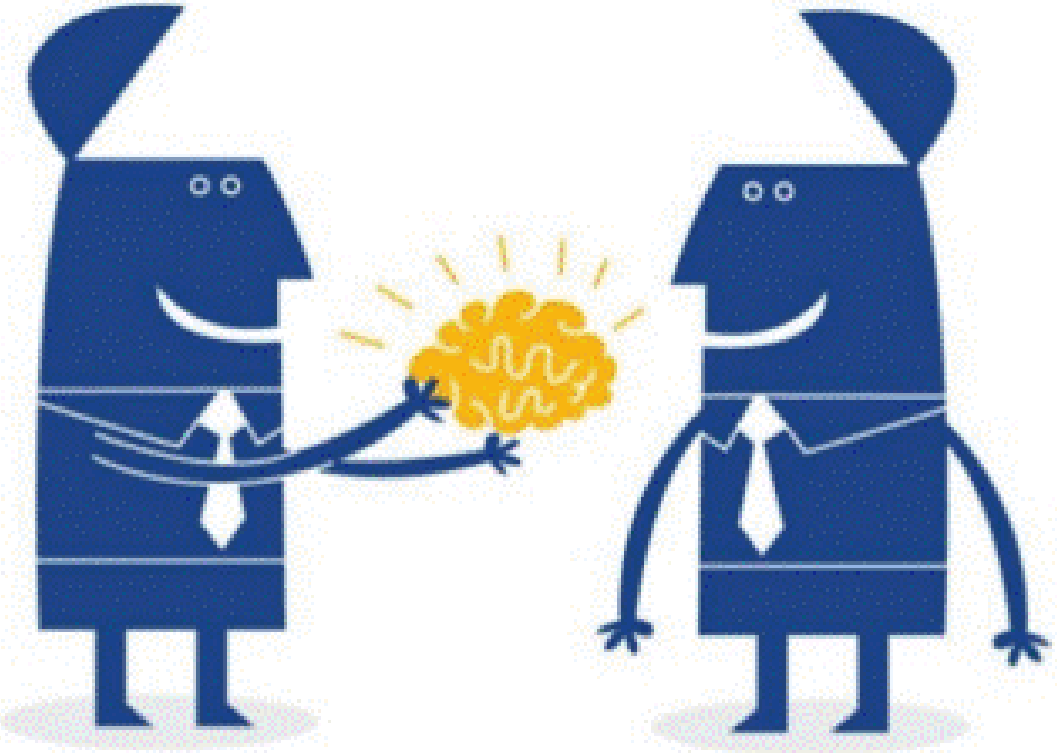
# 世界紀錄 State-of-the-Art

https://paperswithcode.com/sota/image-classification-on-cifar-10

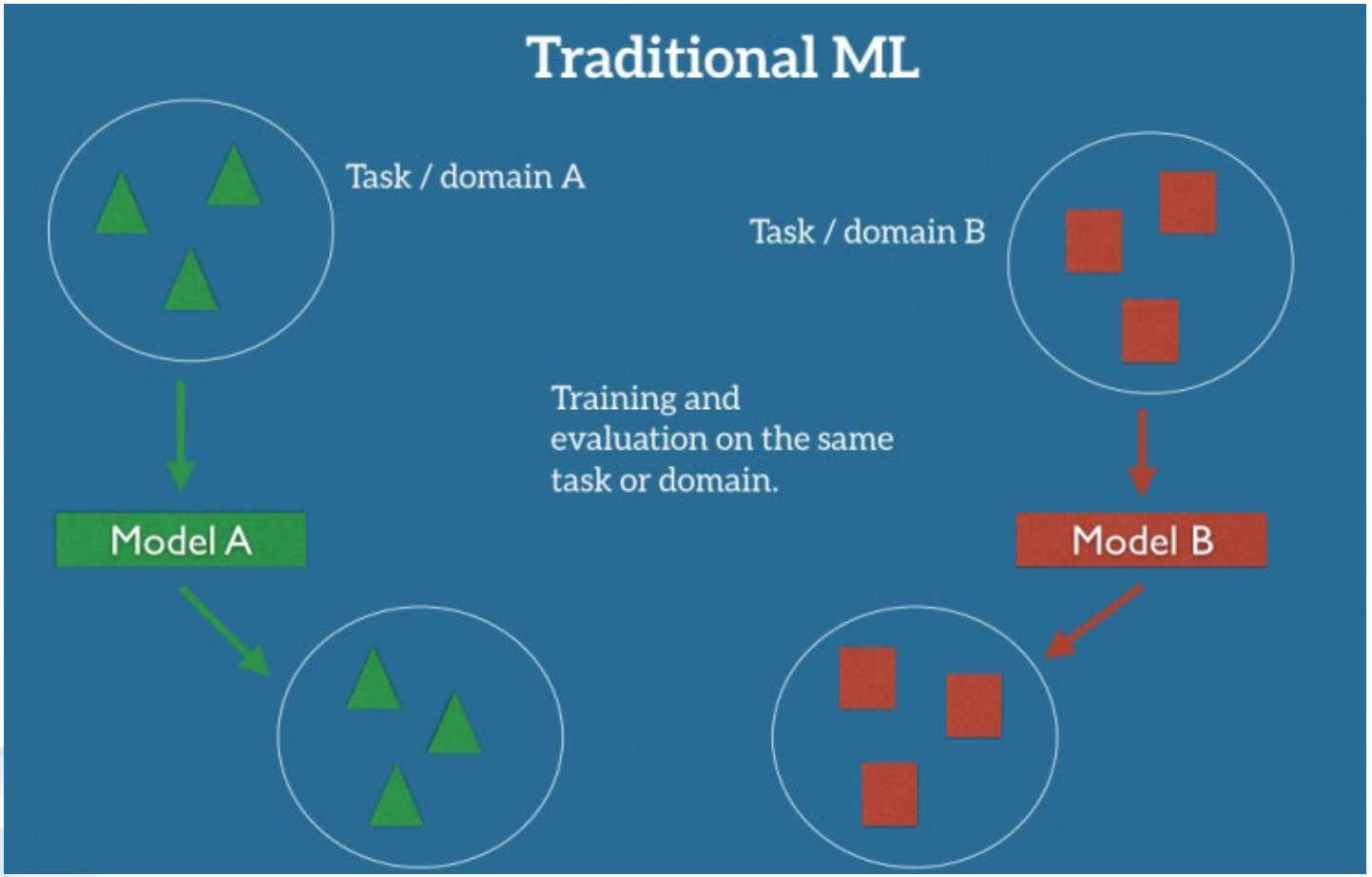


# Transfer Learning
# 遷移學習

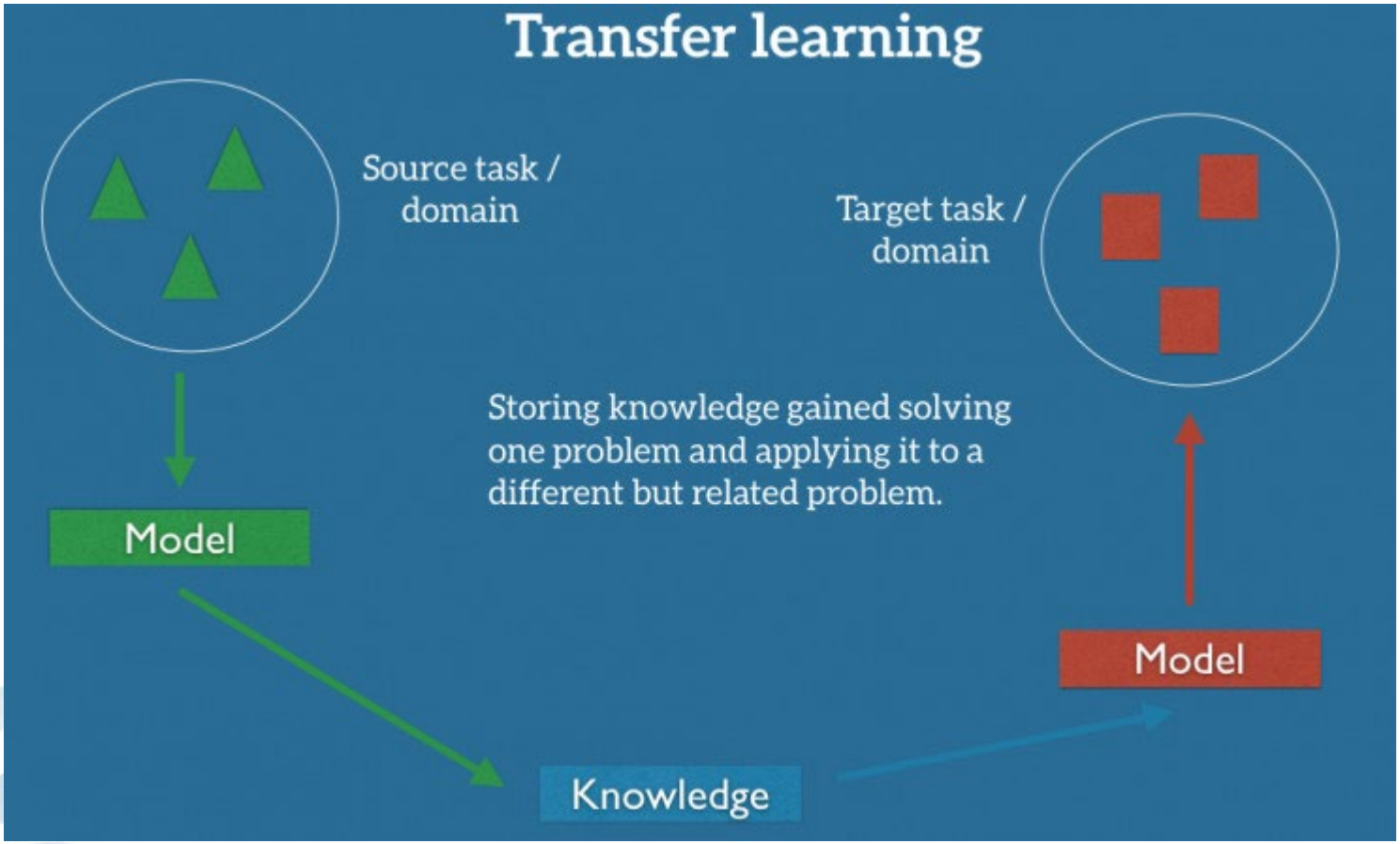
# TRANSFER OF LEARNING

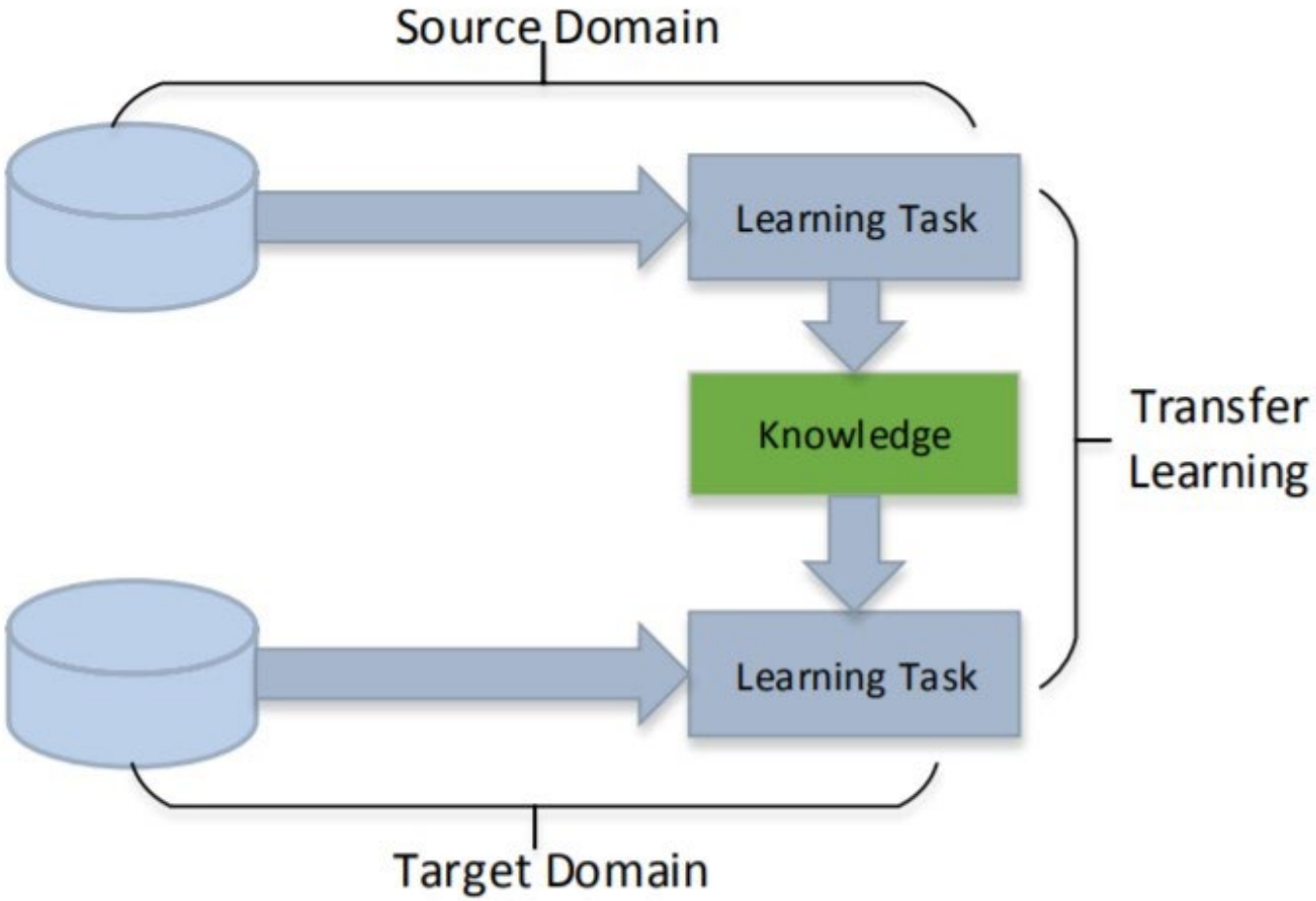


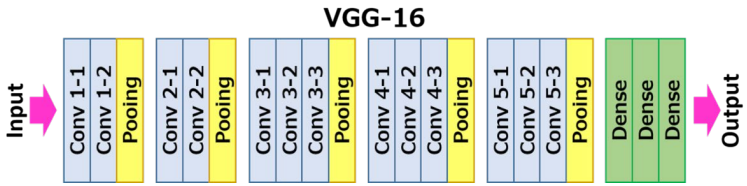
The application of skills, knowledge, and/or attitudes that were learned in one situation to another **learning** situation (Perkins, 1992)




https://codertw.com/程式語言/449190/


Transfer learning
Source task / domain
Target task / domain
Storing knowledge gained solving one problem and applying it to a different but related problem.
Model
Model
Knowledge


Source Domain
Learning Task
Knowledge
Transfer Learning
Learning Task
Target Domain




**Classifier**
**分類器**
**全連接層**

conv

conv

conv

conv

conv

**INPUT DATA**

**Classifier**
**分類器**
**全連接層**

conv

conv

conv

conv

conv

**YOUR DATA**

重新訓練
**分類器**

**微調權重**

**高階特徵**

**凍結權重**

**低階特徵**

方法1:先萃取出資料的特徵，再拿來訓練分類器

方法2:將經典CNN 移植到新模型之中

方法3:微調(Fine-Tuning)預先訓練的CNN網路

```
from tensorflow.keras.applications import VGG16

vgg16 = VGG16(include_top=False,
              weights='imagenet',
              input_shape=(150,150,3),
             )

vgg16.summary()
```

```
# ---- 建立並訓練密集層分類器 ---- #

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.applications import VGG16

vgg16 = VGG16(include_top=False,
              weights='imagenet',
              input_shape=(150,150,3))

model = Sequential()
model.add(vgg16)   # 將 vgg16 做為一層
model.add(Flatten())
model.add(Dense(512, activation='relu', input_dim=4 * 4 * 512))
model.add(Dropout(0.5))  # 丟棄法
model.add(Dense(1, activation='sigmoid'))

vgg16.trainable = False     # 凍結權重
model.summary()

model.compile(optimizer=RMSprop(lr=2e-5),   # 學習速率降低一點
              loss='binary_crossentropy',
              metrics=['acc'])
```

Transfer Learning方法2:
將經典CNN 移植到新模型之中

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.applications import VGG16

vgg16 = VGG16(include_top=False,
              weights='imagenet',
              input_shape=(150,150,3))
```

Transfer Learning方法3:
微調(Fine-Tuning)預先訓練的CNN網路

```
unfreeze = ['block5_conv1', 'block5_conv2', 'block5_conv3'] # 最後 3 層的名稱

for layer in vgg16.layers:
    if layer.name in unfreeze:
        layer.trainable = True  # 最後 3 層解凍
    else:
        layer.trainable = False # 其他凍結權重

vgg16.summary()
```

# CNN in security

# CNN Malware Detection

https://arxiv.org/search/?query=CNN+Malware+Detection+&searchtype=all&source=header

# CNN and malware detection

https://arxiv.org/abs/2002.06383



**Computer Science > Cryptography and Security**

*[Submitted on 15 Feb 2020]*

## Analyzing CNN Based Behavioural Malware Detection Techniques on Cloud IaaS

Andrew McDole, Mahmoud Abdelsalam, Maanak Gupta, Sudip Mittal

Cloud Infrastructure as a Service (IaaS) is vulnerable to malware due to its exposure to external adversaries, making it a lucrative attack vector for malicious actors. A datacenter infected with malware can cause data loss and/or major disruptions to service for its users. This paper analyzes and compares various Convolutional Neural Networks (CNNs) for online detection of malware in cloud IaaS. The detection is performed based on behavioural data using process level performance metrics including cpu usage, memory usage, disk usage etc. We have used the state of the art DenseNets and ResNets in effectively detecting malware in online cloud system. CNN are designed to extract features from data gathered from a live malware running on a real cloud environment. Experiments are performed on OpenStack (a cloud IaaS software) testbed designed to replicate a typical 3-tier web architecture. Comparative analysis is performed for different metrics for different CNN models used in this research.

# https://arxiv.org/abs/2003.12805

## Real-Time Detection of Dictionary DGA Network Traffic using Deep Learning

Kate Highnam, Domenic Puzio, Song Luo, Nicholas R. Jennings

Botnets and malware continue to avoid detection by static rules engines when using domain generation algorithms (DGAs) for callouts to unique, dynamically generated web addresses. Common DGA detection techniques fail to reliably detect DGA variants that combine random dictionary words to create domain names that closely mirror legitimate domains. To combat this, we created a novel hybrid neural network, Bilbo the `bagging` model, that analyses domains and scores the likelihood they are generated by such algorithms and therefore are potentially malicious. Bilbo is the first parallel usage of a convolutional neural network (CNN) and a long short-term memory (LSTM) network for DGA detection. Our unique architecture is found to be the most consistent in performance in terms of AUC, F1 score, and accuracy when generalising across different dictionary DGA classification tasks compared to current state-of-the-art deep learning architectures. We validate using reverse-engineered dictionary DGA domains and detail our real-time implementation strategy for scoring real-world network logs within a large financial enterprise. In four hours of actual network traffic, the model discovered at least five potential command-and-control networks that commercial vendor tools did not flag.

Bilbo is the first parallel usage of a convolutional neural network (CNN) and a long short-term memory (LSTM) network for DGA detection

# https://arxiv.org/abs/1910.10958

deep learning based malware detection (DLMD)

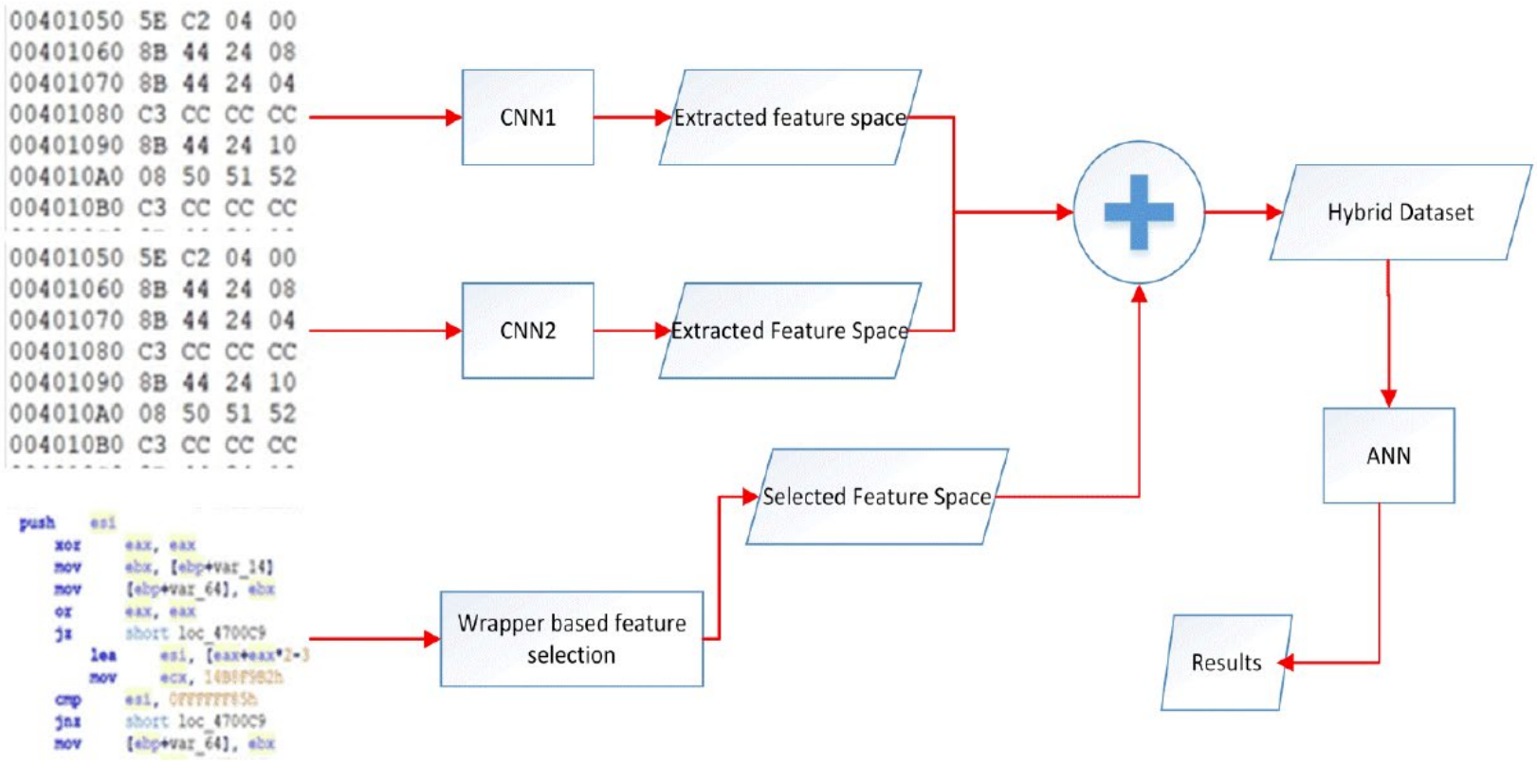
## Malware Classification using Deep Learning based Feature Extraction and Wrapper based Feature Selection Technique

Muhammad Furqan Rafique, Muhammad Ali, Aqsa Saeed Qureshi, Asifullah Khan, Anwar Majid Mirza

In case of behavior analysis of a malware, categorization of malicious files is an essential part after malware detection. Numerous static and dynamic techniques have been reported so far for categorizing malwares. This research work presents a deep learning based malware detection (DLMD) technique based on static methods for classifying different malware families. The proposed DLMD technique uses both the byte and ASM files for feature engineering and thus classifying malwares families. First, features are extracted from byte files using two different types of Deep Convolutional Neural Networks (CNN). After that, important and discriminative opcode features are selected using a wrapper-based mechanism, where Support Vector Machine (SVM) is used as a classifier. The idea is to construct a hybrid feature space by combining the different feature spaces in order that the shortcoming of a particular feature space may be overcome by another feature space. And consequently to reduce the chances of missing a malware. Finally, the hybrid feature space is then used to train a Multilayer Perceptron, which classifies all the nine different malware families. Experimental results show that proposed DLMD technique achieves log-loss of 0.09 for ten independent runs. Moreover, the performance of the proposed DLMD technique is compared against different classifiers and shows its effectiveness in categorizing malwares. The relevant code and database can be found at this https URL.

https://github.com/cyberhunters/Malware-Detection-Using-Machine-Learning

# DLMD technique



# DataSet 資料集

| Class | Frequency |
|---|---|
| Ramnit | 1541 |
| Lollipop | 2478 |
| Kelihos_ver3 | 2942 |
| Vundo | 475 |
| Simda | 42 |
| Tracur | 751 |
| Kelihos_ver1 | 398 |
| Obfuscator.ACY | 1228 |
| Gatak | 1013 |

# https://arxiv.org/abs/2206.08004



**Computer Science > Cryptography and Security**

*[Submitted on 16 Jun 2022]*

## When a RF Beats a CNN and GRU, Together -- A Comparison of Deep Learning and Classical Machine Learning Approaches for Encrypted Malware Traffic Classification

Adi Lichy, Ofek Bader, Ran Dubin, Amit Dvir, Chen Hajaj

Internet traffic classification is widely used to facilitate network management. It plays a crucial role in Quality of Services (QoS), Quality of Experience (QoE), network visibility, intrusion detection, and traffic trend analyses. While there is no theoretical guarantee that deep learning (DL)-based solutions perform better than classic machine learning (ML)-based ones, DL-based models have become the common default. This paper compares well-known DL-based and ML-based models and shows that in the case of malicious traffic classification, state-of-the-art DL-based solutions do not necessarily outperform the classical ML-based ones. We exemplify this finding using two well-known datasets for a varied set of tasks, such as: malware detection, malware family classification, detection of zero-day attacks, and classification of an iteratively growing dataset. Note that, it is not feasible to evaluate all possible models to make a concrete statement, thus, the above finding is not a recommendation to avoid DL-based models, but rather empirical proof that in some cases, there are more simplistic solutions, that may perform even better.

# https://arxiv.org/abs/2107.04008



**Computer Science > Cryptography and Security**

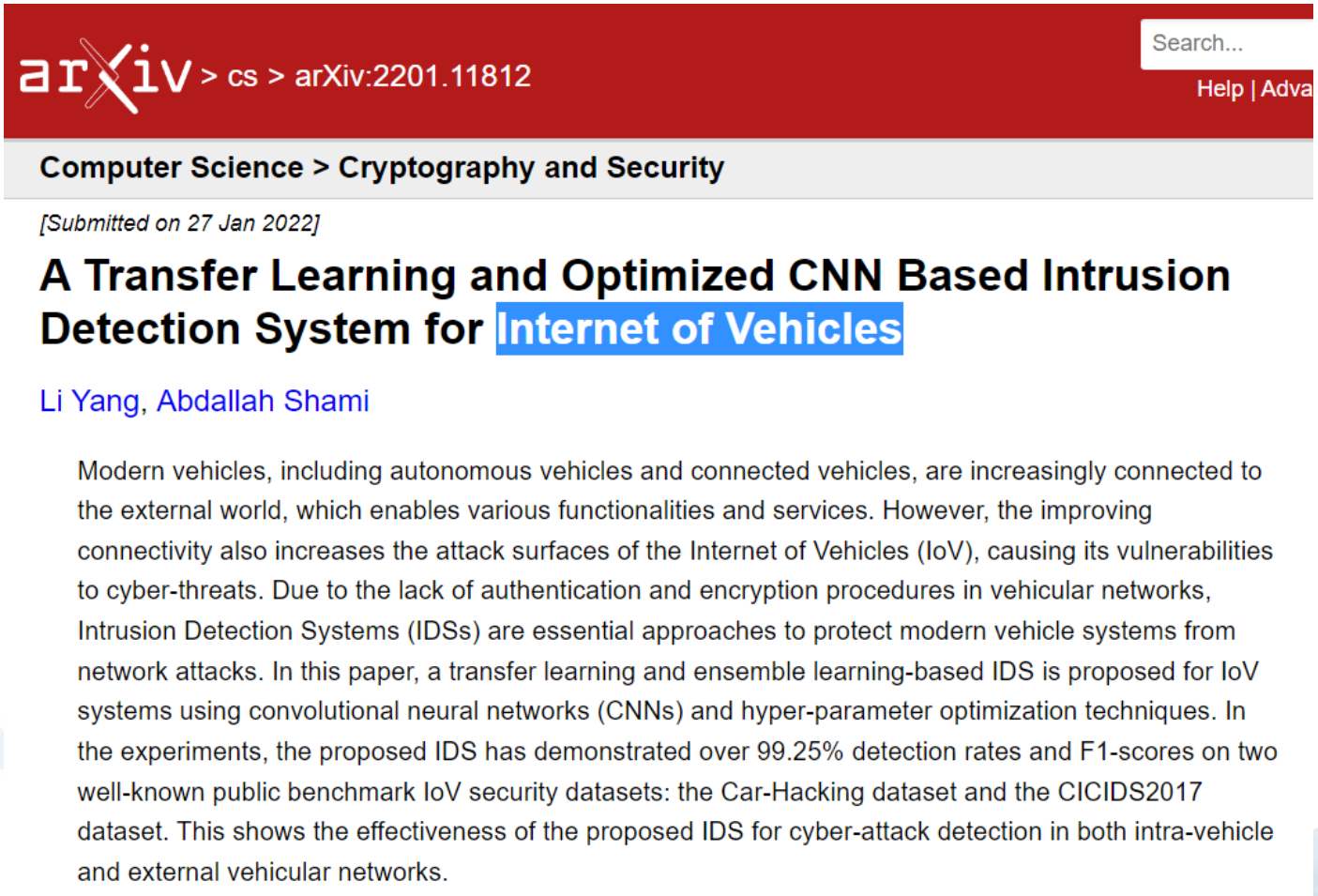
*[Submitted on 8 Jul 2021]*

## Malware Classification Using Deep Boosted Learning

Muhammad Asam, Saddam Hussain Khan, Tauseef Jamal, Umme Zahoora, Asifullah Khan

Malicious activities in cyberspace have gone further than simply hacking machines and spreading viruses. It has become a challenge for a nations survival and hence has evolved to cyber warfare. Malware is a key component of cyber-crime, and its analysis is the first line of defence against attack. This work proposes a novel deep boosted hybrid learning-based malware classification framework and named as Deep boosted Feature Space-based Malware classification (DFS-MC). In the proposed framework, the discrimination power is enhanced by fusing the feature spaces of the best performing customized CNN architectures models and its discrimination by an SVM for classification. The discrimination capacity of the proposed classification framework is assessed by comparing it against the standard customized CNNs. The customized CNN models are implemented in two ways: softmax classifier and deep hybrid learning-based malware classification. In the hybrid learning, Deep features are extracted from customized CNN architectures and fed into the conventional machine learning classifier to improve the classification performance. We also introduced the concept of transfer learning in a customized CNN architecture based malware classification framework through fine-tuning. The performance of the proposed malware classification approaches are validated on the MalImg malware dataset using the hold-out cross-validation technique. Experimental comparisons were conducted by employing innovative, customized CNN, trained from scratch and fine-tuning the customized CNN using transfer learning. The proposed classification framework DFS-MC showed improved results, Accuracy: 98.61%, F-score: 0.96, Precision: 0.96, and Recall: 0.96.

# https://arxiv.org/abs/2201.11812



arXiv > cs > arXiv:2201.11812

Search...

Help | Adva

**Computer Science > Cryptography and Security**

*[Submitted on 27 Jan 2022]*

## A Transfer Learning and Optimized CNN Based Intrusion Detection System for Internet of Vehicles

Li Yang, Abdallah Shami

Modern vehicles, including autonomous vehicles and connected vehicles, are increasingly connected to the external world, which enables various functionalities and services. However, the improving connectivity also increases the attack surfaces of the Internet of Vehicles (IoV), causing its vulnerabilities to cyber-threats. Due to the lack of authentication and encryption procedures in vehicular networks, Intrusion Detection Systems (IDSs) are essential approaches to protect modern vehicle systems from network attacks. In this paper, a transfer learning and ensemble learning-based IDS is proposed for IoV systems using convolutional neural networks (CNNs) and hyper-parameter optimization techniques. In the experiments, the proposed IDS has demonstrated over 99.25% detection rates and F1-scores on two well-known public benchmark IoV security datasets: the Car-Hacking dataset and the CICIDS2017 dataset. This shows the effectiveness of the proposed IDS for cyber-attack detection in both intra-vehicle and external vehicular networks.

https://www.hindawi.com/journals/wcmc/2021/3697536/

**Research Article | Open Access**
Volume 2021 | Article ID 3697536 | https://doi.org/10.1155/2021/3697536

Show citation

# Network Threat Detection Based on Group CNN for Privacy Protection

Yanping Xu,[1] Xia Zhang,[1] **Chengdan Lu**,[2] Zhenliang Qiu,[1] Chunfang Bi,[3] Yuping Lai,[4] Jian Qiu,[5] and Hua Zhang[6]

Show more

**Academic Editor:** Yaguang Lin

| Received | Revised | Accepted | Published |
|---|---|---|---|
| 08 Jun 2021 | 15 Jul 2021 | 03 Aug 2021 | 03 Sep 2021 |

## Abstract

The Internet of Things (IoT) contains a large amount of data, which attracts various types of network attacks that lead to privacy leaks. With the upgrading of network attacks and the increase in network security data, traditional machine learning methods are no longer suitable for network threat detection. At the same time, data analysis techniques and deep learning algorithms have developed rapidly and have been successfully applied to a variety of tasks for privacy protection. Convolutional neural networks (CNNs) are typical deep learning models that can learn and reconstruct features accurately and efficiently. Therefore, in this paper, we propose a group CNN models that is based on feature correlations to learn features and reconstruct security data. First, feature correlation coefficients are computed to measure the relationships among the features. Then, we sort the correlation coefficients in descending order and group the data by columns. Second, a 1D group CNN model with multiple 1D convolution kernels and 1D pooling filters is built to address the grouped data for feature learning and reconstruction. Third, the reconstructed features are input to shadow machine learning models for network threat prediction. The experimental results show that features reconstructed by the group CNN can reduce the dimensions and achieve the best performance compared to the other present dimension reduction algorithms. At the same time, the group CNN can decrease the floating point of operations (FLOP), parameters, and running time compared to the basic 1D CNN.