Day 2: Hands-on AI

Part 2: Building Deep Learning Model

2021

2.1:
Building Deep Learning Model

Contents

2.1 Building Deep Learning Model

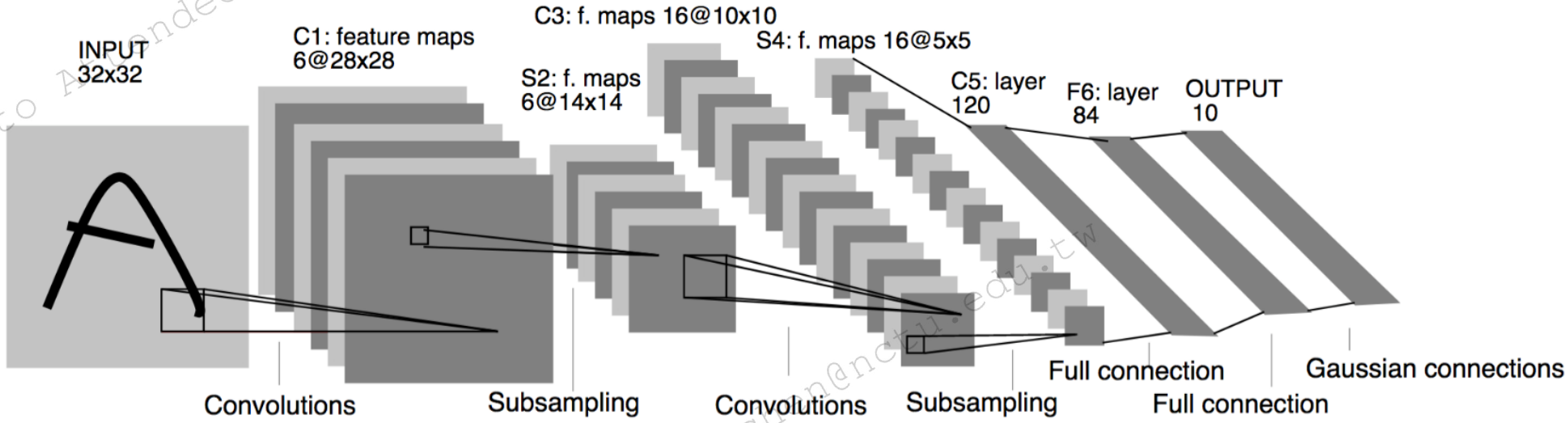
◆ 2.1.1 Classic models

- LeNet
- AlexNet
- VGG
- Network in Network (NiN)
- GoogLeNet
- ResNet
- DenseNet

Qualcomm

2.2.1 Classic models: LeNet

- LeNet was designed by Yann LeCun in AT&T Bell Labs in 1990s for handwritten digits recognition. It's the first time that CNNs reach state-of-the-art performance in the time. (Comparing with SVMs)



Gradient-based learning applied to document recognition, 1998

LeNet

- Modern deep learning models are significantly affected by design ideas in LeNet. Here are some key features of LeNet:

1. The model parameters are initialized randomly and trained by back-propagation.
2. There are no manual design of feature extraction. The raw image are fed into CNN directly and features are automatically extracted and classified.
3. A pooling layer is always after a convolution layer with non-linear activation layer.
4. It contained 5 layers if we consider [conv - activation - pooling] layers as one *.

* Sometimes we don't take pooling layers into consideration when we count total layers in a network, because there is no parameter in pooling layer.

Classic models: AlexNet

- ImageNet dataset was first released in 2009 with 14 million examples and 1000 categories by Fei-Fei Li in Stanford.

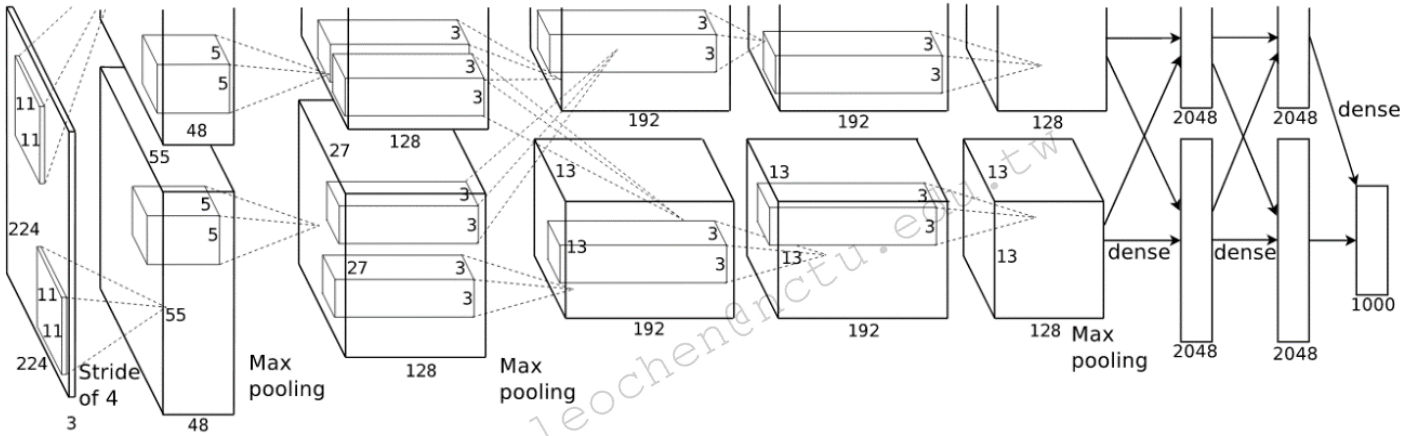
- AlexNet was proposed by Alex Krizhevsky in 2012 and won the ImageNet Large Scale Visual Recognition Challenge 2012 competition by a larger margin (error rates: 15.3% of AlexNet VS. 26.2% of 2nd place).

- AlexNet was considered as the breakthrough of Deep learning on Computer Vision and the marks of the beginning of Renaissance of Artificial Neural Network.

AlexNet

◆ Key points:

- 1. Why two parallel parts (one up and one down) in the architecture figure? Because it's trained on two GTX 580 GPUs originally by the author. But now we can train it on one GPU benefit from the advances of computing power.
- 2. AlexNet is very similar with LeNet but much deeper than it. (5 conv layers and 3 fc layers.) In addition, activation function is ReLU instead of sigmoid.
- 3. Dropout and data augmentation are firstly introduced.

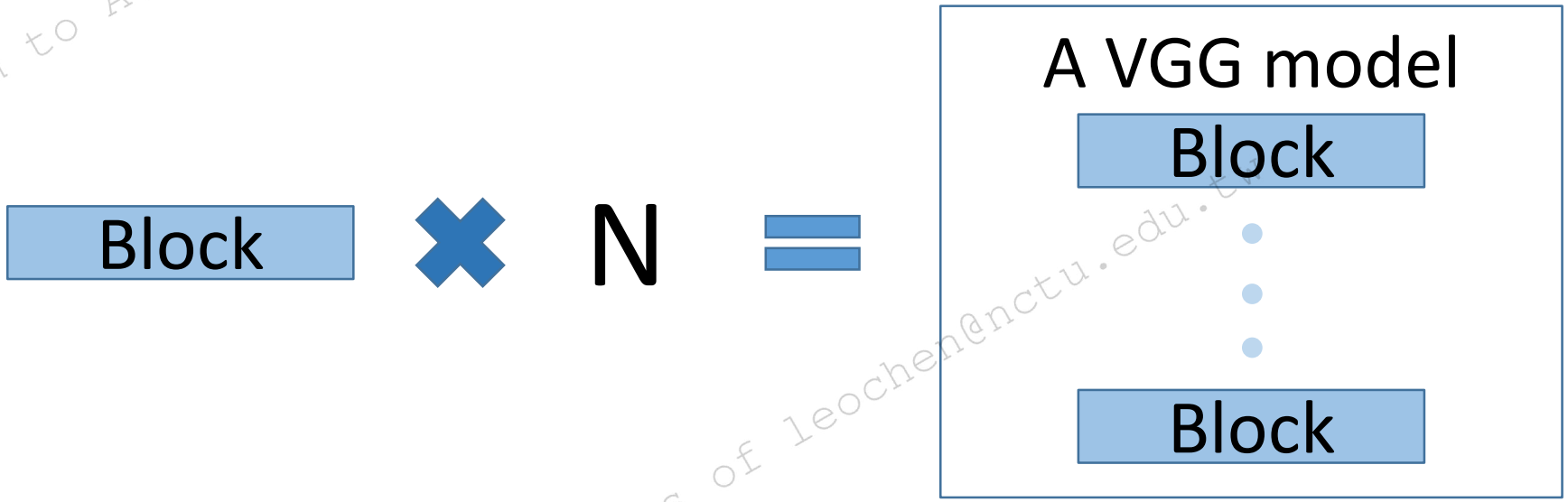


ImageNet classification with deep convolutional neural networks, NIPS 2012

Qualcomm

Classic models: VGG

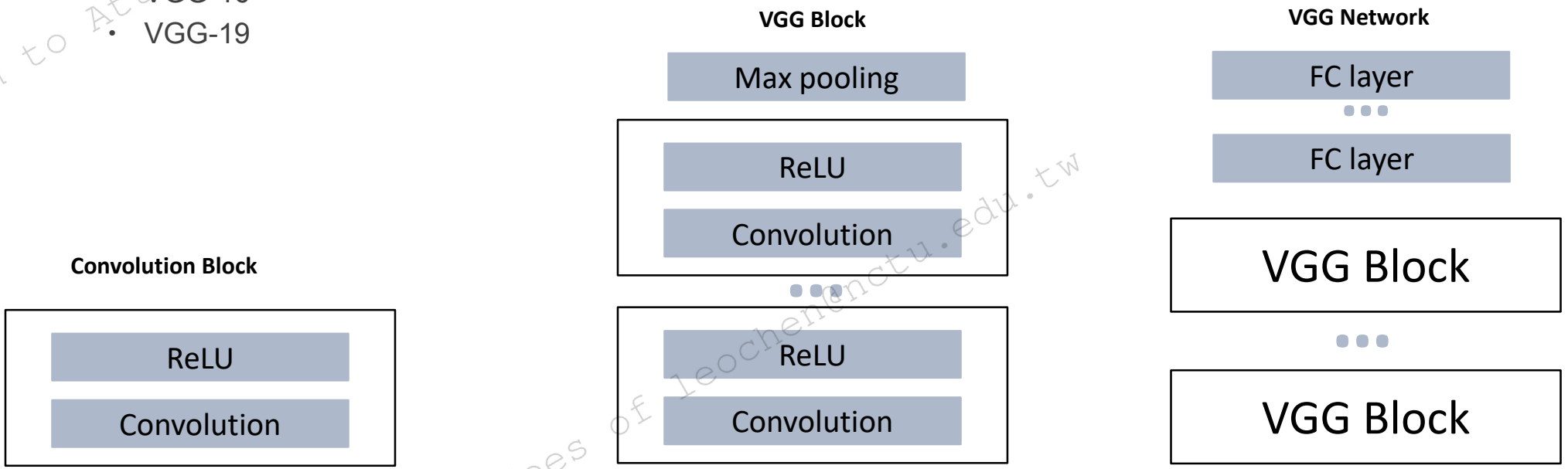
- VGG was named by Visual Geometry Group (VGG) at Oxford University. It reflected the thinking of using blocks to construct deep network.
- Blocks are some well-designed layers, the structure and hyper-parameters of blocks have fixed rules to follow. Blocks are regularly repeating to build network, which provides an effective way to design a deeper model.



VGG

◆ Key patterns:

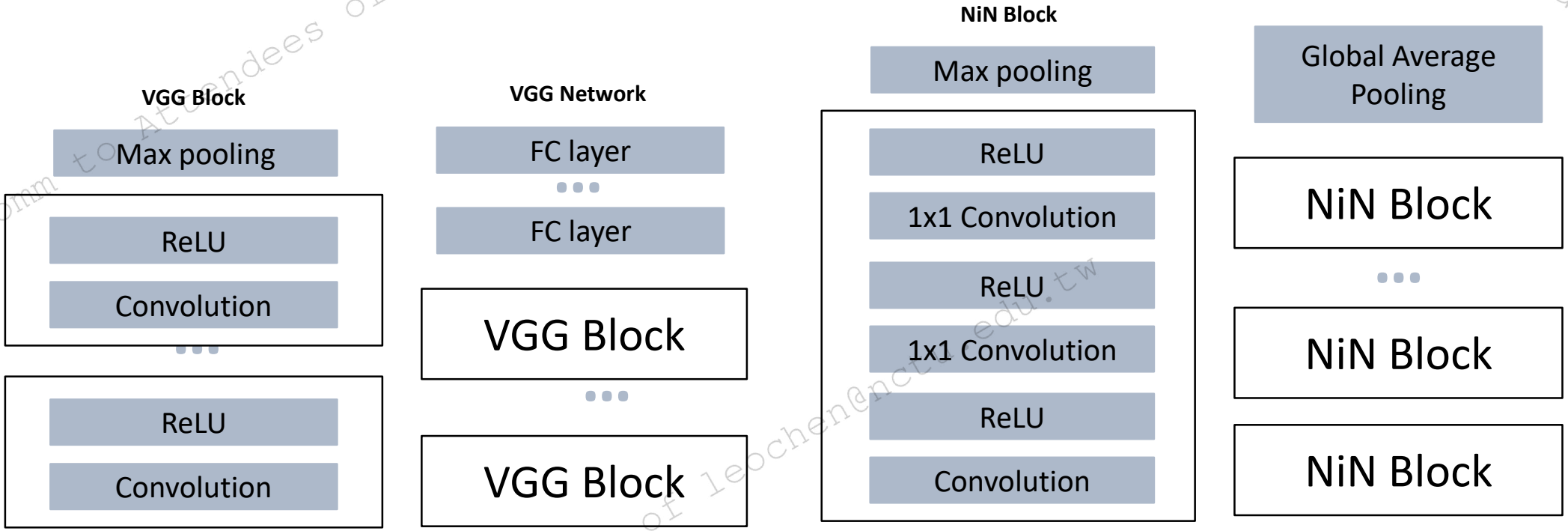
1. Small convolution kernels are used, like 3*3 and 1*1.
2. A ReLU activation function is always after a Convolution layer, which is considered as a Convolution Block.
3. A VGG Block consists of several Convolution Block and one Pooling layer, which reduces the size of feature map to half. The whole network are composed by several VGG Block and one or more FC layers.
4. Deeper than AlexNet (8 layers):
   - VGG-11
   - VGG-16
   - VGG-19



Qualcomm

Network in Network

- Each block is a small network, it contains convolution and FC layers (1x1 convolution).
- Network in Network (NiN) improved the basic block by put in FC layers that appear at the end of previous Network. In addition, NiN used global average pooling instead of FC layers as classifier.



1x1 convolution

- You may ask: Where are FC layers in NiN blocks? Only 1x1 convolution layers appear!
- Isn't 1x1 convolution layer meaningless?

1	2	1	0
0	2	1	2
1	2	1	2
1	2	1	1

*

2

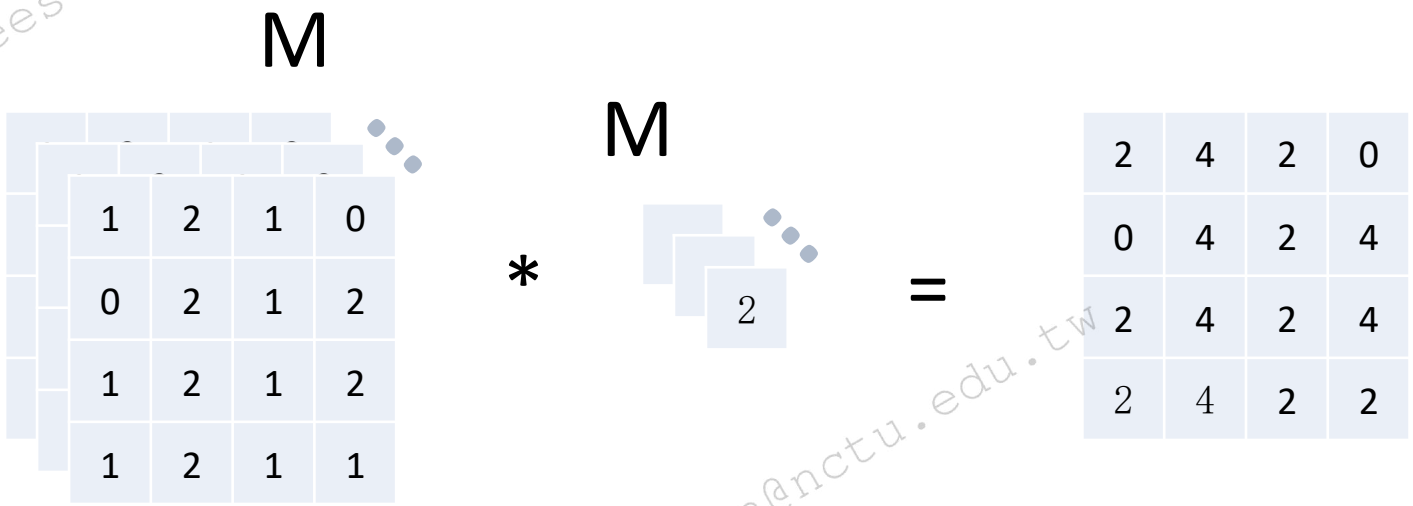
=

2	4	2	0
0	4	2	4
2	4	2	4
2	4	2	2

Qualcomm

1x1 convolution

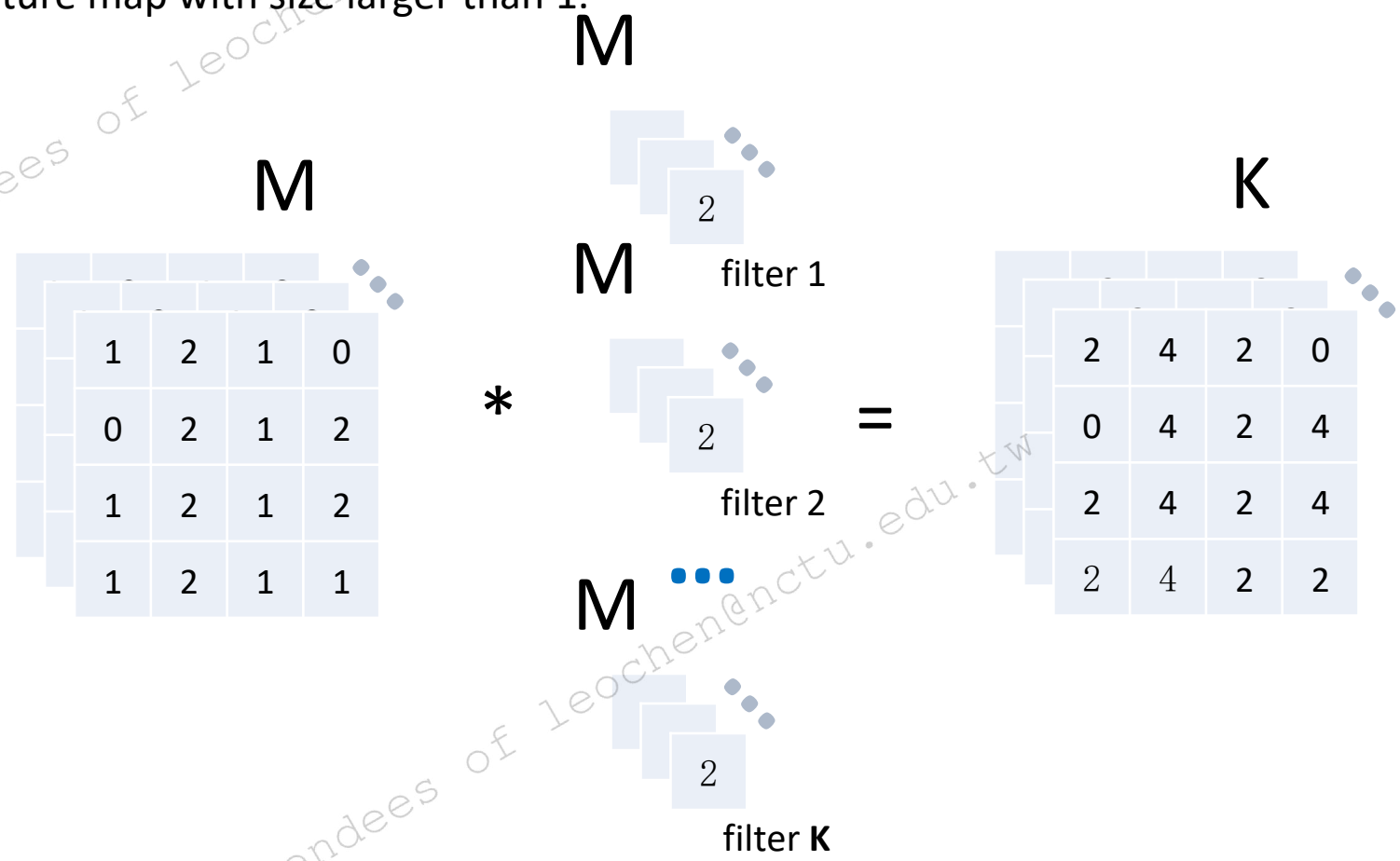
- **Case 1**: Input feature map with size larger than 1.



Qualcomm

1x1 convolution

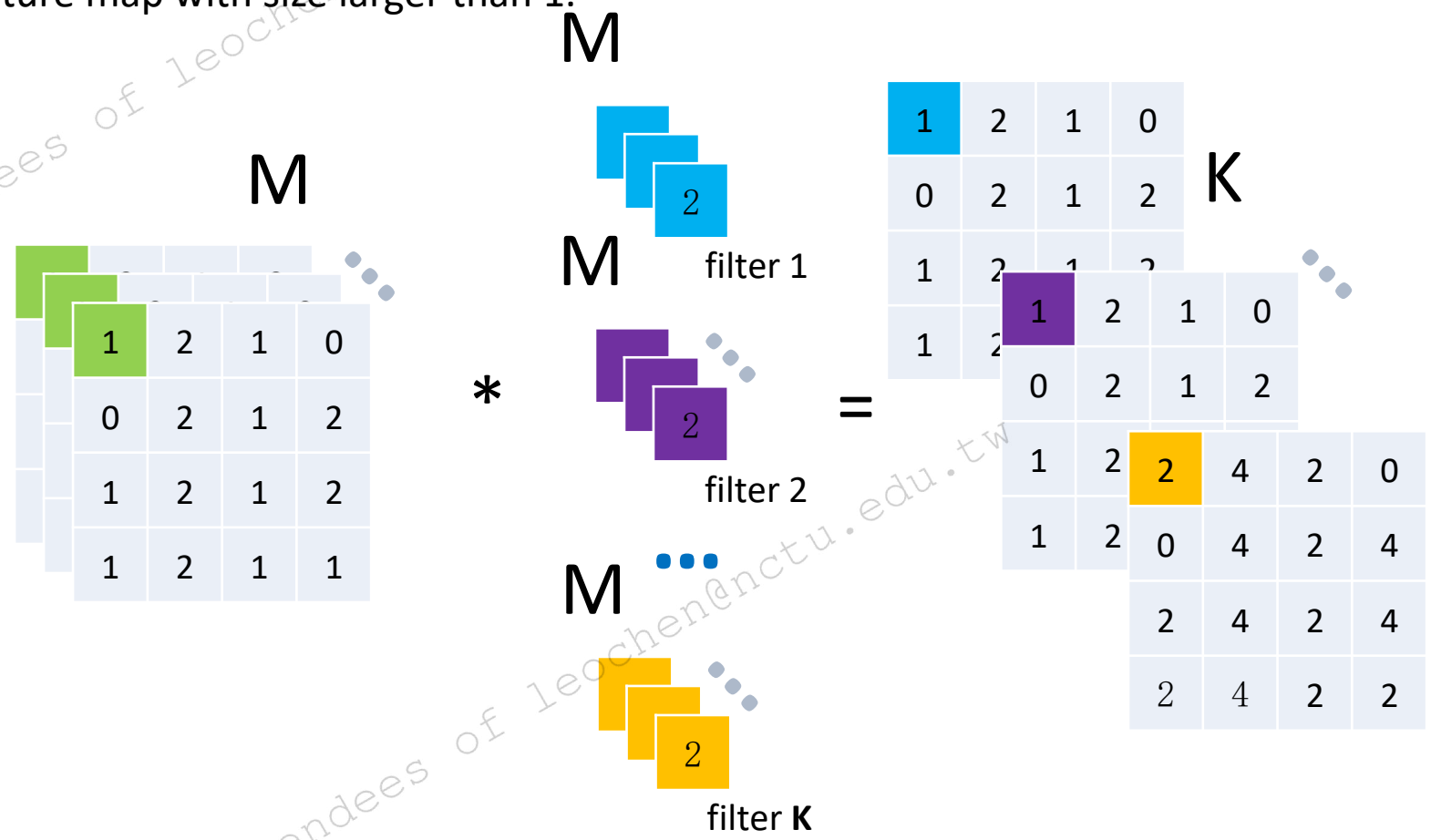
- **Case 1**: Input feature map with size larger than 1.



Qualcomm

1x1 convolution

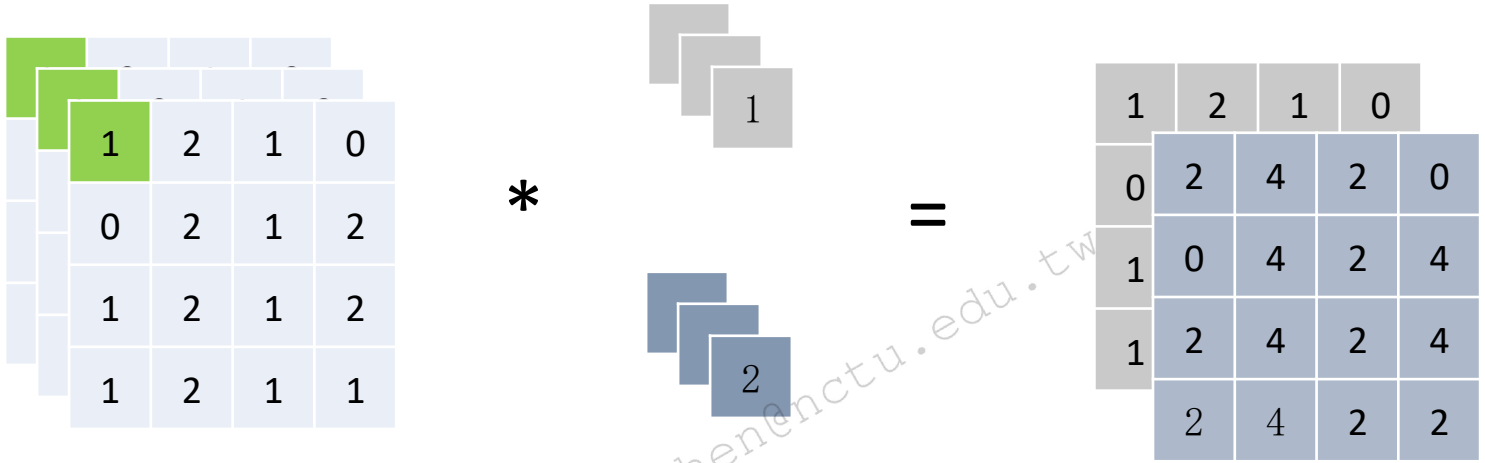
- **Case 1**: Input feature map with size larger than 1.



Qualcomm

1x1 convolution

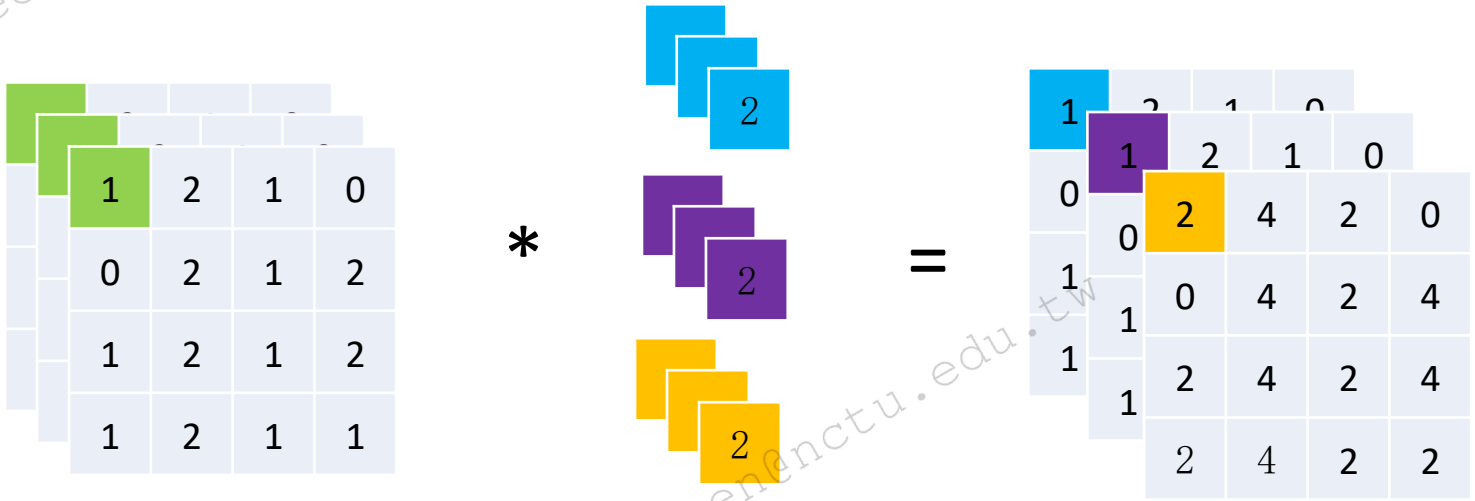
- **Case 1**: Input feature map with size larger than 1.
  - 1. Reduce or increase the dimension.



Qualcomm

1x1 convolution

- **Case 1**: Input feature map with size larger than 1.
  - 1. Reduce or increase the dimension.
  - 2. If you keep the dimension, 1x1 convolution is equivalent to a non-linear transformation.



Qualcomm

1x1 convolution

- **Case 2**: Input feature map with size 1, 1x1 convolution is equivalent to with FC layers.

VGG:

Input of 3 FC layers : (1, 512, 7, 7)
Output of FC 1: (1, 4096)
Output of FC 2: (1, 4096)
Output of FC 3: (1, 1000)

Number of parameters:
FC 1: 512 x 7 x 7 x 4096 = 102,760,448
FC 2: 4096 x 4096 = 16,777,216
FC 3: 4096 x 1000 = 4,096,000
Total = 123633664

Re-implementation by 1x1 convolution:

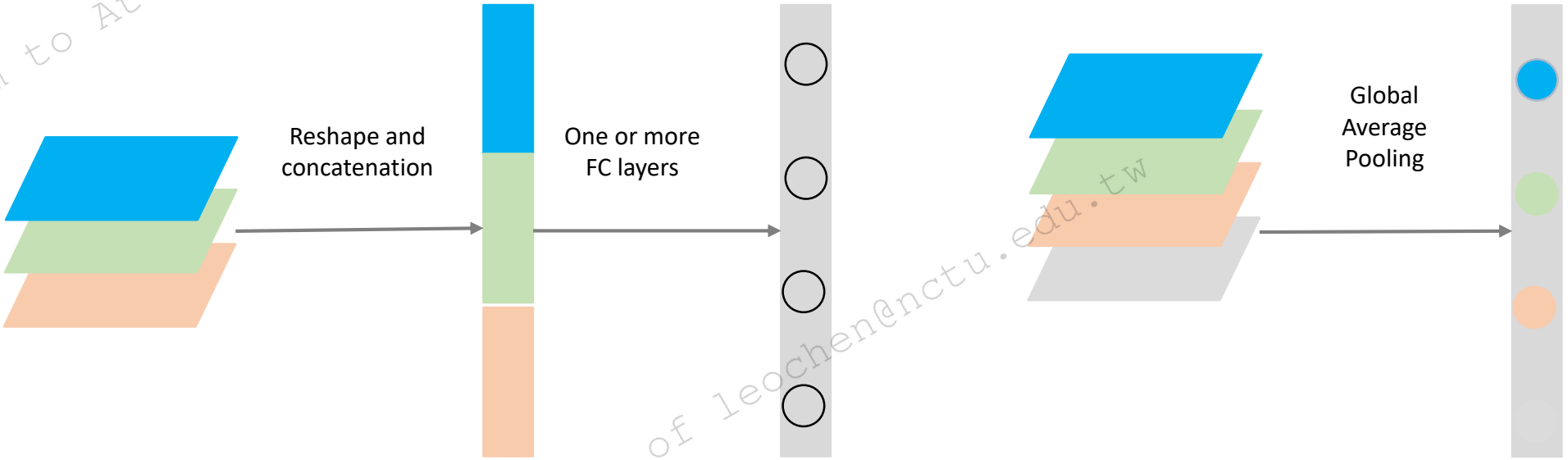
Input of classifier: (1, 512, 7, 7)
Output of 4096 conv kernels with size 7: (1, 4096)
Output of 4096 conv kernels with size 1: (1, 4096)
Output of 1000 conv kernels with size 1: (1, 1000)

Number of parameters:
Layer 1: 7 x 7 x 512 x 4096 = 102,760,448
Layer 2: 4096 x 1 x 1 x 4096 = 16,777,216
Layer 3: 4096 x 1 x 1 x 1000 = 4,096,000
Total = 123633664

Qualcomm

Global Average Pooling

- Fully connected layers for classification have a large amount of parameters (1, calculation, 2. overfitting), global average pooling achieve the same purpose with no parameters.
- The last NiN block generates as many as the number of categories. Global Average Pooling layer compute average of all values in each feature map as prediction of one category.



Qualcomm

GoogLeNet

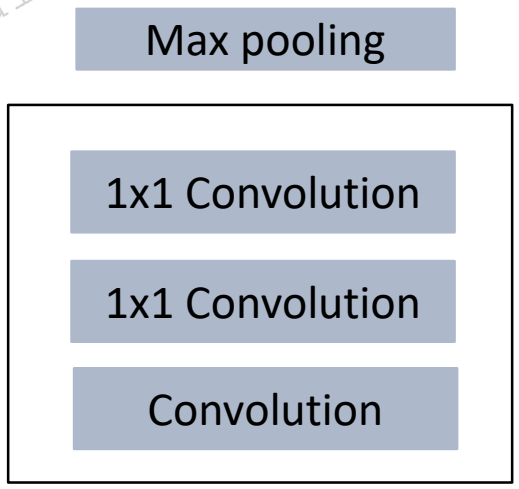
- GoogLeNet is proposed by Google and the uppercase letter L is paying tribute to LeNet by Lecun.
- The basic block of GoogLeNet is Inception ("We need to go deeper." from movie Inception).

GoogLeNet

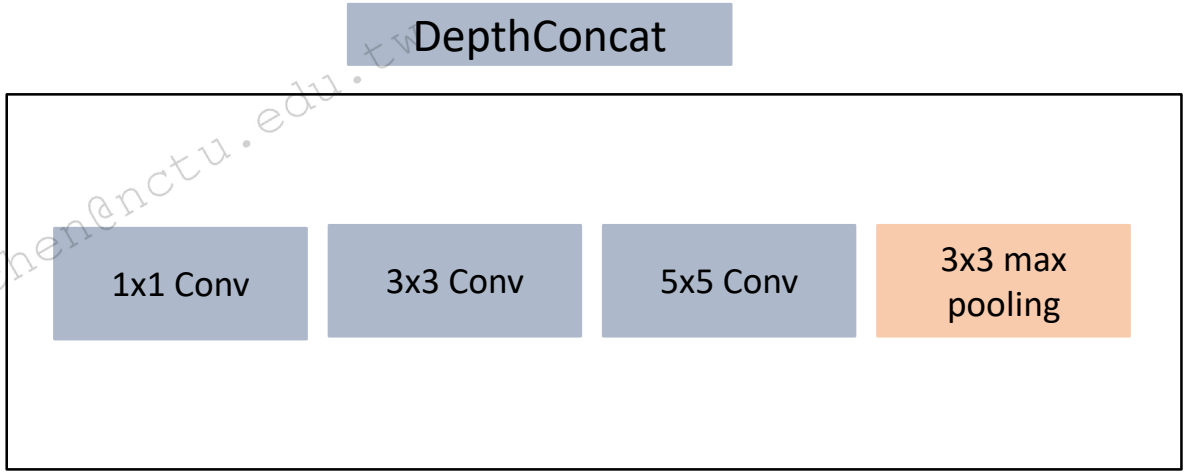
- Inception block wanted to use a combination of different sizes of conv kernels in parallel. By using multi-size filters, one layer can learn multi-scale features.

- ReLU appears after each convolution, so it is omitted for simplicity.

NiN Block



Inception Block Naive version



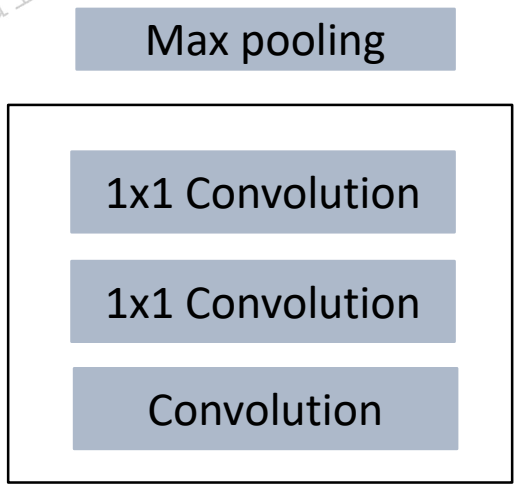
Qualcomm

GoogLeNet

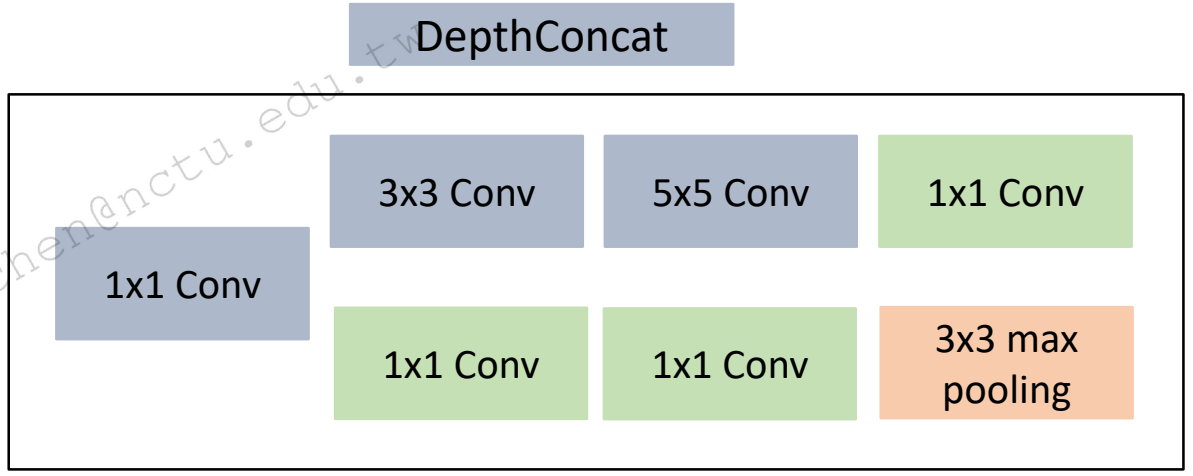
- Inception block wanted to use a combination of different size of conv kernels in parallel. By using multi-size filters, one layer can learn multi-scale features.

- ReLU appears after every convolution, so it is omitted for simplicity.

NiN Block



Inception Block



Qualcomm

GoogLeNet

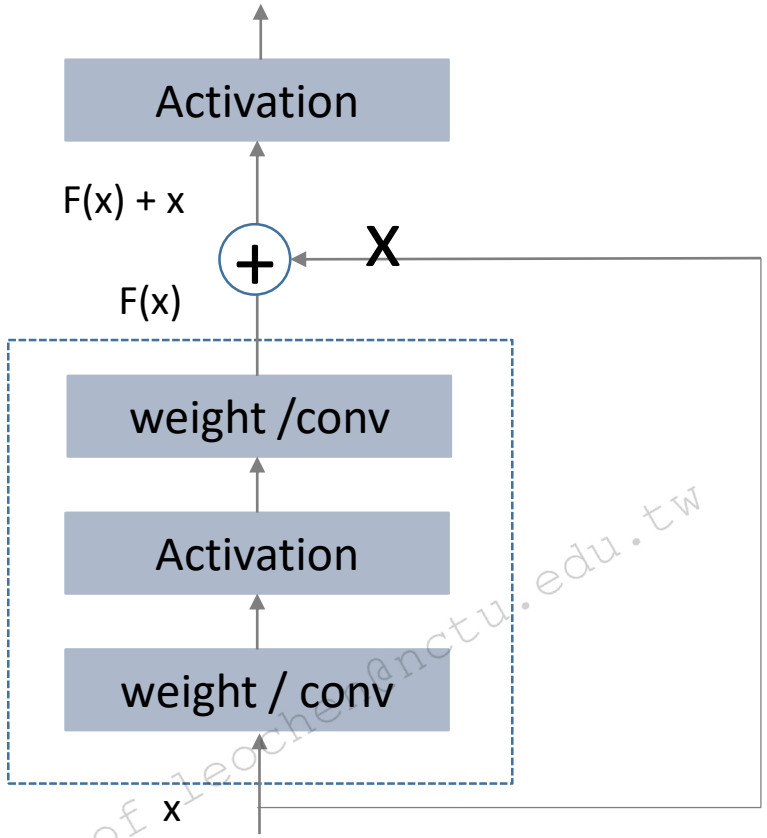
- Key points:
  - Why conv layers with different kernel size? To catch spatial features in multi scale.
  - Why add 1x1 conv ? 1x1 conv in the middle two paths reduce the dimension of depth. Max pooling doesn't change output channel, 1x1 conv in the last path is used to decrease the output dimension.
  - The input and output **the same height and width** benefit from appropriate padding values of each convolution layer. So the output of the four paths are cascaded along the depth. The depth of this four outputs are adjustable.
  - Why is it so strange? CNNs is still a black-box problem. This architecture is the best one of many candidates. Perhaps it is not the global optimal hyper-parameters.

ResNet

- We are facing a very real problem about the depth of the network.
  - Theoretically, the deeper the network, the stronger performance it has.
  - But in practice, the deeper network is **easy to overfitting**, and **hard to training** until convergence because of the vanishing gradient problem.
- How to resolve it?
  - Batch norm.
  - GoogLeNet use auxiliary networks to increase the gradient signal in training stage.
  - ResNet provide another way of thinking.

ResNet

- The key idea of ResNet is building short-cut connection, which is equivalent to providing a fast track for gradient of loss to back-propagate to front layer.

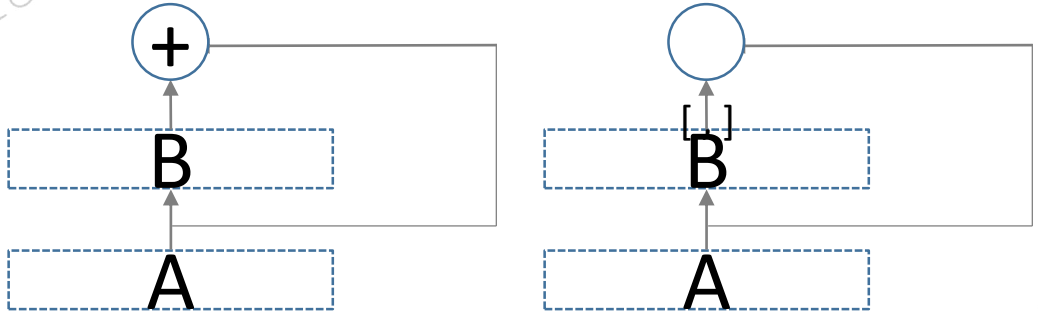


Qualcomm

DenseNet

- Someone[1] designed experiments and proved that, the shortcut connection plays a more important role. So an intuitive idea is, can we add more shortcut connections in a deep network?
- DenseNet is so called because the shortcut connections are dense.

Additionally, the fusion way of shortcut in DenseNet is concatenation, instead of addition in ResNet.



Shortcut in ResNet

Shortcut in DenseNet

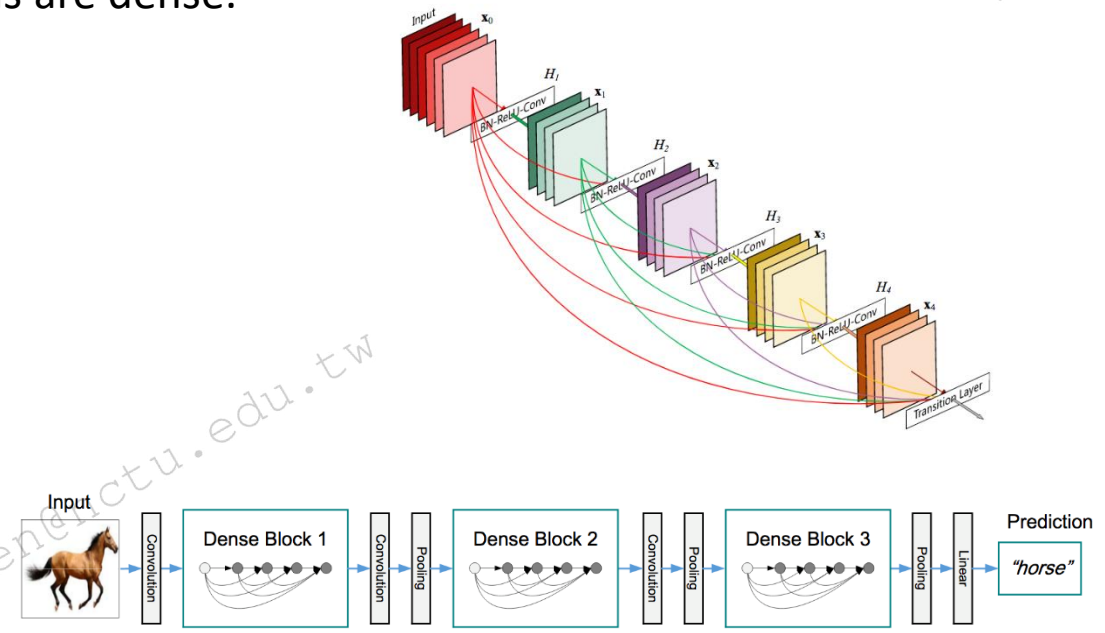


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

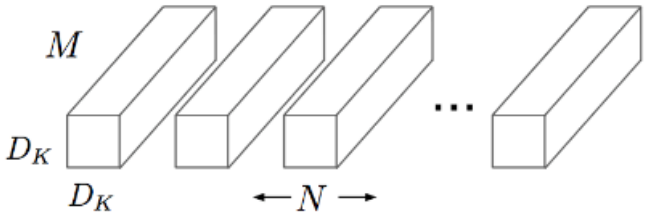
[1] Residual Networks Behave Like Ensembles of Relatively Shallow Networks

Qualcomm

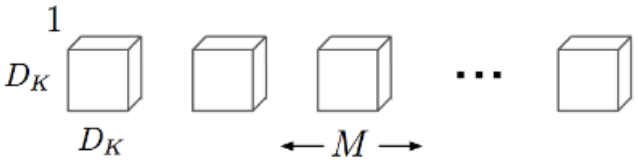
MobileNet

- Separable convolution is introduced to replace traditional convolution.
- traditional convolution kernel: [N, M, D, D]
- Separable convolutions
  - 1. Depthwise convolution: [M, 1, D, D]
  - 2. Pointwise convolution: [N, M, 1, 1]
- The output shape of separable convolutions are the same with that of traditional convolution

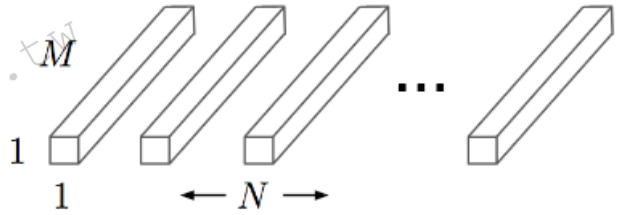
but with less parameters.



(a) Standard Convolution Filters



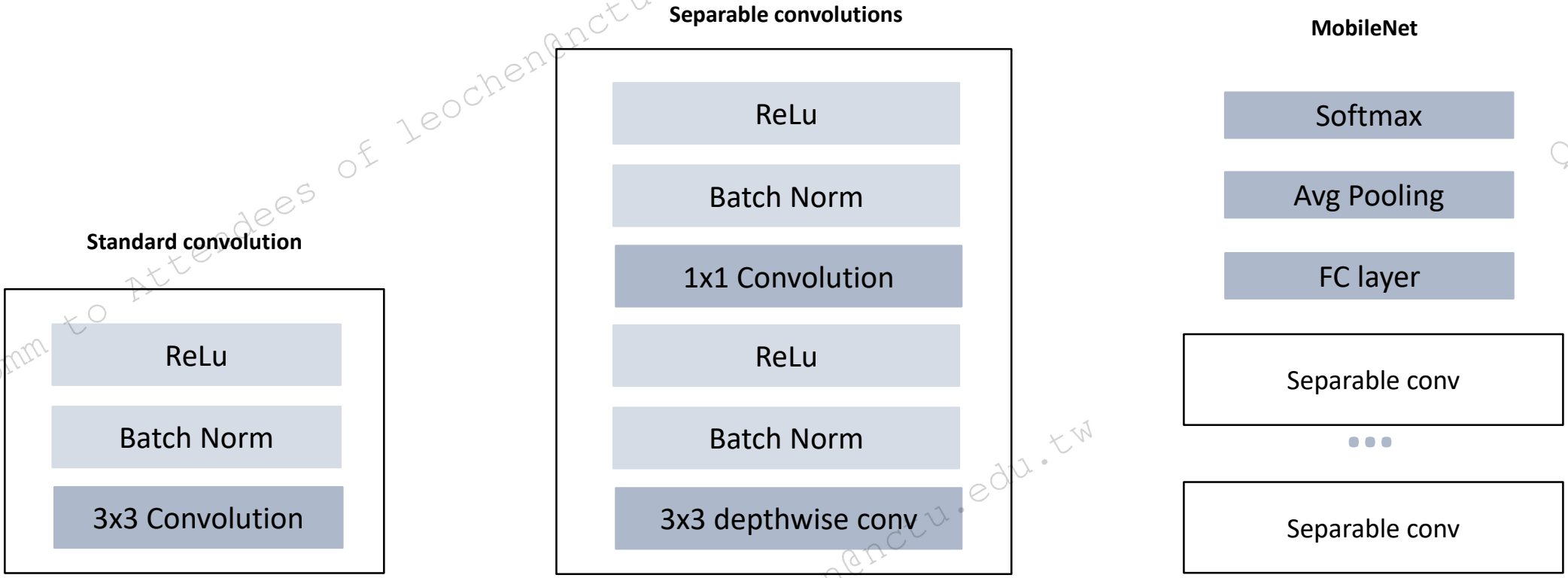
(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, axiv 1704.04861

MobileNet



Qualcomm

MobileNet

- MobileNet reaches equivalent performance but with less parameters comparing with other popular models.

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

- MobileNet is widely used in offline scenes without high computational performance unit, like mobile devices.

MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, axiv 1704.04861

Qualcomm

2.1 Building Deep Learning Model

◆ 2.1.2 Hyper-parameters & Tuning Tricks

- What is Hyper-parameters?
- Epoch, Batch size, Iteration
- Hyper-parameters Tuning Tricks

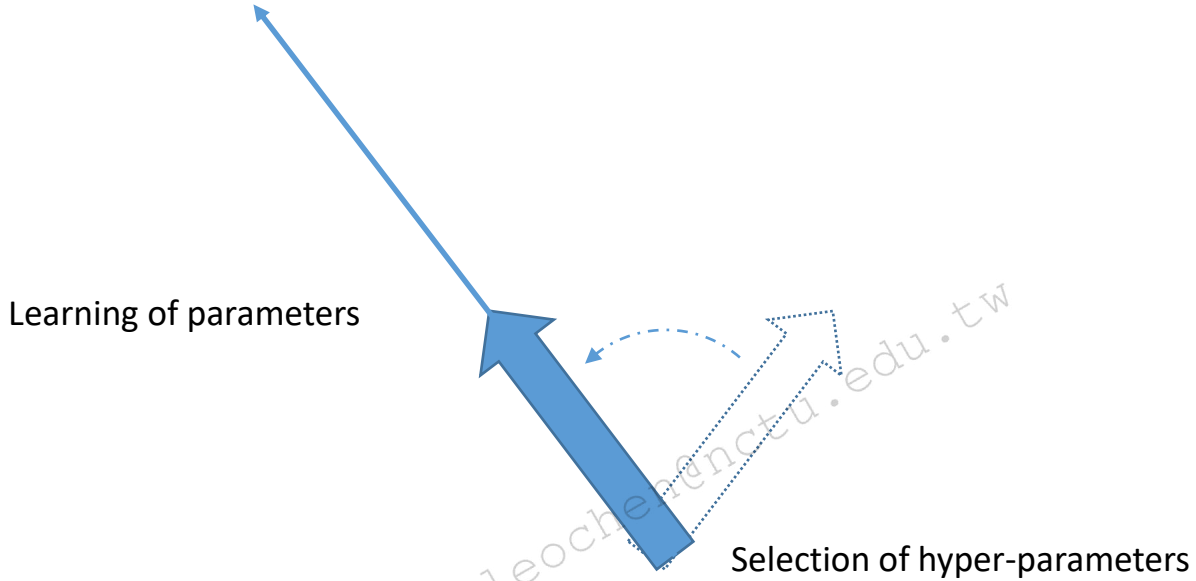
What is Hyper-parameters?

- **Parameters** are internal to the model and whose values are estimated from training data.
  - e.g. weighs and bias of FC layer, weight vector of Convolution layer.

- **Hyper-parameters** are external to the model and whose values cannot be estimated from data and only can be set manually.
  - Model hyper-parameters: Number of layers, filter size of pooling layers, type of activation function and optimizer.
  - Training hyper-parameters: Learning rate, batch size, epoch, iteration.

Neural network and hyper-parameters

- **Parameters**: Specifically refers to the parameters that are learned from training data **automatically**, rather than human manually setting.
- **Hyper-parameters** are a kind of special parameters about the architecture of the model, which are manually set before the training starts, and immediately impact the direction of parameters learning.



Qualcomm

Epoch, Batch size, Iteration

- Epoch: one forward pass and one backward pass of entire the training examples.
- Batch size: Total number of training examples present in a single batch.
- Iteration (a.k.a. Step): the number of batches needed to complete one epoch.

- For instance if you have 20,000 images, an epoch of 1, and a batch size of 64 then the epoch should contain [20,000 / 64] = 313 iterations/steps.

Hyper-parameters Tuning Tricks

- Learning rate is the speed that the parameters are updated. At the beginning of training, we need a larger learning rate. While a smaller learning rate is suitable for fine-tuning and the second half of training.

Learning rate decay is a useful tools in popular frameworks like Caffe, TensorFlow. It provides a mechanism to decay learning rate as processes.

- Epoch should be enough for convergence of loss function. It depends on specific tasks, its empirical value is usually larger than 1 and less than 50.
- Batch size is also an empirical value. [1][2] Its upper limits is related to your GPU memory, and its lower limits is usually 8 or 16.

[1] Don't decay the learning rate, increase the batch size, arxiv 1711.00489
[2] Revisiting Small Batch Training for Deep Neural Networks, 1804.07612

2.1 Building Deep Learning Model

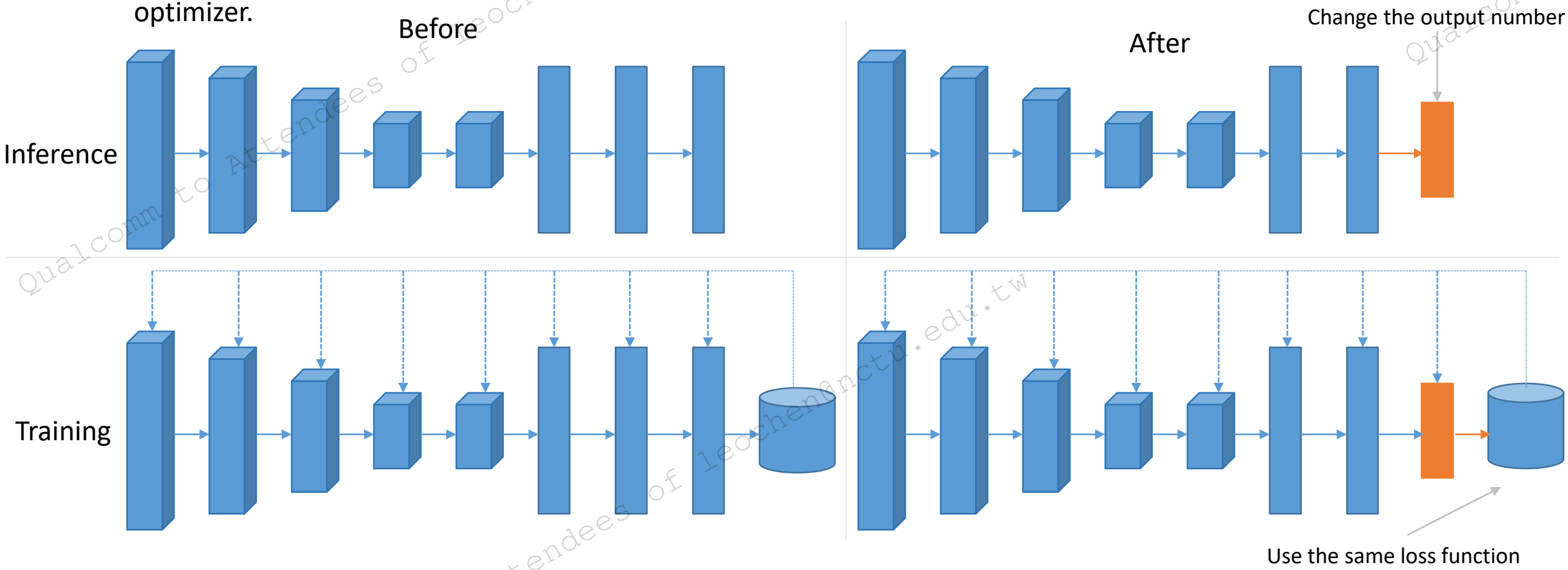
◆ 2.1.3 Fine-tune & Transfer Learning

Fine-tune

- Why we need Fine-tuning?
  - Classic models above are initialized randomly and trained on ImageNet (14 million examples and 1000 categories). Our datasets usually are much smaller than it. So it is likely to overfit if we train the model from scratch.
  - Even if out datasets have enough data, it's a quite hard task to initialized parameters and optimize parameters until it converge.
  - Fine-tuning is equivalent to another parameters initialization instead of random initialization. So it's suitable for you even your dataset is very large. It's at least not worse, so why not?
  - Fine-tuning provides a good way for us to stand on the shoulders of giants.
- What is Fine-tuning?
  - 1. Pre-train a deep net on a large-scale dataset, like ImageNet, COCO, Pascal VOC, etc. (By authors of classic models or someone who repeats the experiments)
  - 2. We start training with pre-trained models on a new task.

Types of Fine-tune

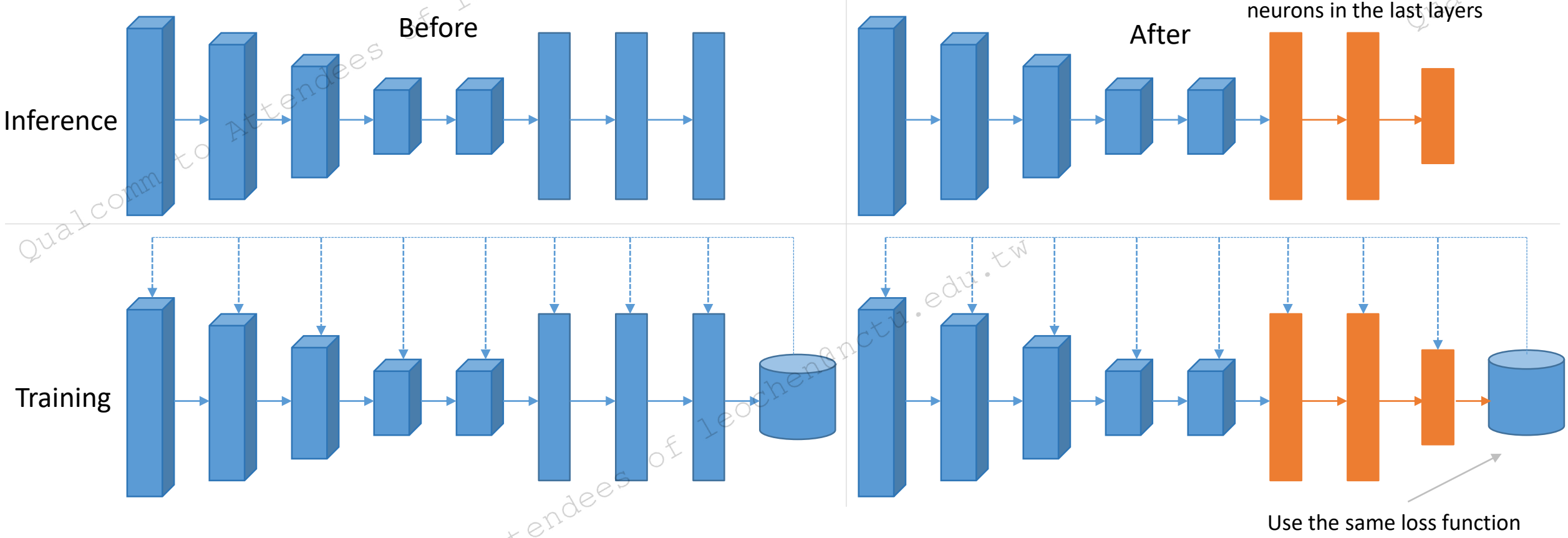
- 1. If our task is still image classification, we can just change the number of the last layer and keep the optimizer.



Qualcomm

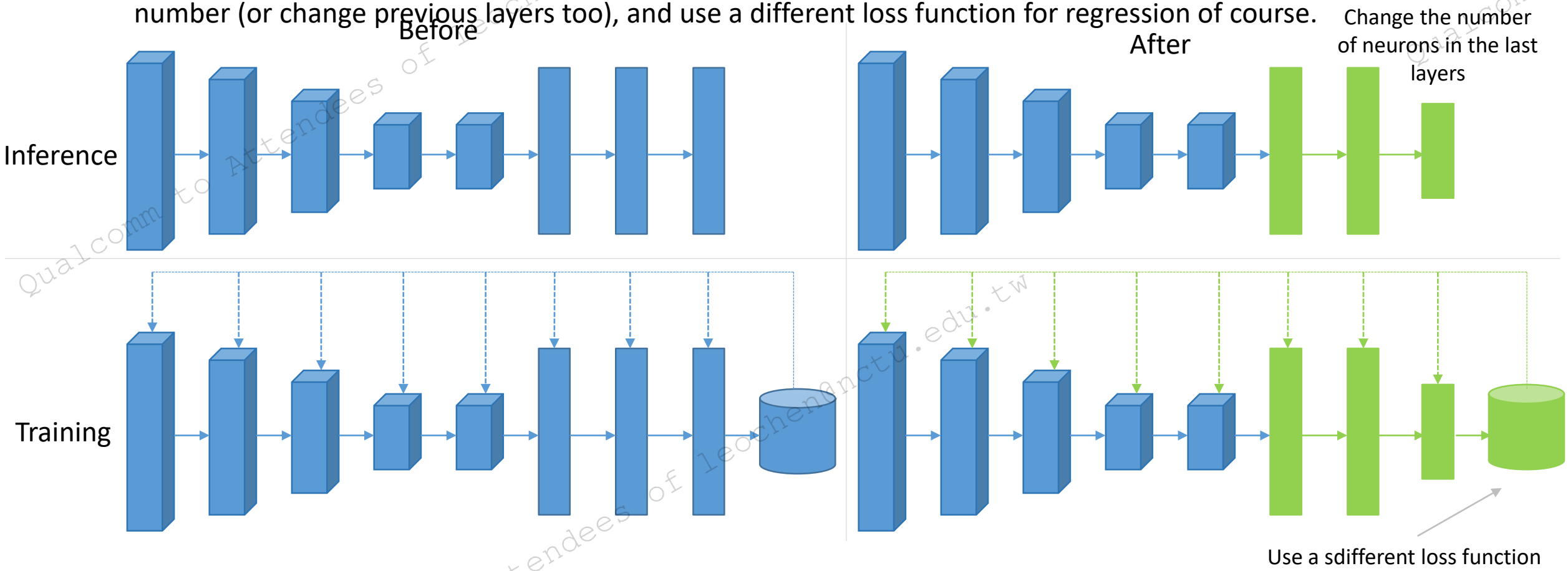
Types of Fine-tune

- 1. If our task is still image classification, we can just change the number of the last layer. We can also change all the number of neurons of classification layers (Optional, e.g. FC layers in VGG).



Types of Fine-tune

- 2. If our task is image regression (e.g. facial landmark detection), we usually have to change the output number (or change previous layers too), and use a different loss function for regression of course.

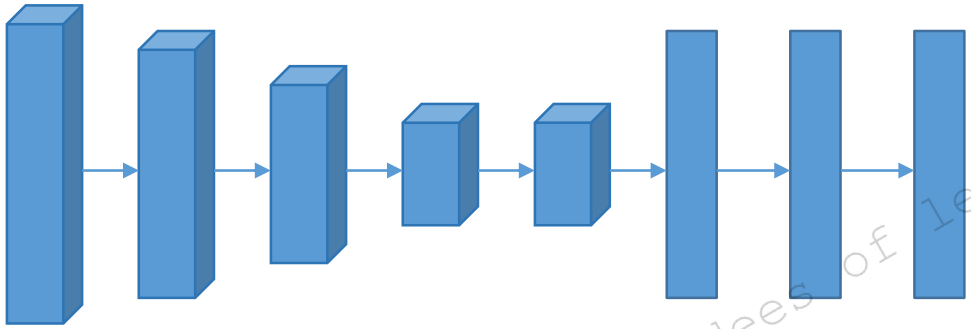


Qualcomm

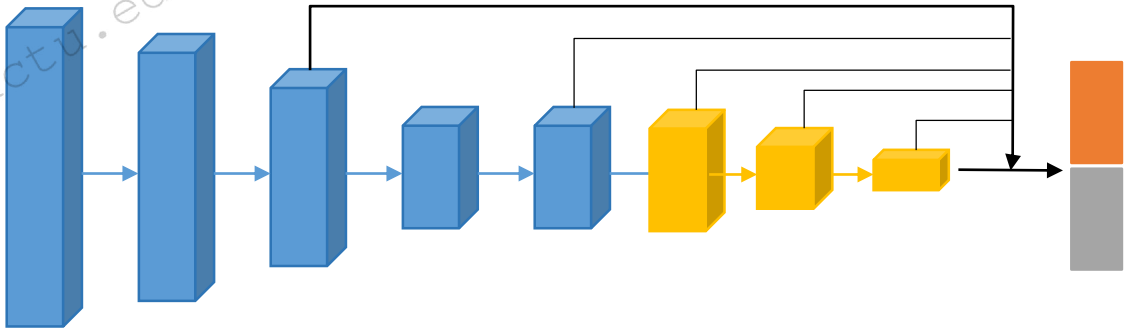
Types of Fine-tune

- 3. If our task is a more complex task (e.g. object detection), the model is corresponding more complicated as well. Classic models are generally used as a part of the whole model to extract image features.

VGG



SSD-VGG

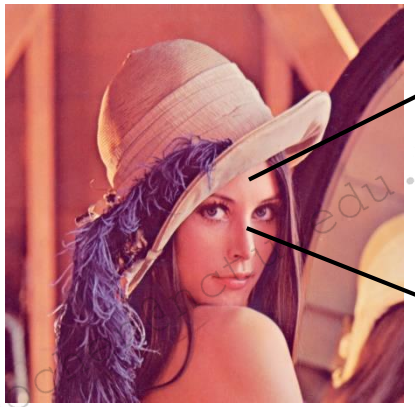


Data pre-process & augmentation

- RGB Channels, Grayscale
  - Each RGB pixel take 3 bytes to storage (raw image, 1 bytes = 8 bit = 2^8 = 256)
  - Conversion: Gray = R*0.299 + G*0.587 + B*0.114

Gray

192	194	196	187
193	195	198	200
196	194	197	198
195	198	196	194



R

216	218	220	211
217			
220			
219			

G

186	188	190	181
187			
190			
189			

B

156	158	160	151
157	159		164
160	158	161	162
159	162	160	158

Qualcomm

2.1 Building Deep Learning Model

◆ 2.1.4 Data pre-process & Data augmentation

Data pre-process & augmentation

- RGB or YUV?
  - YUV is another useful color space. It contains one luma component (Y′) and two chrominance (UV) components.
  - YUV and RGB can transform to each other. We use RGB here.
- RGB or BGR?
  - OpenCV and Caffe use BGR as default.
- NCHW or NHWC?
  - Caffe use NCHW while TensorFlow use NHWC as default.
  - NHWC(channels_last) is faster in CPU while NCHW(channels_first) is faster in GPU.

Data pre-process & augmentation

- We need learn some basic image operations for image IO, preprocess and data augmentation.
- OpenCV is a widely-used, open source library in computer vision.
- Python interface:
  - cv.imread
  - cv.imwrite
  - cv.resize

Data pre-process & augmentation

- TensorFlow also provide a module named tf.image for image processing and decode ops. It provides more useful high-level functions than OpenCV.
- Here are some example ops:
  - adjust_brightness(...): Adjust the brightness of RGB or Grayscale images.
  - crop_and_resize(...): Extracts crops from the input image tensor and resizes them.
  - flip_up_down(...): Flip an image vertically (upside down).
  - random_crop(...): Randomly crops a tensor to a given size.
  - …

2.1 Building Deep Learning Model

◆ 2.1.5 Hands-on - Keras_MNIST

Hands-on - Keras_MNIST

- The MNIST database is a large database of **handwritten digits** that is commonly used for training various image processing systems.

- It contains 60,000 training images and 10,000 testing images. Each image with a resolution of 28*28 is grayscale.

- It is a good database for beginners who want to try machine learning techniques and pattern recognition methods while spending minimal efforts on preprocessing and formatting, because it is well-preprocessed.

Thank You