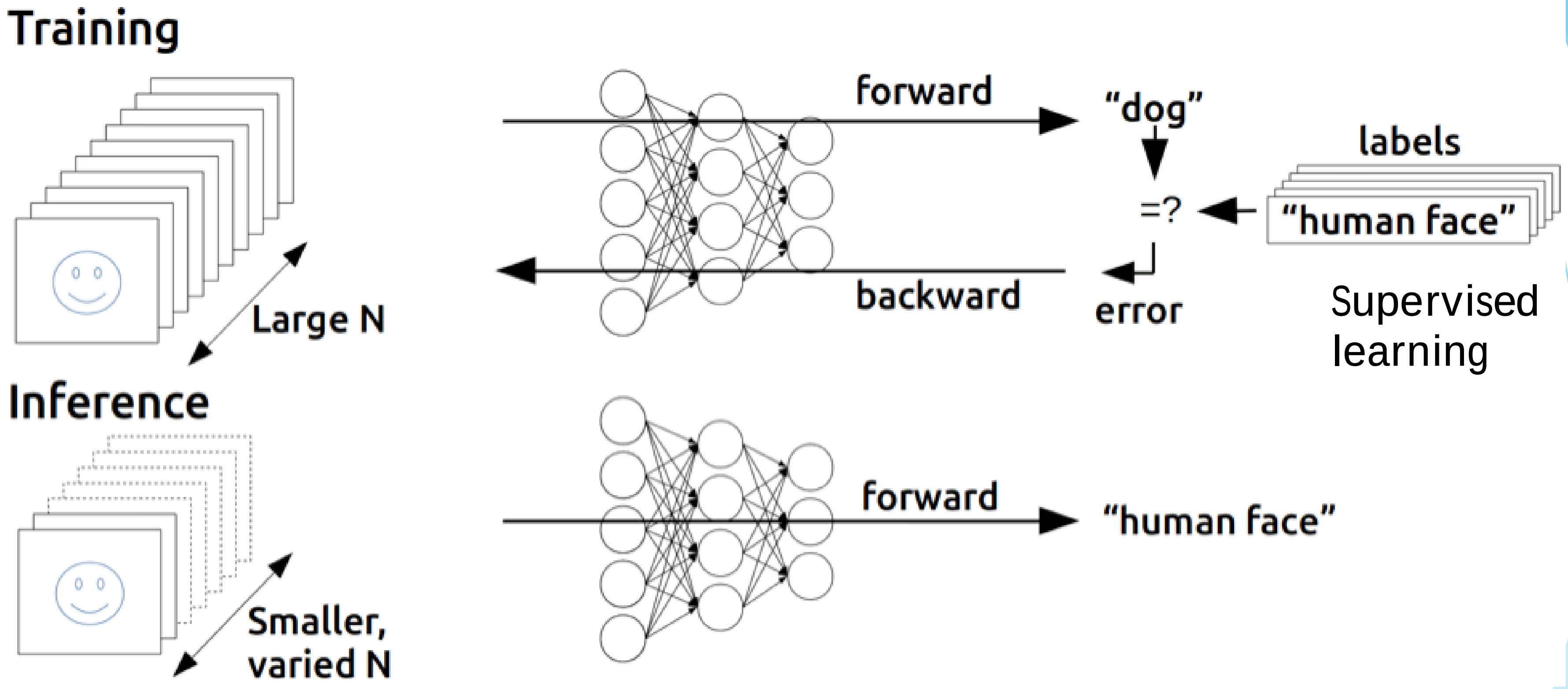
Lecture 1

AI-on-Chip System Overview

Chung-Ho Chen

NCKU EE

AI Training Versus Inference

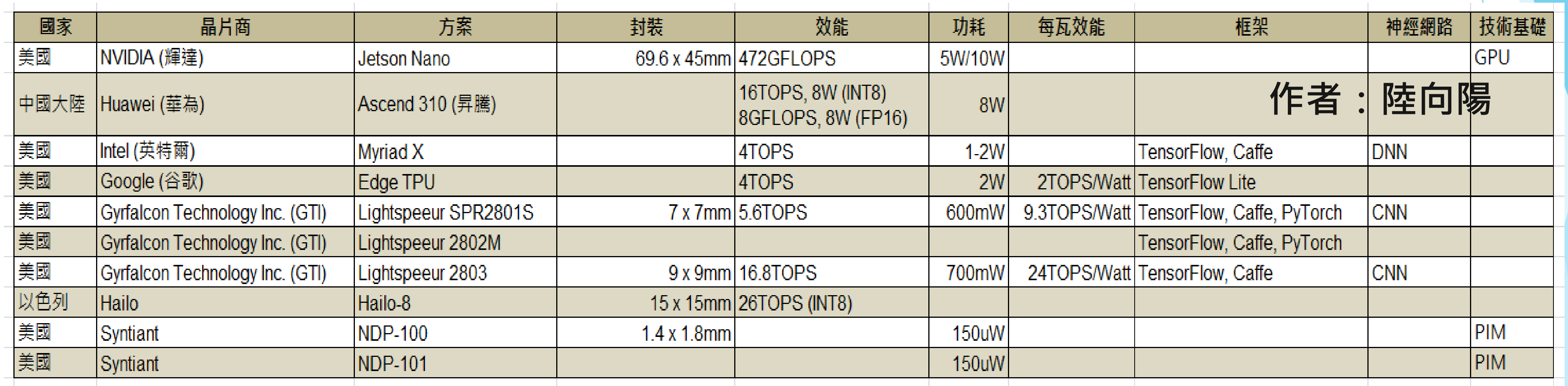


In **training**, many inputs, often in large batches, are used to train a deep neural network.
In **inference**, the **trained network** is used to discover information within **new inputs** that are fed through the network in smaller batches.

https://devblogs.nvidia.com/inference-next-step-gpu-accelerated-deep-learning/

Edge AI v.s. Cloud AI

Example：Edge AI Devices, 2019



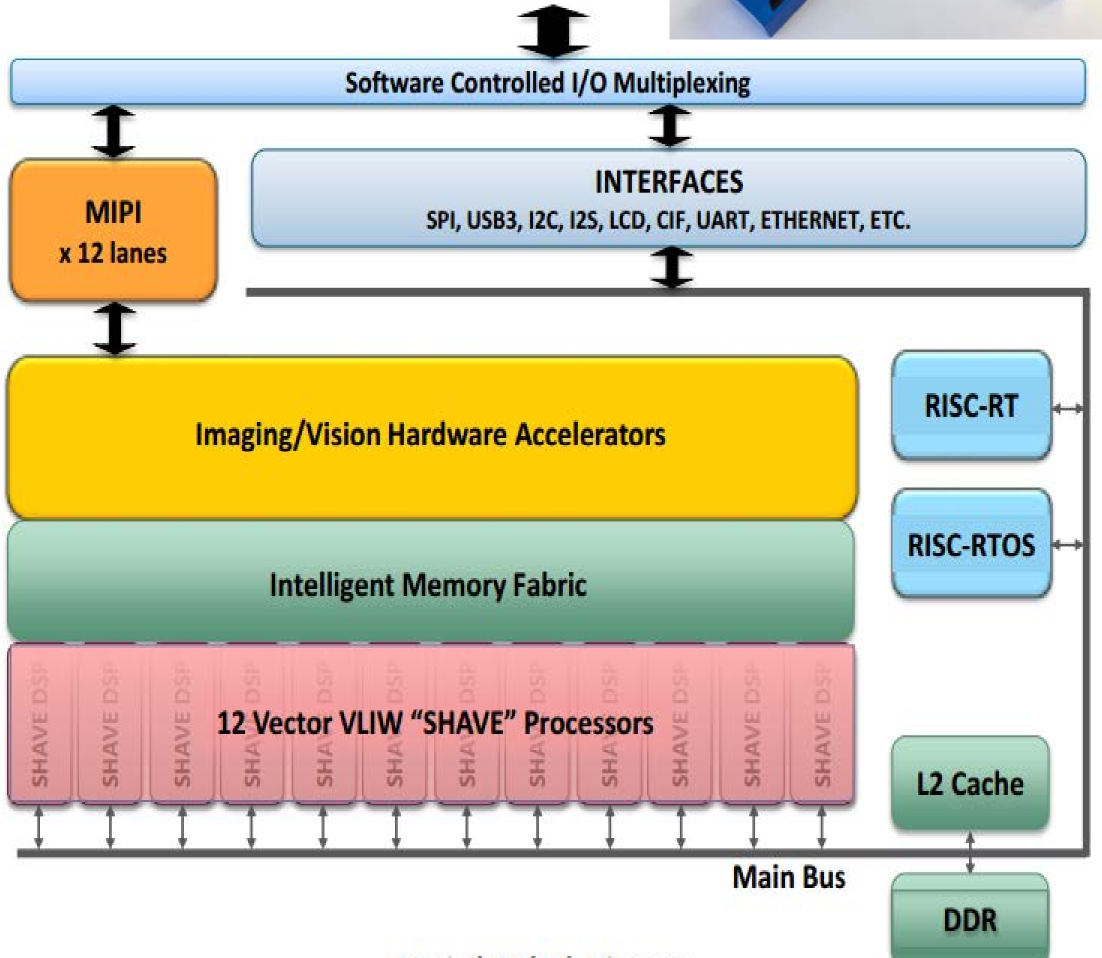
國家	晶片商	方案	封裝	效能	功耗	每瓦效能	框架	神經網路	技術基礎
美國	NVIDIA (輝達)	Jetson Nano	69.6 x 45mm	472GFLOPS	5W/10W				GPU
中國大陸	Huawei (華為)	Ascend 310 (昇騰)		16TOPS, 8W (INT8) 8GFLOPS, 8W (FP16)	8W		作者：陸向陽		
美國	Intel (英特爾)	Myriad X		4TOPS	1-2W		TensorFlow, Caffe	DNN	
美國	Google (谷歌)	Edge TPU		4TOPS	2W	2TOPS/Watt	TensorFlow Lite		
美國	Gyrfalcon Technology Inc. (GTI)	Lightspeeur SPR2801S	7 x 7mm	5.6TOPS	600mW	9.3TOPS/Watt	TensorFlow, Caffe, PyTorch	CNN	
美國	Gyrfalcon Technology Inc. (GTI)	Lightspeeur 2802M					TensorFlow, Caffe, PyTorch		
美國	Gyrfalcon Technology Inc. (GTI)	Lightspeeur 2803	9 x 9mm	16.8TOPS	700mW	24TOPS/Watt	TensorFlow, Caffe	CNN	
以色列	Hailo	Hailo-8	15 x 15mm	26TOPS (INT8)					
美國	Syntiant	NDP-100	1.4 x 1.8mm		150uW				PIM
美國	Syntiant	NDP-101			150uW				PIM

- Cloud AI
  - High performance GPU farm, FPGAs, Cluster Computing, …..
  - Microsoft Project Brainwave, CPUs + FPGAs for Microsoft Azure and Azure Databox Edge
  - Google AI, GPUs + TPUs (Massive MAC hardware)

Know the scope of developing an AI-on-Chip System

Intel Movidius Compute Stick USB AI accelerator

- Myriad X
  - 1T DNN operations/s, 2 Watts
  - 12 Streaming Hybrid Architecture Vector Engine Core
    - C/C++ with **intrinsic support**
  - GCC toolchains for RISC CPUs (ELF/DWARF linker)
  - Full chip simulator, bit/cycle accurate
  - Debugger support for SHAVE and CPU cores
  - System/Function profiler
  - Eclipse IDE

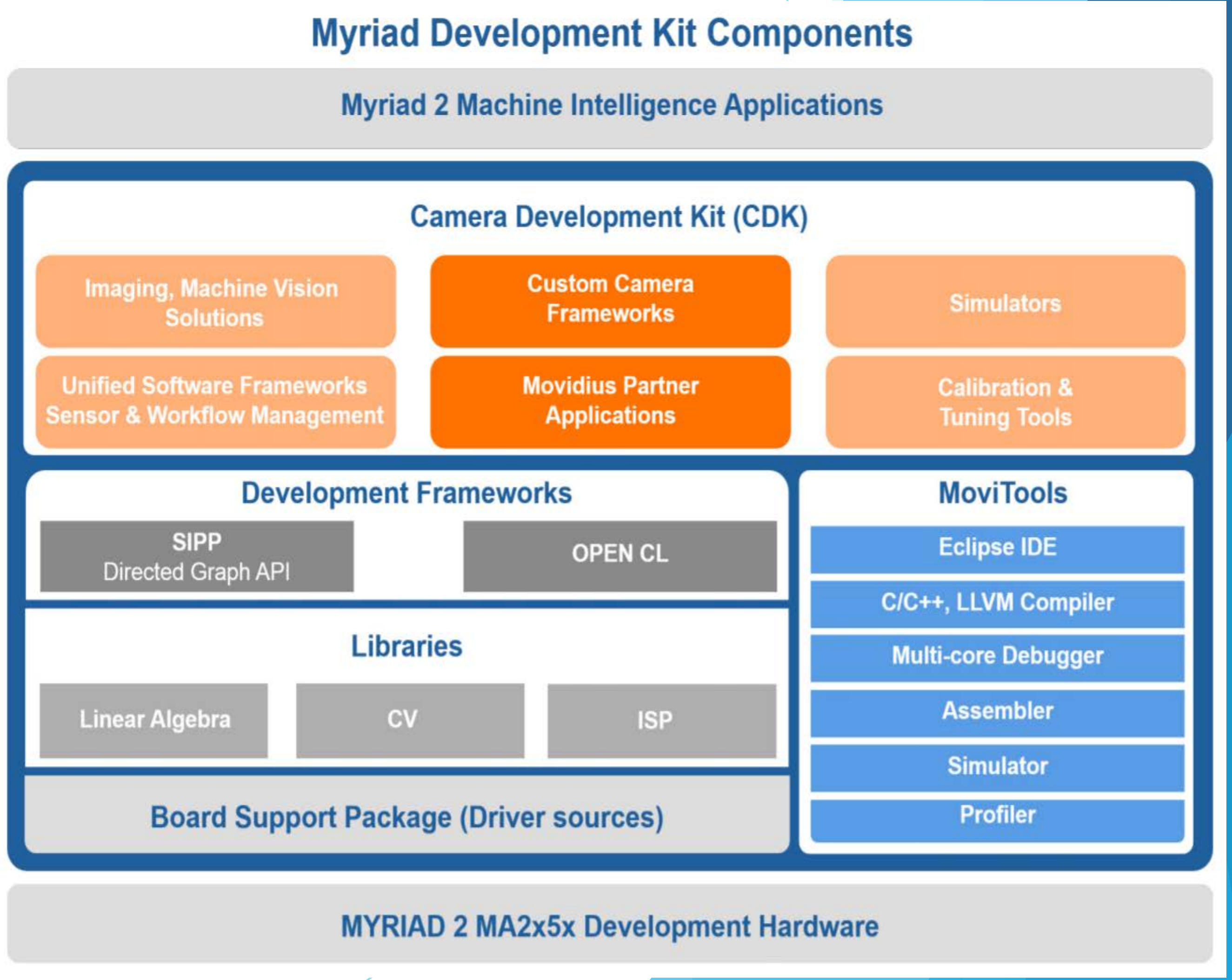


Software Controlled I/O Multiplexing
MIPI
x 12 lanes
INTERFACES
SPI, USB3, I2C, I2S, LCD, CIF, UART, ETHERNET, ETC.
Imaging/Vision Hardware Accelerators
RISC-RT
RISC-RTOS
Intelligent Memory Fabric
12 Vector VLIW "SHAVE" Processors
SHAVE DSP
L2 Cache
Main Bus
DDR

Myriad 2 Block Diagram

Intel Movidius Compute Stick USB AI accelerator

- Myriad X Development Frameworks (software)
  - OpenCL : Open computing language
  - SIPP: Proprietary, Streaming Image Processing Pipeline
  - ISP: Image Signal Processing
  - OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library, 2500+ algorithms. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS.
  - OpenCV: CUDA and OpenCL interfaces are being developed. 2019/11/05

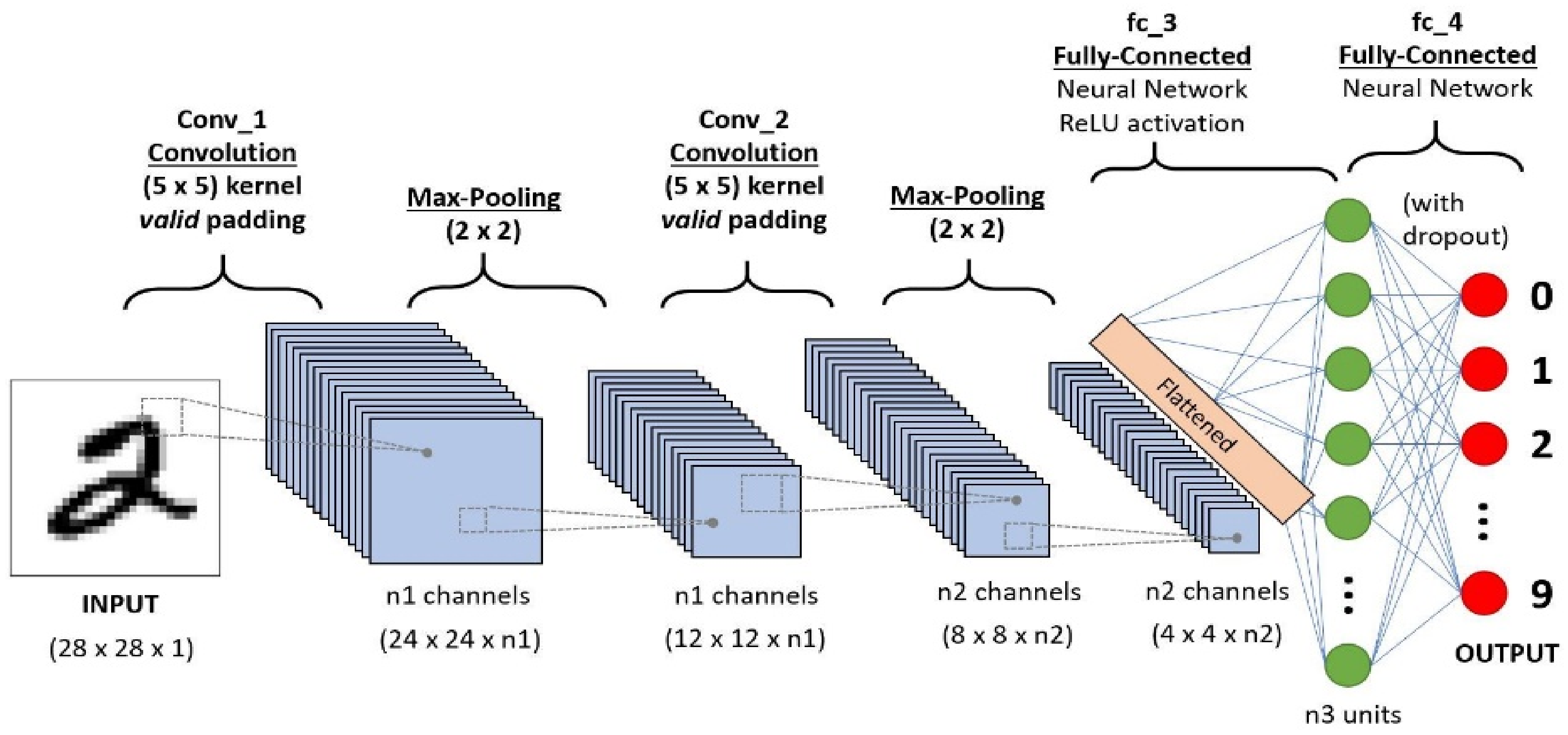

Myriad Development Kit Components
Myriad 2 Machine Intelligence Applications
Camera Development Kit (CDK)
Imaging, Machine Vision Solutions
Custom Camera Frameworks
Simulators
Unified Software Frameworks Sensor & Workflow Management
Movidius Partner Applications
Calibration & Tuning Tools
Development Frameworks
SIPP Directed Graph API
OPEN CL
Libraries
Linear Algebra
CV
ISP
Board Support Package (Driver sources)
MoviTools
Eclipse IDE
C/C++, LLVM Compiler
Multi-core Debugger
Assembler
Simulator
Profiler
MYRIAD 2 MA2x5x Development Hardware

Movidius Myriad 2: PPT

Technology inside Movidius Myriad 2

- Computer Architecture
  - Data parallelism execution model
  - Vector processor
  - ASIC accelerators
  - RISC processor and control firmware
  - Interconnection
  - Memory Hierarchy
- Software Development Kit, tools
  - Libraries, OpenCL lib (runtime)…
  - Hardware Abstraction layer (API)
  - Compiler
  - Simulator
  - Profiler

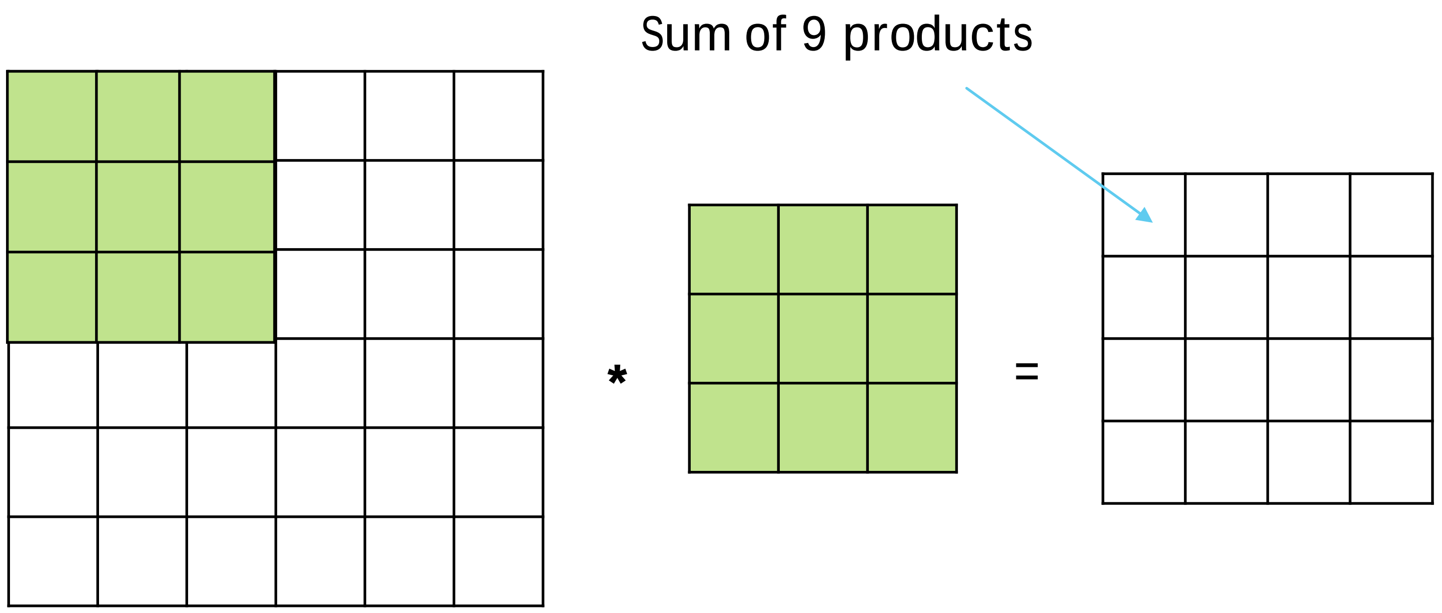
CNN: Convolution Neural Network



- A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Convolution Operation: Product Sum Multiply-Add (MAC)



No padding

1: Output size shrinking

2: Edge pixel contributes less to output, information is thrown away.

$$\longleftrightarrow \frac{n-f+2 \times p}{s}+1$$

6 * 6
n * n

3 * 3
f * f

4 * 4
(n – f + 1) * (n – f + 1)
(6 – 3 + 1) = 4

The dot product of two vectors $\mathbf{a} = [a_1, a_2, \ldots, a_n]$ and $\mathbf{b} = [b_1, b_2, \ldots, b_n]$ is defined as:

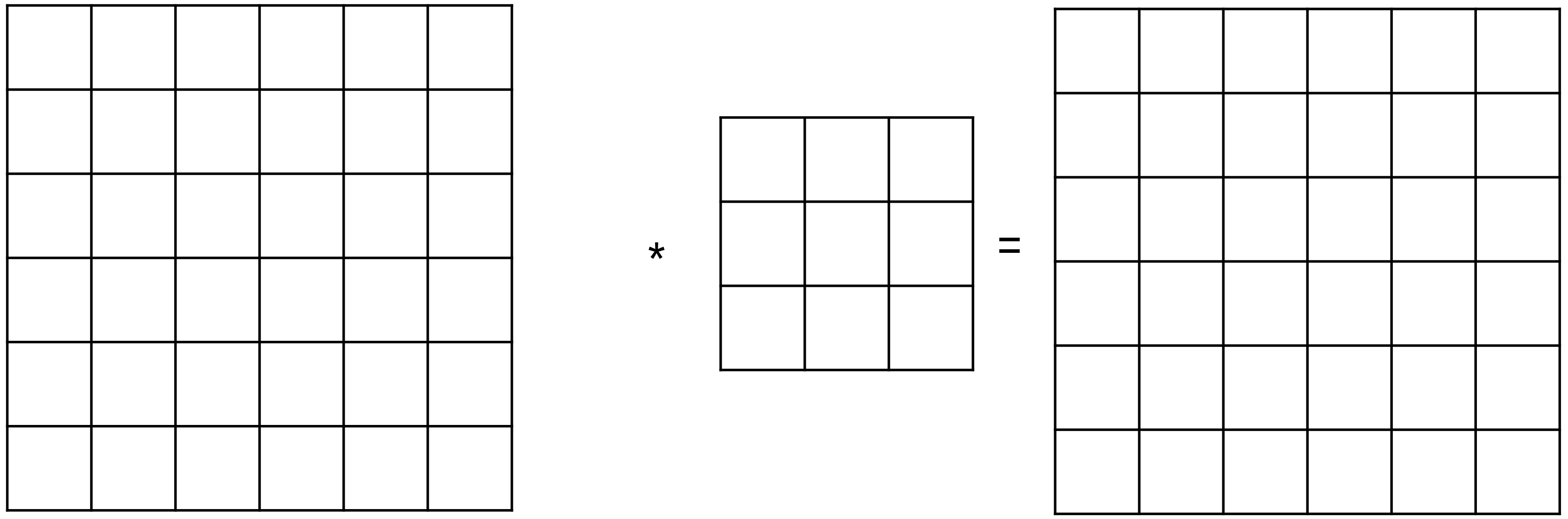
$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

Here n is the filter size (fxf) or convolution block size

Convolutional Neural Networks (Course 4 of the Deep Learning Specialization) by Andrew Ng

Padding Operation

- Shrinking output
- Throwing away a lot of the information



Padding

For the same output size,

Let output size
$n+2p-f + 1 = n$ (input size)

So, $p = (f-1)/2$

f is usually odd, 3x3, 5x5, 7x7...

6 * 6 → (Padding to 8 * 8)
n * n

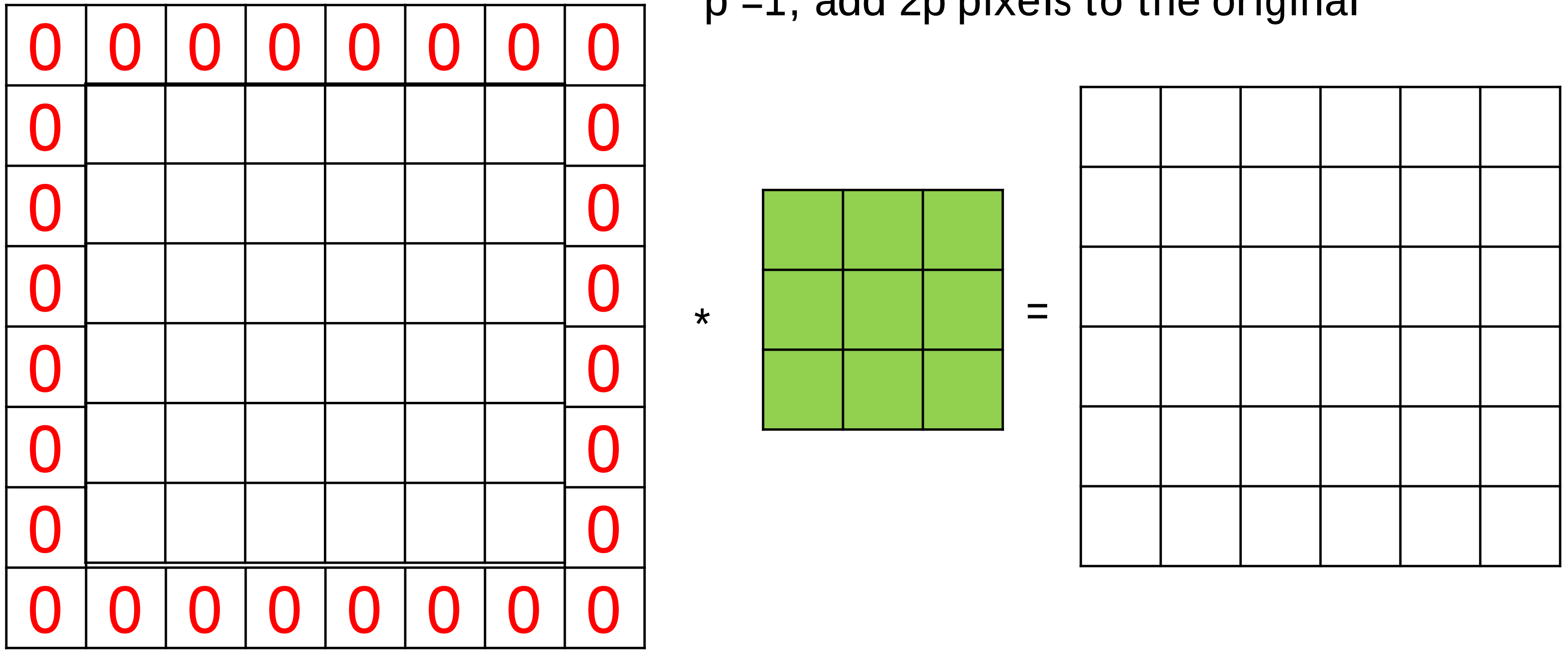
3 * 3
f * f

6 * 6
(n + 2p - f + 1) * (n + 2p - f + 1)

Fro the above example,
Padding = 1

Convolutional Neural Networks (Course 4 of the Deep Learning Specialization) by Andrew Ng

Padding Operation



p =1, add 2p pixels to the original

Padding

For the same output size,

Let output size
 n+2p-f + 1 = n (input size)

So, p = (f-1)/2

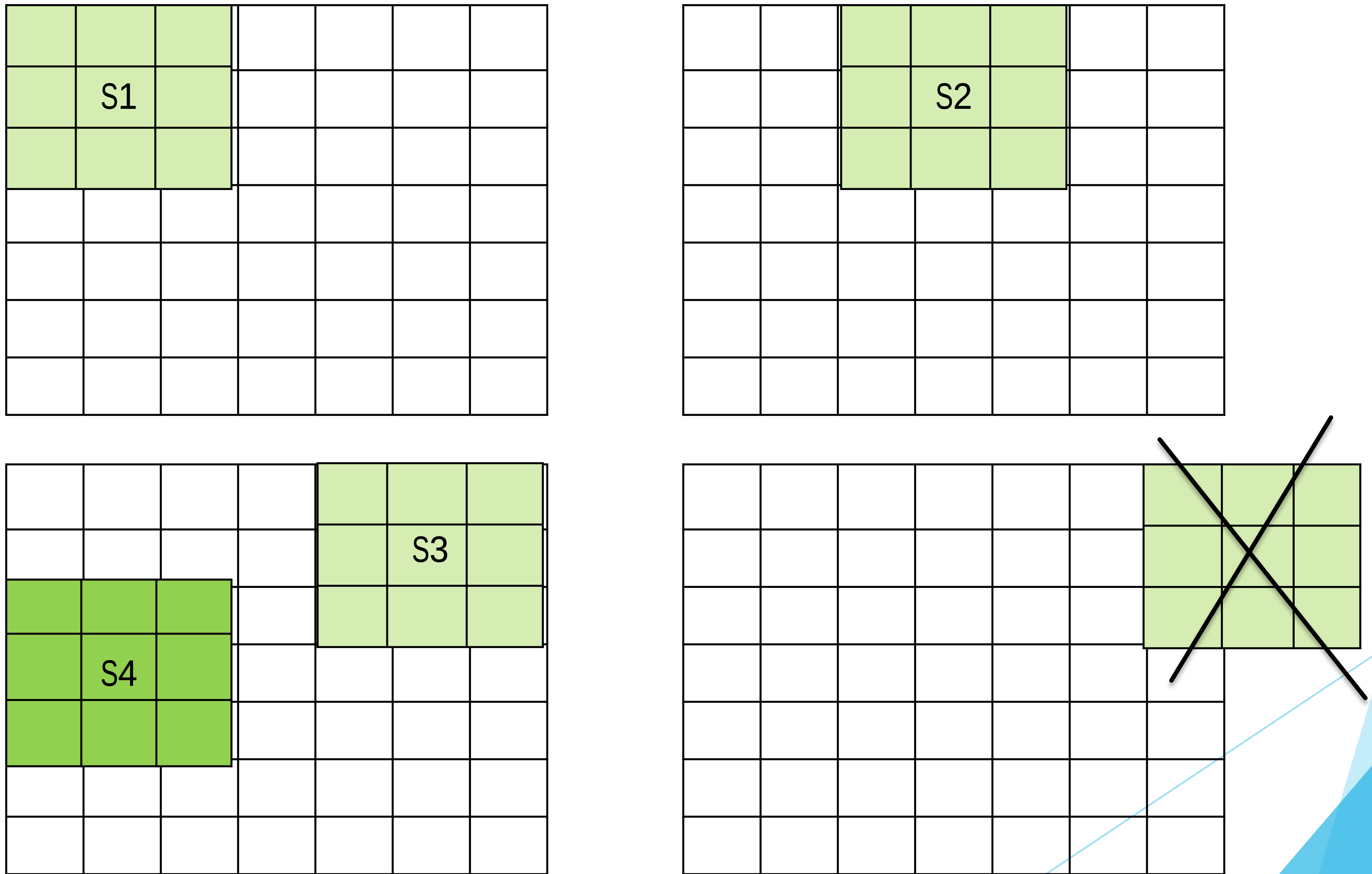
f is usually odd, 3x3, 5x5, 7x7…

6 * 6 → 8 * 8
n * n

3 * 3
f * f

6 * 6
(n + 2p - f + 1) * (n + 2p - f + 1)

Strided Convolution: S = 2



Strided Convolution

Output dimension

$$\left(\left\lfloor \frac{(n + 2p - f)}{s} \right\rfloor + 1\right) * \left(\left\lfloor \frac{(n + 2p - f)}{s} \right\rfloor + 1\right)$$

$$\left\lfloor \frac{(7 + 0 - 3)}{2} \right\rfloor + 1 = 3$$

floor operation

2	5	6	8	2	3	1
8	4	6	2	1	5	4
3	6	2	5	8	9	2
3	2	5	7	8	6	3
7	9	2	5	5	5	4
6	3	2	5	4	7	8
1	0	2	3	0	2	5

*

2	1	5
-6	5	1
-1	2	-1

=

24	5	43
28	57	21
11	55	53

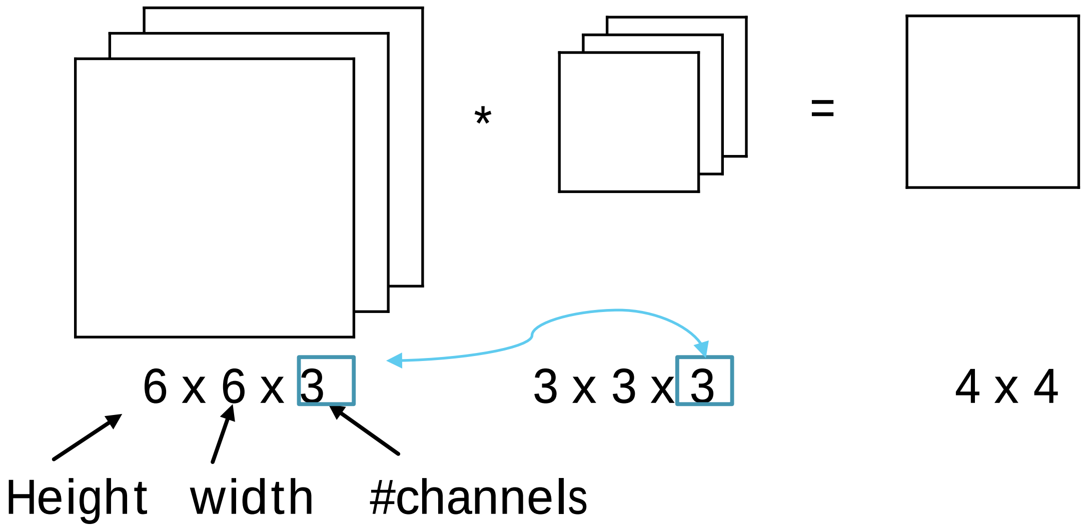
7 * 7
n * n

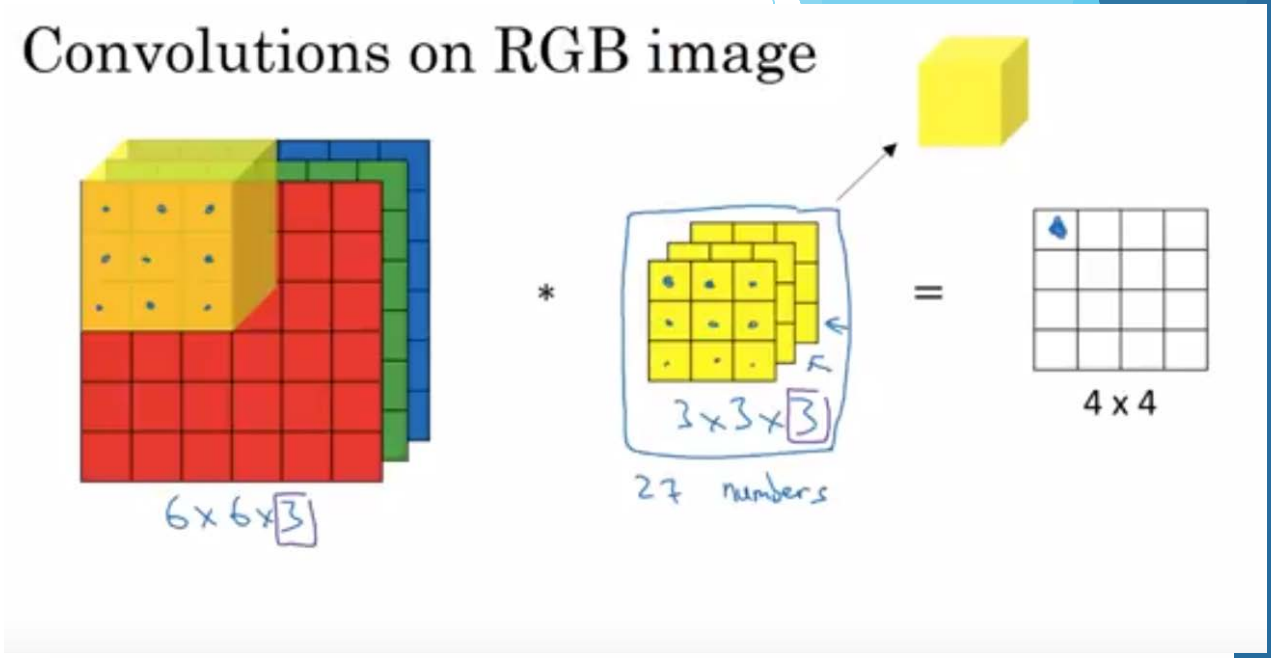
3 * 3
f * f

Stride = 2
Padding = 0

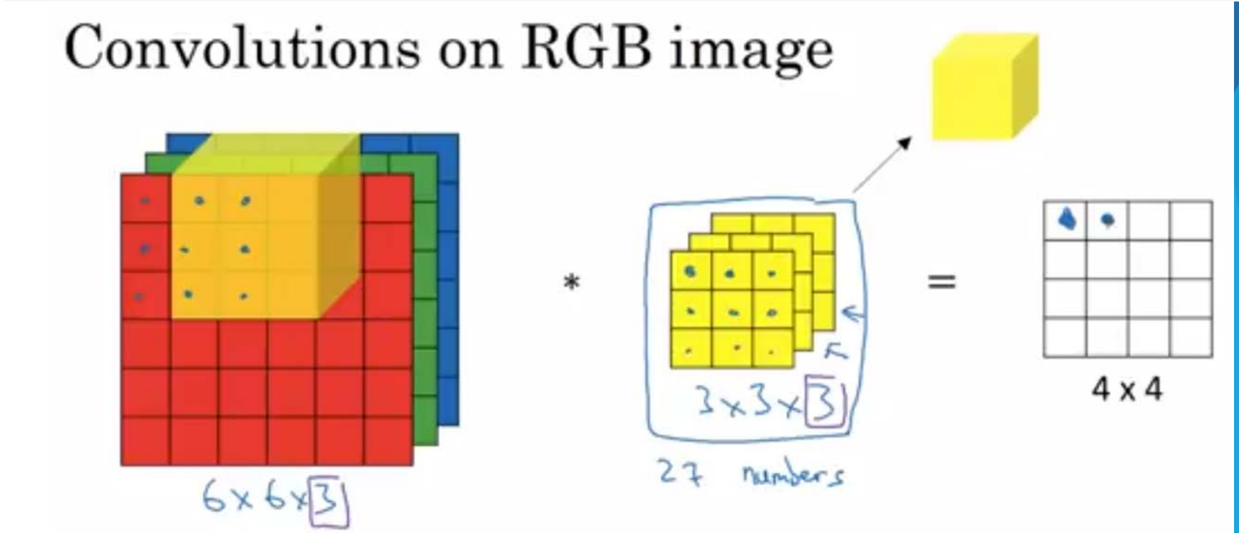
Convolution with volume

Convolution on RBG images





Here, have the same number of filters (depth) as the input channels.

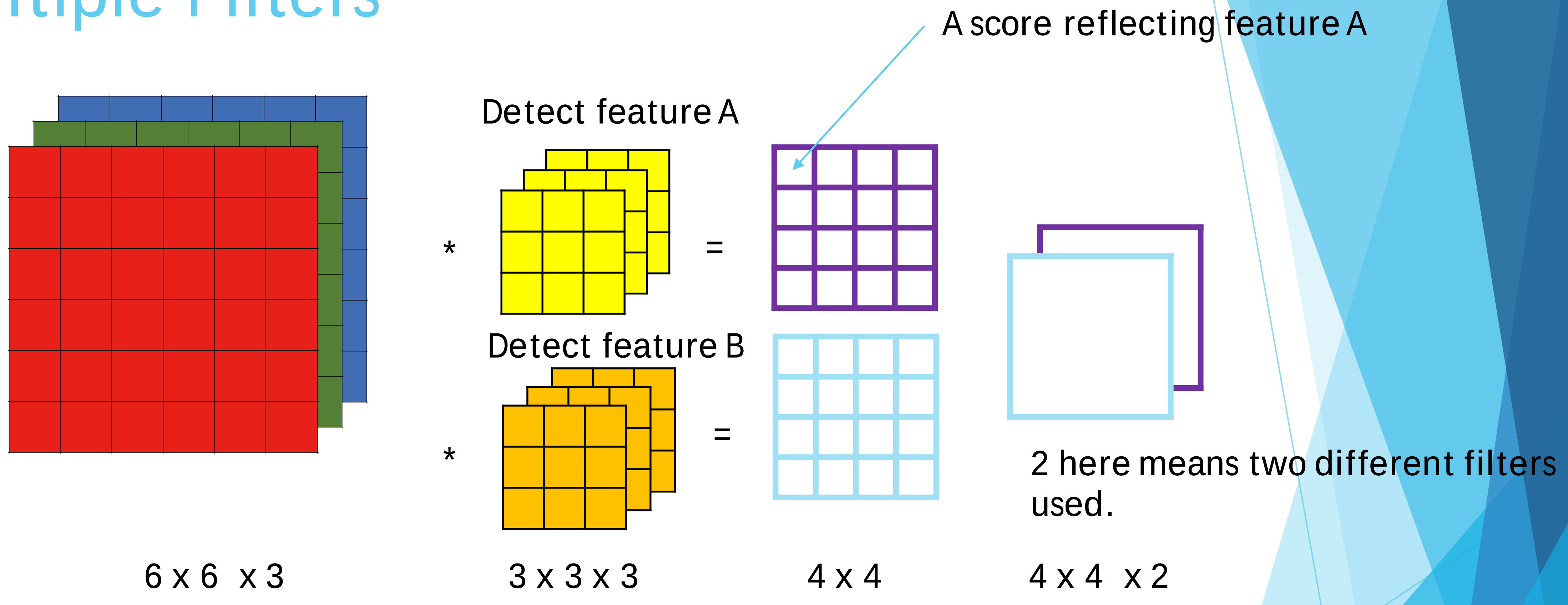


Think of this:
* the value in a filter for each of the respective channel. (Within a filter of different channels)

Angrew Ng

https://www.youtube.com/watch?v=KTB_OFoAQcc&list=PLkDaE6sCZn6GI29AoE31iwdVwSG-KnDzF&index=6

Multiple Filters



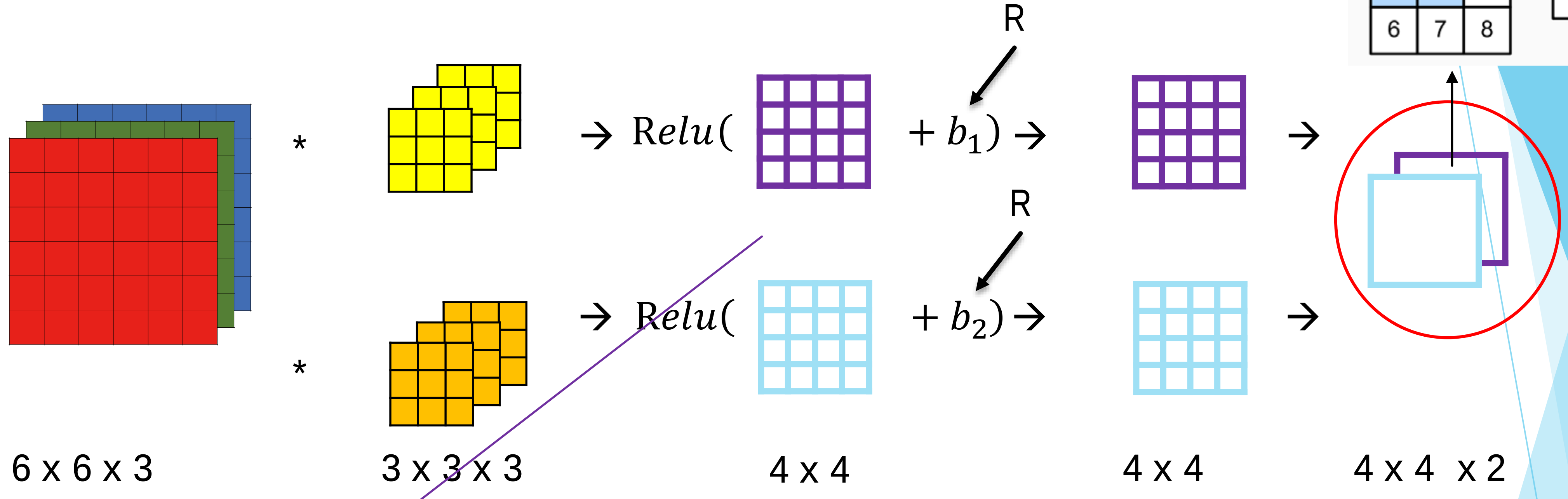
A score reflecting feature A
Detect feature A
Detect feature B
*
=
*
=
2 here means two different filters used.
6 x 6 x 3
3 x 3 x 3
4 x 4
4 x 4 x 2

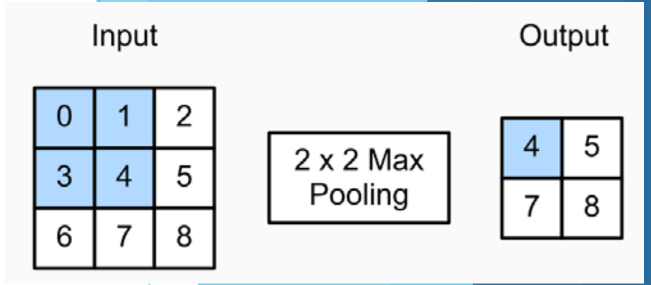
Different filters (yellow v.s. blown) used means to detect different features, like vertical edges or horizontal edges.

n x n x nc * f x f x nc -> (n-f+1) x (n-f+1) x nc’

nc: number of channels
nc’: number of filters used

Operation of a Layer



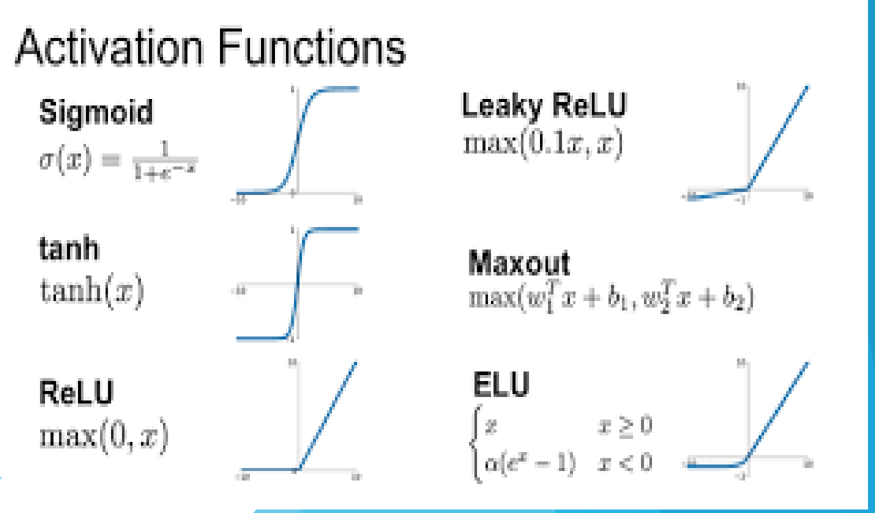


$$z^{(1)} = \boxed{W^{(1)} a^{(0)}} + b^{(1)}$$

$$a^{(1)} = g(z^{(1)})$$

Layer 1, take layer 0 as input

Convolution is a linear operation
Activation function is nonlinear
Neuron nature is non-linear



Wrap Up on Convolution

- A convolution layer is composed by a configurable number of filters, where each filter is a HxWxC matrix of trainable weights.
- A convolution operation is performed between the image and each filter, producing as output a new image (tensor in DL) with height and weight determined by the input image, stride, and padding (the output height and weight are inversely proportional to the stride) and as many channels as the number of filters.

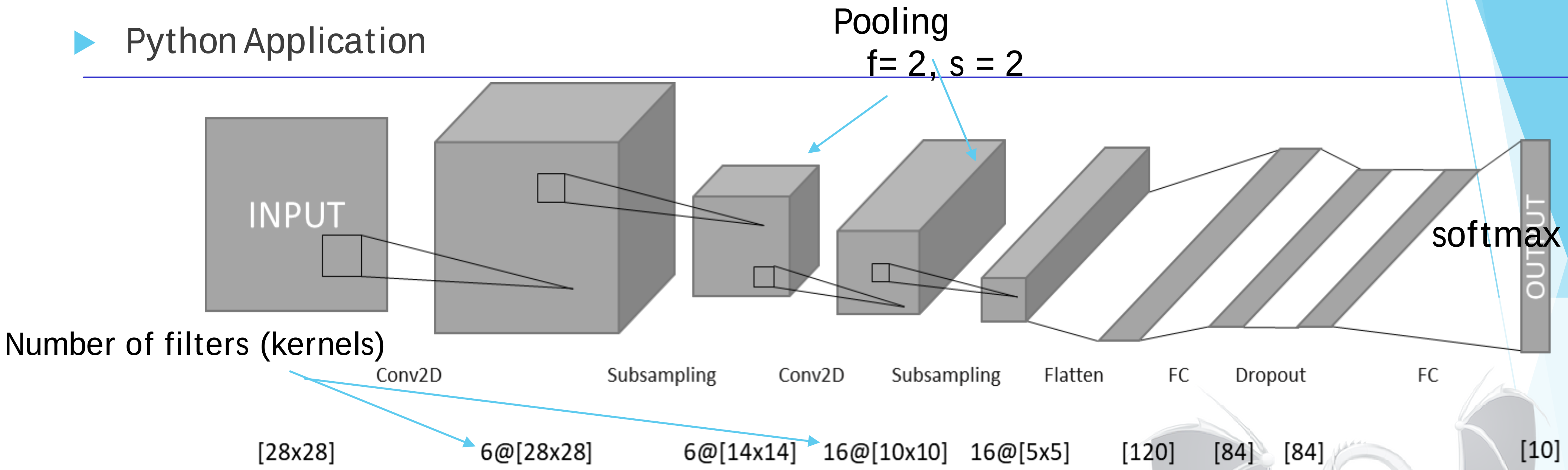
Output dimension in a channel

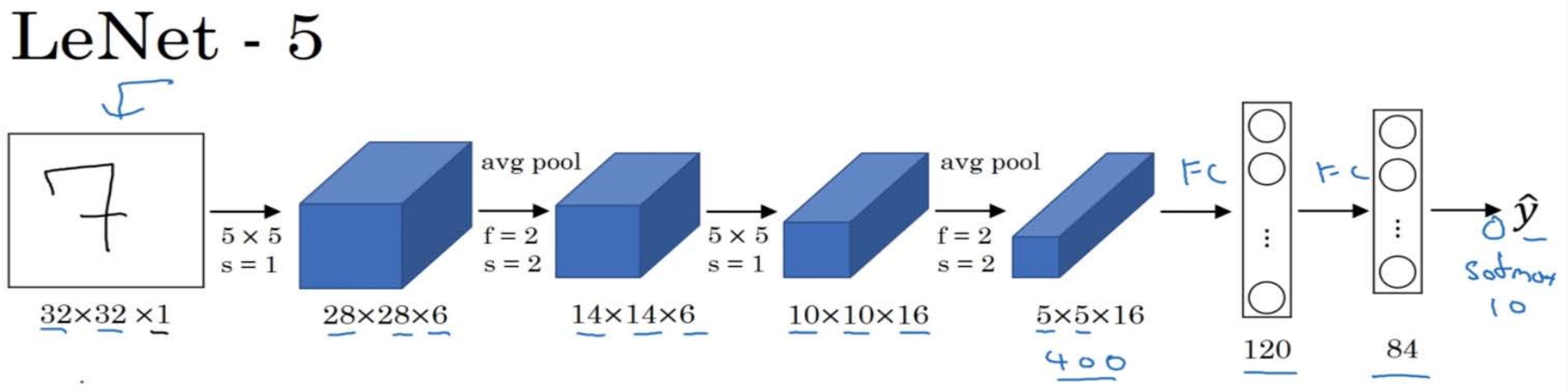
$$\left(\left\lfloor\frac{(n + 2p - f)}{s}\right\rfloor + 1\right) * \left(\left\lfloor\frac{(n + 2p - f)}{s}\right\rfloor + 1\right)$$

- Every value in the tensor is then fed through an activation function to introduce a nonlinearity.

A Simple Convolution Network: LeNet-5

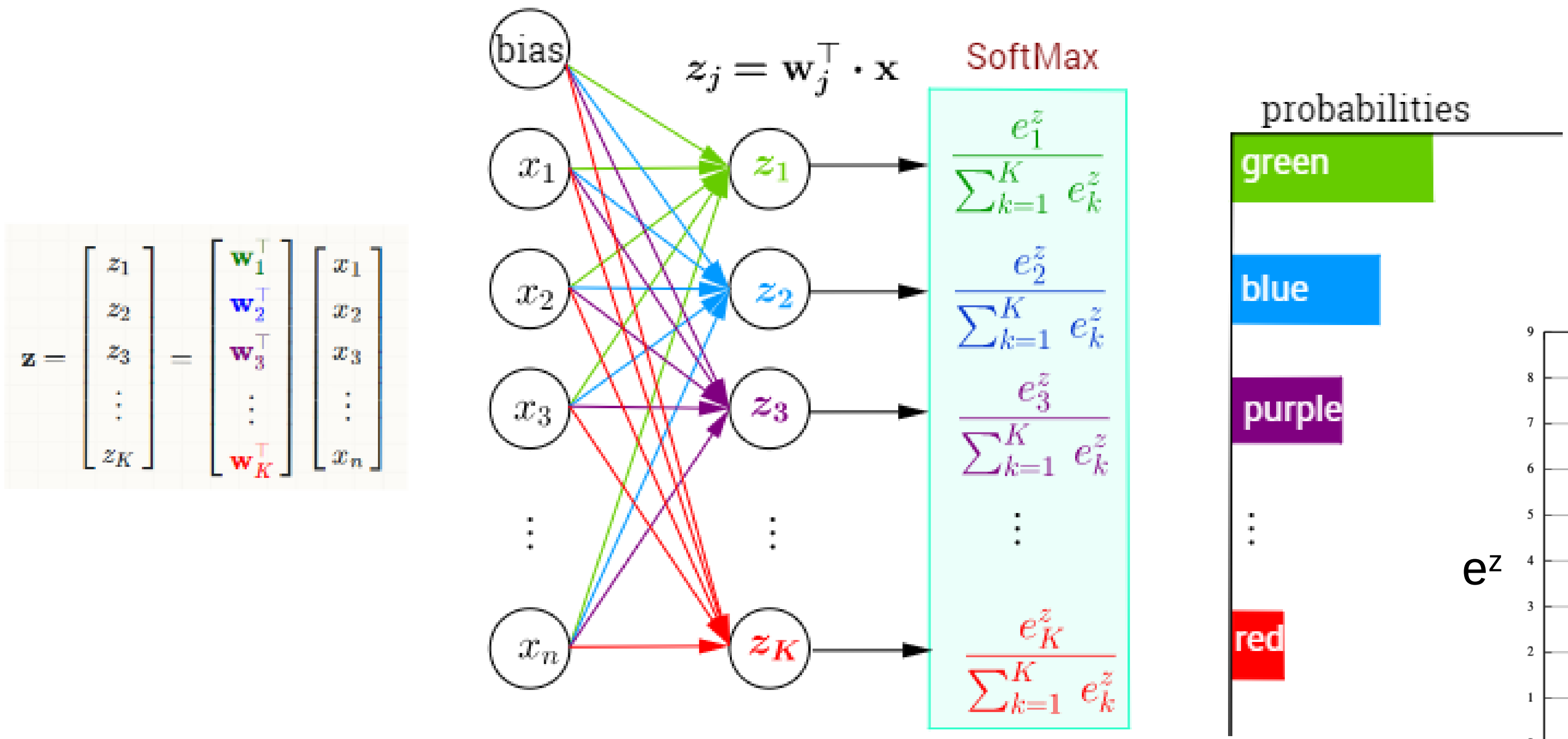
- Python Application



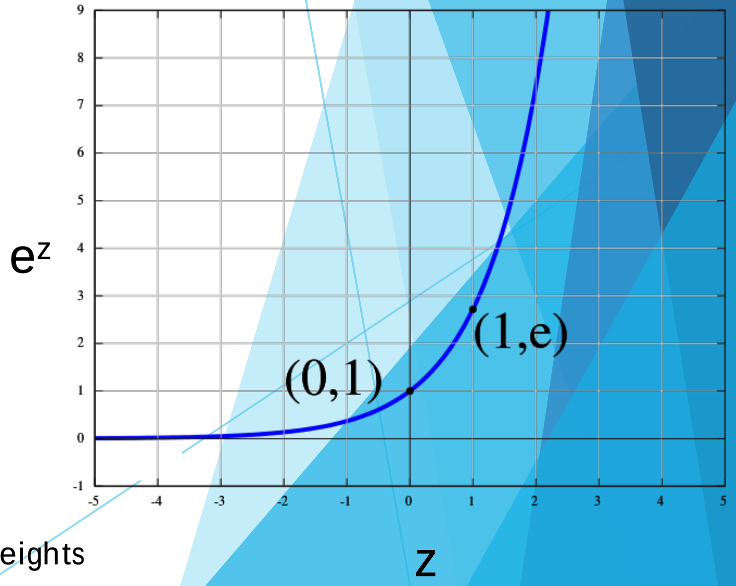


Softmax Function

Multi-Class Classification with NN and SoftMax Function



$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$
bias
x_1
x_2
x_3
x_n
$z_j = \mathbf{w}_j^\top \cdot \mathbf{x}$
z_1
z_2
z_3
z_K
SoftMax
$\frac{e^{z}_1}{\sum_{k=1}^{K} e^{z}_k}$
$\frac{e^{z}_2}{\sum_{k=1}^{K} e^{z}_k}$
$\frac{e^{z}_3}{\sum_{k=1}^{K} e^{z}_k}$
$\frac{e^{z}_K}{\sum_{k=1}^{K} e^{z}_k}$
probabilities
green
blue
purple
red



https://stats.stackexchange.com/questions/265905/derivative-of-softmax-with-respect-to-weights

Why using natural exponential function for output z ?

Online Resources for Basics and Terminology

- Google Machine Learning Crash Course
  - https://developers.google.com/machine-learning/crash-course/ml-intro
- Machine Learning Glossary
  - https://developers.google.com/machine-learning/glossary

e.g.,

outliers

Values distant from most other values. In machine learning, any of the following are outliers:

•**Weights** with high absolute values.

•Predicted values relatively far away from the actual values.

•Input data whose values are more than roughly 3 standard deviations from the mean.

Outliers often cause problems in model training. **Clipping** is one way of managing outliers.

DNN glossary

https://developers.google.com/machine-learning/glossary

- Batch
  - The set of examples used in one **iteration** (that is, one **gradient** update) of **model training**.
- Batch size
  - A hyperparameter that defines the number of samples in a batch.
- Batch normalization
  - In a neural network, batch normalization is achieved through a normalization step that fixes the means and variances of each layer's input. It can improve the efficiency and stability of learning process.
  - **Normalizing** the input or output of the **activation functions** in a **hidden layer**. Batch normalization can provide the following benefits:
    - Make **neural networks** more stable by protecting against **outlier** weights.
    - Enable higher **learning rates**.
    - Reduce **overfitting**.

DNN glossary

https://developers.google.com/machine-learning/glossary

- Training set
  - The subset of the dataset used to train a model.
- Test set
  - The subset of the dataset that you use to test your **model** after the model has gone through initial vetting by the validation set.
- Validation set
  - A subset of the dataset—**disjoint from the training set**—used in **validation**.
- Validation
  - A process used, as part of **training**, to evaluate the quality of a **machine learning** model using the **validation set**. Because the validation set is disjoint from the training set, validation helps ensure that the model’s performance generalizes beyond the training set.

DNN glossary

https://developers.google.com/machine-learning/glossary

- Gradient
  - The vector of **partial derivatives** with respect to all of the independent variables. In machine learning, the gradient is the vector of partial derivatives of the model function. The gradient points in the direction of steepest ascent.
- Exploding gradient problem
  - The tendency for **gradients** in a **deep neural networks** (especially **recurrent neural networks**) to become surprisingly steep (high).

    Steep gradients result in very large updates to the weights of each node in a deep neural network.

    Models suffering from the exploding gradient problem become difficult or impossible to train.

    Gradient clipping can mitigate this problem.
- Gradient Vanishing Problem
  - Gradient Vanishing problem is a difficulty found in training certain ANN with gradient based methods, happening when the gradient becomes very small in deep NN caused by certain activation functions.
  - The tendency for the gradients of early **hidden layers** of some **deep neural networks** to become surprisingly flat (low). Increasingly lower gradients result in increasingly smaller changes to the weights on nodes in a deep neural network, leading to little or no learning. Models suffering from the vanishing gradient problem become difficult or impossible to train. **Long Short-Term Memory** cells address this issue.

DNN glossary

https://developers.google.com/machine-learning/glossary

- Learning rate
  - A scalar used to train a model via gradient descent. During each iteration, the **gradient descent** algorithm multiplies the learning rate by the gradient. The resulting product is called the **gradient step**. Learning rate is a key **hyperparameter**.
- Overfitting
  - Creating a model that matches the **training data** so closely that the model fails to make correct predictions on new data.
- Underfitting
  - Producing a model with poor predictive ability because the model hasn't captured the complexity of the training data. Many problems can cause underfitting, including:
    - Training on the wrong set of features.
    - Training for too few epochs or at too low a learning rate.
    - Training with too high a regularization rate.
    - Providing too few hidden layers in a deep neural network.
- Epoch
  - A full training pass over the entire dataset such that each example has been seen once. Thus, an epoch represents N/batch size training iterations, where N is the total number of examples.

DNN glossary

https://developers.google.com/machine-learning/glossary

- Loss
  - A measure of how far a model's predictions are from its label. Or, to phrase it more pessimistically, a measure of how bad the model is. To determine this value, a model must define a loss function. For example, linear regression models typically use mean squared error for a loss function, while logistic regression models use Log Loss.
- L1 loss
  - **Loss** function based on the absolute value of the difference between the values that a model is predicting and the actual values of the **labels**. L_1 loss is less sensitive to outliers than **L_2 loss**.
- L2 loss
  - The **loss** function used in **linear regression**. (Also known as **L_2 Loss**.) This function calculates the squares of the difference between a model's predicted value for a labeled **example** and the actual value of the **label**. Due to squaring, this loss function amplifies the influence of bad predictions. That is, squared loss reacts more strongly to outliers than **L_1 loss**.

DNN glossary

https://developers.google.com/machine-learning/glossary

- Regularization
  - The penalty on a model's complexity. Regularization helps prevent **overfitting** by penalizing bad weights. Different kinds of regularization include:
  - **L_1 regularization**
  - **L_2 regularization**
  - **dropout regularization**
  - **early stopping** (this is not a formal regularization method, but can effectively limit overfitting)
- L1 regularization
  - A type of **regularization** that penalizes weights in proportion to the sum of the absolute values of the weights. In models relying on **sparse features**, L_1 regularization helps drive the weights of irrelevant or barely relevant features to exactly 0, which removes those features from the model.
- L2 regularization
  - A type of **regularization** that penalizes weights in proportion to the sum of the *squares* of the weights. L_2 regularization helps drive outlier weights (those with high positive or low negative values) closer to 0 but not quite to 0. (Contrast with **L1 regularization**.) L_2 regularization always improves generalization in linear models.

DNN glossary

https://developers.google.com/machine-learning/glossary

- Activation function
  - Activation function decides if a neuron should be fired after weighted sum added a bias.
- Batch
  - A group of samples to work through before updating the internal model parameters.
  - The set of examples used in one iteration (that is, one gradient update) of model training.
- Batch size
  - A hyperparameter that defines the number of samples in a batch.
- Batch Normalization
  - In a neural network, batch normalization is achieved through a normalization step that fixes the means and variances of each layer's input. It can improve the efficiency and stability of learning process.
  - **Normalizing** the input or output of the **activation functions** in a **hidden layer**. Batch normalization can provide the following benefits:
    - Make **neural networks** more stable by protecting against **outlier** weights.
    - Enable higher **learning rates**.
    - Reduce **overfitting**.

DNN glossary

https://developers.google.com/machine-learning/glossary

- Backpropagation
  - The primary algorithm for performing gradient descent on neural networks. First, the output values of each node are calculated (and cached) in a forward pass. Then, the partial derivative of the error with respect to each parameter is calculated in a backward pass through the graph.
- Bias (math)
  - When talking about error: Bias is the algorithm‘s tendency to consistently learn the wrong thing by not taking into account all the information in the data (underfitting).
  - In feedforward operation: Bias is a constant added after weight is multiplied.
  - An intercept or offset from an origin. Bias (also known as the **bias term**) is referred to as ***b*** or w_0 in machine learning models. For example, bias is the *b* in the following formula: $y' = \mathbf{b} + w1x1 + w2x2 + \ldots wnxn$
-

DNN glossary

https://developers.google.com/machine-learning/glossary

- Dataset
  - A collection of data used to train a model.
- Dropout
  - Dropout is a regularization technique for reducing overfitting in neural networks.
  - A form of **regularization** useful in training **neural networks**. Dropout regularization works by removing a random selection of a fixed number of the units in a network layer for a single gradient step. The more units dropped out, the stronger the regularization.
- Epoch
  - In the context of machine learning, an epoch is one complete pass through the training data.
- Exploding gradient problem
  - The tendency for **gradients** in a **deep neural networks** (especially **recurrent neural networks**) to become surprisingly steep (high).

    Steep gradients result in very large updates to the weights of each node in a deep neural network.

    Models suffering from the exploding gradient problem become difficult or impossible to train.

    Gradient clipping can mitigate this problem.
- Feature
  - A feature is an individual measurable property or characteristic of a phenomenon being observed.
- Feature Map
  - In machine learning, a feature map, or activation map, is the output activations for a given filter.

DNN glossary

https://developers.google.com/machine-learning/glossary

- Gradient descent
  - Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function.
  - A technique to minimize **loss** by computing the gradients of loss with respect to the model's parameters, conditioned on training data. Informally, gradient descent iteratively adjusts parameters, gradually finding the best combination of **weights** and bias to minimize loss.
- Gradient Vanishing Problem
  - Gradient Vanishing problem is a difficulty found in training certain ANN with gradient based methods, happening when the gradient becomes very small in deep NN caused by certain activation functions.
  - The tendency for the gradients of early **hidden layers** of some **deep neural networks** to become surprisingly flat (low). Increasingly lower gradients result in increasingly smaller changes to the weights on nodes in a deep neural network, leading to little or no learning. Models suffering from the vanishing gradient problem become difficult or impossible to train. **Long Short-Term Memory** cells address this issue.
- Label
  - A label is the desired output with respect to an input data.
- Loss function
  - A loss function depicts how much the model has missed the target.
- Momentum method
  - A technique designed to speed up convergence of gradient descent by taking previous gradient into account.

DNN glossary

- Optimizer
  - In machine learning, an optimizer is an algorithm to optimize the model by minimizing the loss function
- Overfitting
  - In machine learning, overfitting is the case the loss on training set is small, but the model is not reliable on testing set.
  - Creating a model that matches the **training data** so closely that the model fails to make correct predictions on new data
- Regularization
  - Regularization is a technique that discourages learning a more complex or flexible model, so as to avoid the risk of overfitting. Regularization is nothing but adding a penalty term to the objective function and control the model complexity using that penalty term.
- Variance
  - In machine learning, variance depicts how scattered are the predicted values.
- Validation dataset
  - The validation dataset provides an unbiased evaluation of a model fit on the training dataset while tuning the model's hyperparameters.

DNN glossary

- Underfitting
  - In machine learning, underfitting occurs when a model capability can not capture the underlying trend of data.
- Training
  - An iterative process adjusting variables in a model based on the predictions it made.
- Transfer Learning
  - Transfer Learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.
- Testing
  - Use a testing dataset to evaluate the performance of a model.
- Weight
  - **Weights** are used to connect each neurons in one layer to every neuron in the next layer, determining the strength of each connection.

Activation function

- Generally three properties are expected from activation function:

- nonlinearity - that is crucial property of activation function. Thanks to that neural network can be used to solved nonlinear problems.
- continuously differentiable – which means that we have a continuous first order derivative. It is desirable property for enabling gradient-based optimization methods. Activation function which is continuously differentiable does not make any problems for gradient-based optimization methods.
- monotonic – it helps the neural network to converge easier into a more precise model.

Tomasz Bąk
Machine Learning Engineer in Boldare

Why ReLU?

- However, the is no rule that activation function must have all these mathematically nice properties.
- ReLU is nonlinear and monotonic. But it is not continuously differentiable. Other activation functions, like sigmoid and tanh, have all the 3 properties.

Why the ReLU is so popular and why it works so well?

- Activation functions like sigmoid, tanh or softsign suffer from vanishing gradient problem. Both end of these curves are 'almost-horizontal.' Gradient values at these parts of the curve are very small or have vanished. Because of that the network refuses to learn further or the learning is drastically slow.
- Rectifier does not suffer from this. However, it has another problem - dying ReLU problem. For argument lower than 0 the gradient vanishing. Neurons which went into that state stop responding to changes in input or error (simply because gradient is 0, nothing changes). A solution for that problem are ReLU modification mentioned above (Noisy ReLU, Leaky ReLU, ELU).
- Rectifiers (Relu family) are faster. Simply because they involve simpler mathematical operations. They do not require any normalization and exponential computation (such as those required in sigmoid or tanh activation functions). The training of neural network based on ReLU can be faster up to 6 times in comparison to other activation functions (see http://www.cs.toronto.edu/~fritz...).

Tomasz Bąk
Machine Learning Engineer in Boldare

Machine Learning

Hello!! R2-D2

Thanks You