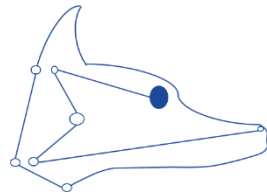
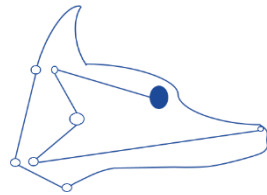
Some implementations of convolution neural network

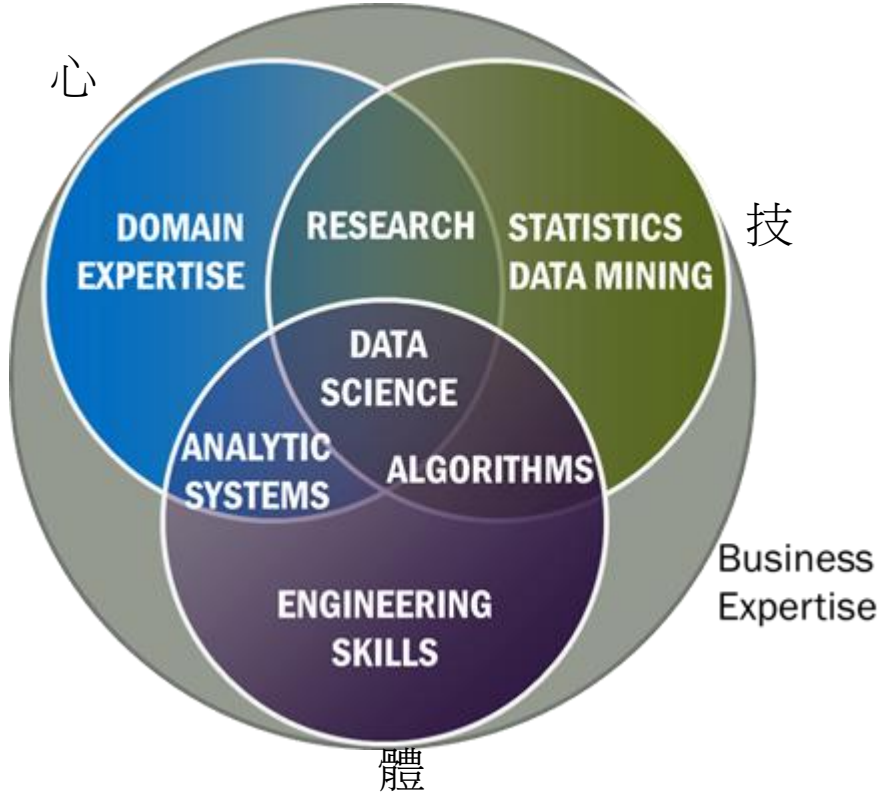
Yi-Fan Liou



What happened last time??



Mind, Skills and Body

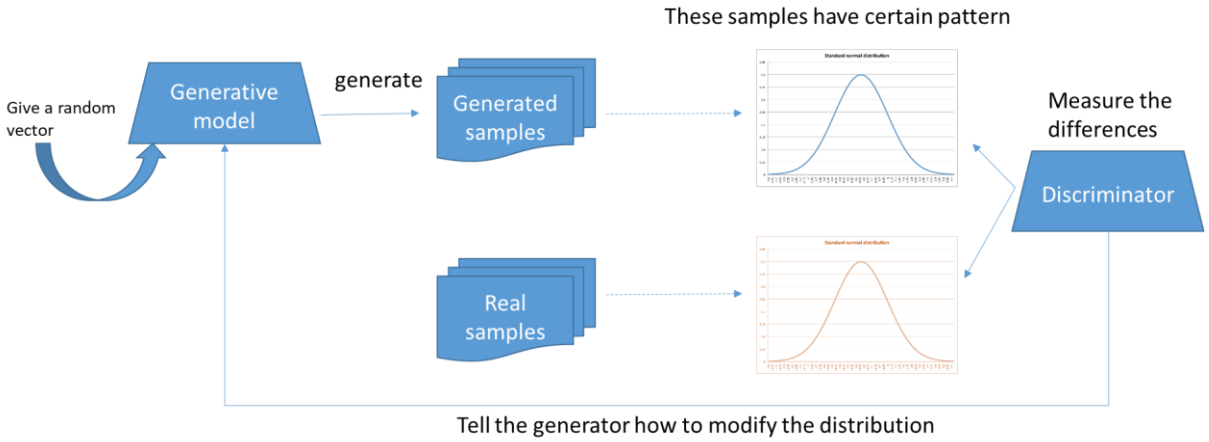


http://semanticommunity.info/Data_Science/Data_Science_for_NIST_Big_Data_Framework

The aims of this workshop

This time, we enhance this part for everyone to better understand how to use the frame work

- Enhancing the basic concepts of the math
  - Avoiding the “math signs”
  - “Math” is a represent form. But the real math could be found in the daily life
- Using the source codes for better understanding the concepts
  - Because many members of this workshop would be engineers. Using source codes would be more easy to communicate with them

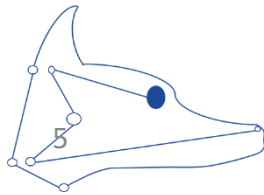


The strong discriminator cannot improve the generators’ performances

About the speaker

- Education
  - Bioinformatics and Systems Biology
  - Biotechnology
  - Life Science

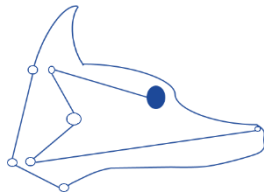
- Experiences
  - Postdoc fellow, NCKU
  - Data Scientist, Light Up Biotech. Corp.
  - Machine learning consultant, Bcondux Corp.
  - Algorithm Engineer, 京悅投資開發股份有限公司
  - Postdoc fellow, NCTU
  - Research Assistant, NCTU
  - 桃園市106年資訊組長初階及進階研習計畫 (Docker 助教)
  - 台中市107年中科自創基地Nvidia開源軟體講師



Agenda

- The Basic of Tensorflow
  - Make sure everything is ready
  - The shape of tensors
  - The basic operators
- MNIST
  - The toy dataset of this topic
- Convolutional Neural Network
- Fully Convolutional Neural Network
- YOLOv3

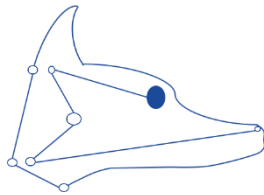
But we have a policy:
We make it step by step!!



What is “everything” in this workshop

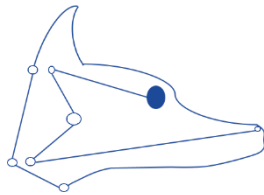
- The environments
  - Docker on Windows (recommend for this workshop)
    - You only need to pull the images we made before.
    - This workshop will also use the docker environment to make example!
  - Linux operating system
    - CUDA + CUDNN (optional)
    - GCC compiler
    - Tensorflow
- To test everything works smoothly
  - Python3
    - Import tensorflow as tf
    - sess = tf.Session()

YES, NOW WE ARE CHECKING!!



The 1st checkpoint

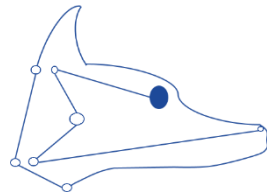
- Triger Tensorflow in docker
  - Docker run -it -v XXXX:/workspace bcondux/python3-tensorflow:cpu



That’s warm up

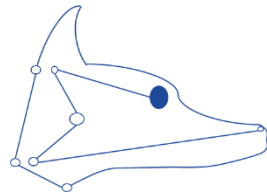
- Why deep?
  - You all would thought there is something!!
  - Let’s try
    - multilayer_perceptron_overfit.py
    - Parameters: 100 vs 200
  - “Something” is “I don’t know”

Basic operation of Tensorflow

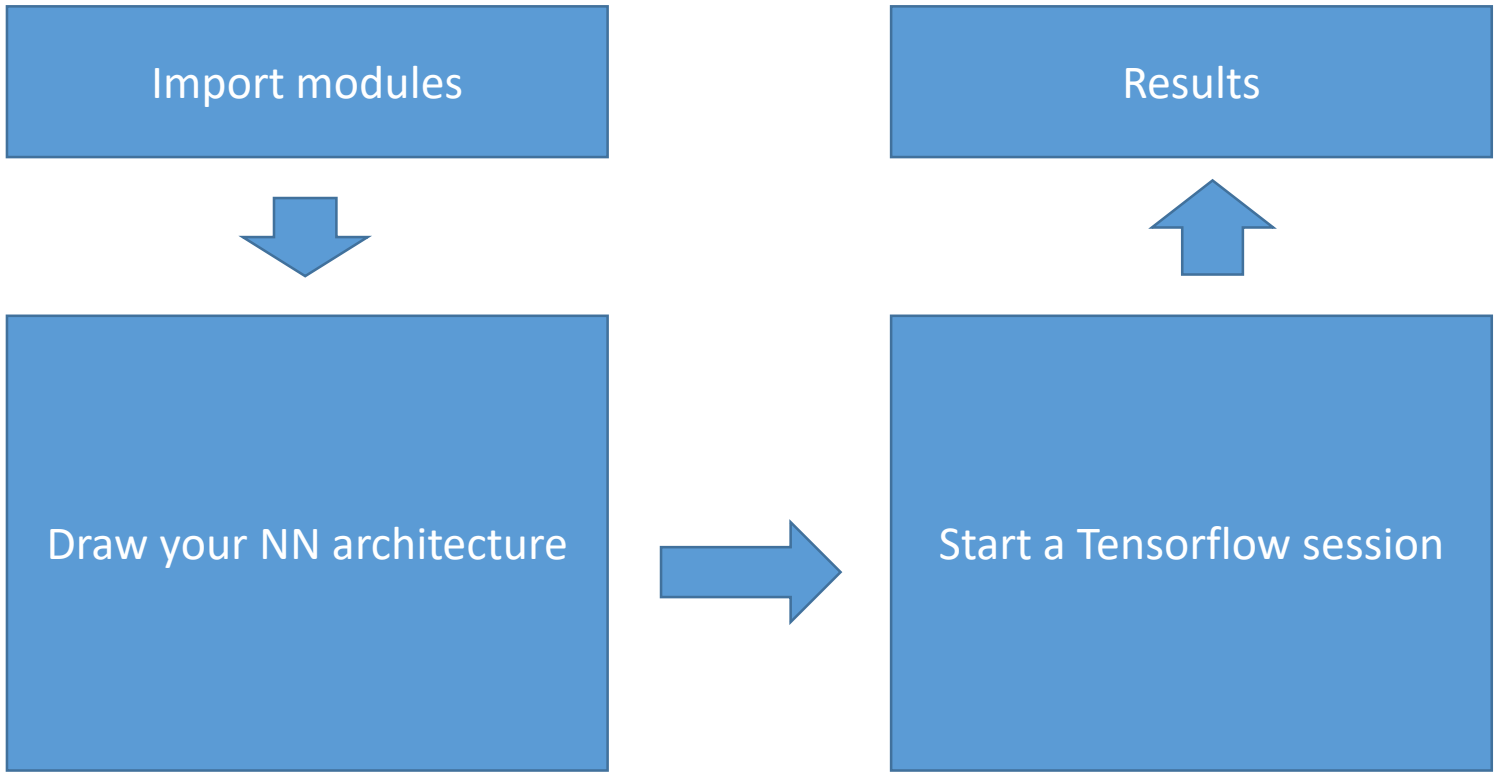


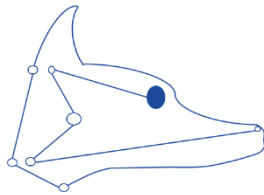
The basic concept of Tensorflow

- Static computation graph
  - You need to define the whole graph before you run a complete process
  - The framework will deploy your job to the corresponding computing nodes
- By the way
  - Dynamic computational graph
    - pyTorch
    - Eager execution



Process diagram

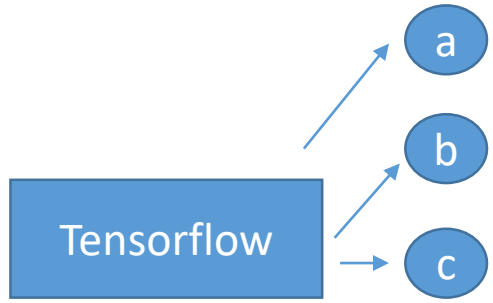




Session is a “canvas”

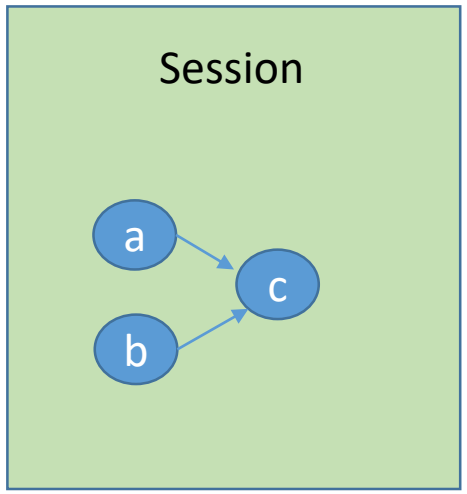
Import tensorflow as tf

a = tf.Variable([1,2,3])
b = tf.Variable([4,5,6])
c = a + b



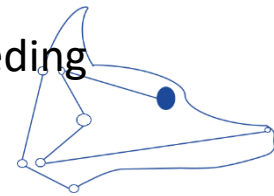
They are just occupy in the python

sess = tf.session()
sess.run(c)
sess.close()



Tip:

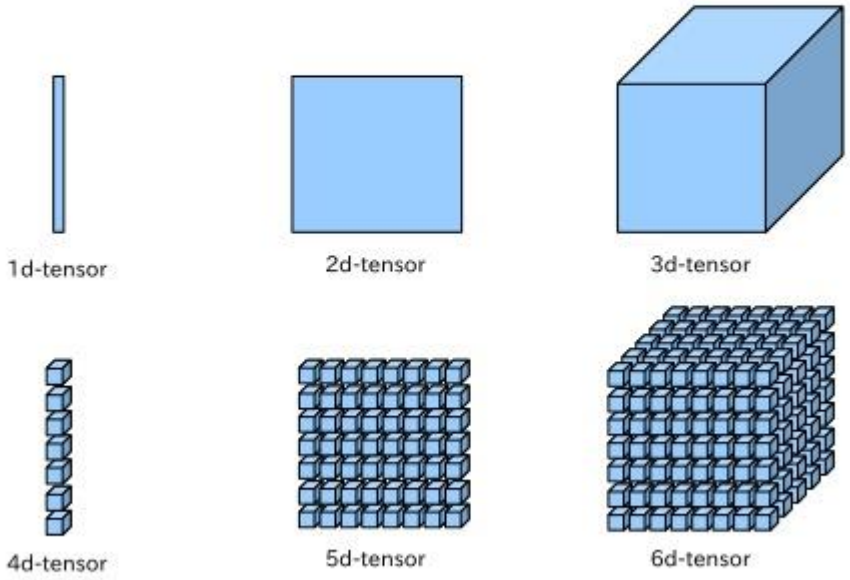
1. Once the Session announced, the architecture fixed.
2. Once the operators announced, it will live until you close Python (advance)
3. Since the architecture is fixed, we usually use ‘ delivery pores’ for feeding the data.



What is “tensor” ?

What's Tensor

Tensor is a general name of multi-way array data. For example, 1d-tensor is a vector, 2d-tensor is a matrix and 3d-tensor is a cube. We can image 4d-tensor as a vector of cubes. In similar way, 5d-tensor is a matrix of cubes, and 6d-tensor is a cube of cubes.

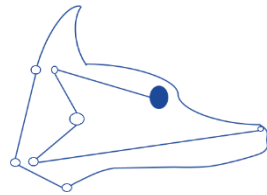


February 2, 2012 7/26

“A tensor is a generalization of vectors and matrices to potentially higher dimension”

--https://www.tensorflow.org/programmers_guide/tensors

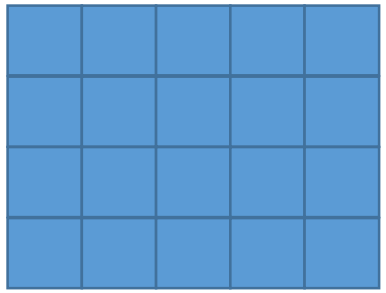
https://stackoverflow.com/questions/37849322/how-to-understand-the-term-tensor-in-tensorflow



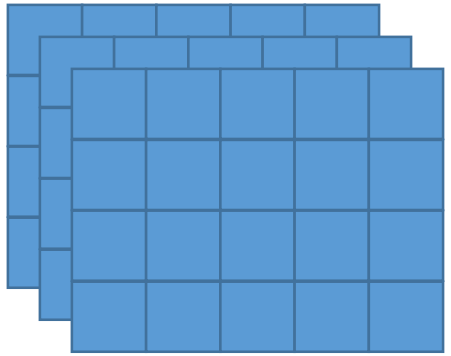
The shape of tensor

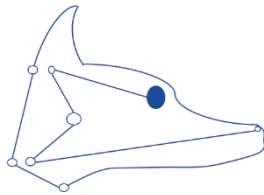
[5]

[4,5]



[3,4,5]





The basic operation in tensor calculating

We will compare the basic operator between python and Tensorflow

- Since python did not have an exact ‘array’, we use the “list” to instead

add

tf.add(x,y)

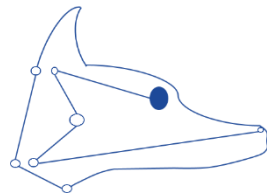
multiply

tf.matmul(x,y)

tf multiply(x,y)??

How about “weighted sum” ?

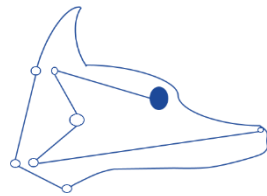
tf.reduce_sum(tf.matmul(x,y))



Example 1.

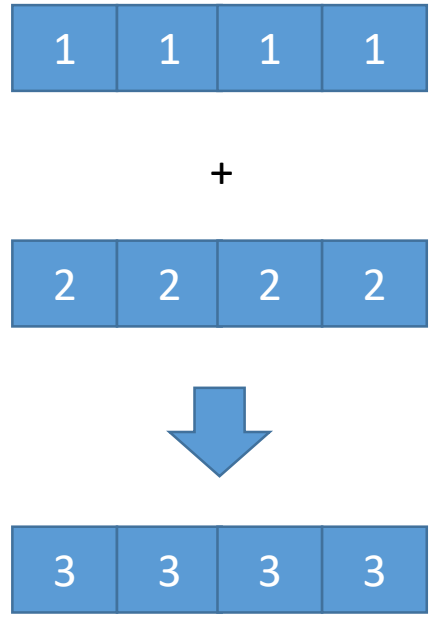
Task : we use Tensorflow to complete some simple operation

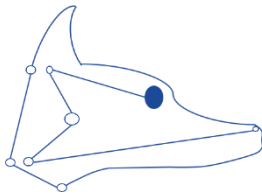
1. Add 2 tensors
2. Multiply 2 tensors
3. Division 2 tensors



tf.add

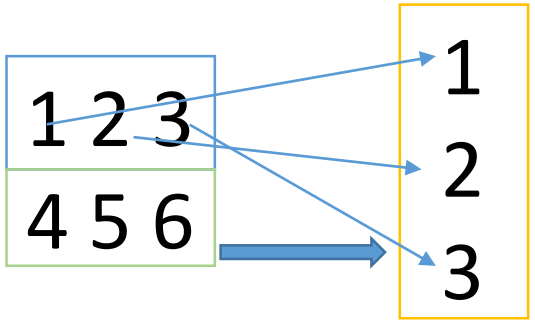
```
>>> a = tf.Variable([1,1,1])
>>> b = tf.Variable([2,2,2])
>>> c = tf.add(a,b)
>>> sess = tf.Session()
2018-06-07 16:21:08.059528: I C:\tf_jenkins\workspace\re
.cc:137] Your CPU supports instructions that this Tensor
>>> c
<tf.Tensor 'Add:0' shape=(3,) dtype=int32>
>>> sess.run(tf.global_variables_initializer())
>>> sess.run(c)
array([3, 3, 3])
>>>
```





tf.matmul

This is same to weighted sum



1 * 1 + 2 * 2 + 3 * 3 = 14

4 * 1 + 5 * 2 + 6 * 3 = 32

14
32

```
a = tf.Variable([[1,2,3],[4,5,6]])
b = tf.Variable([[1,2,3]])
c = tf.matmul(a,b)
```

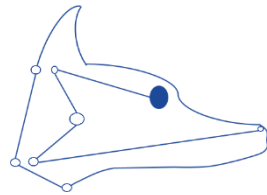
1 2 3
4 5 6

1 2
3 4
5 6

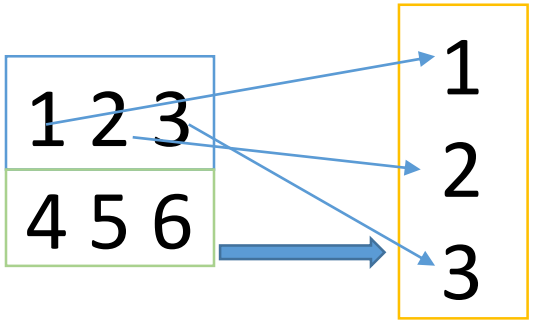
22 28
49 64

What's the answer?

```
a = tf.Variable([[1,2,3],[4,5,6]])
b = tf.Variable([[1,2],[3,4],[5,6]])
c = tf.matmul(a,b)
```



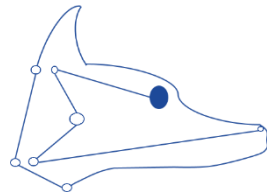
Weighted sum



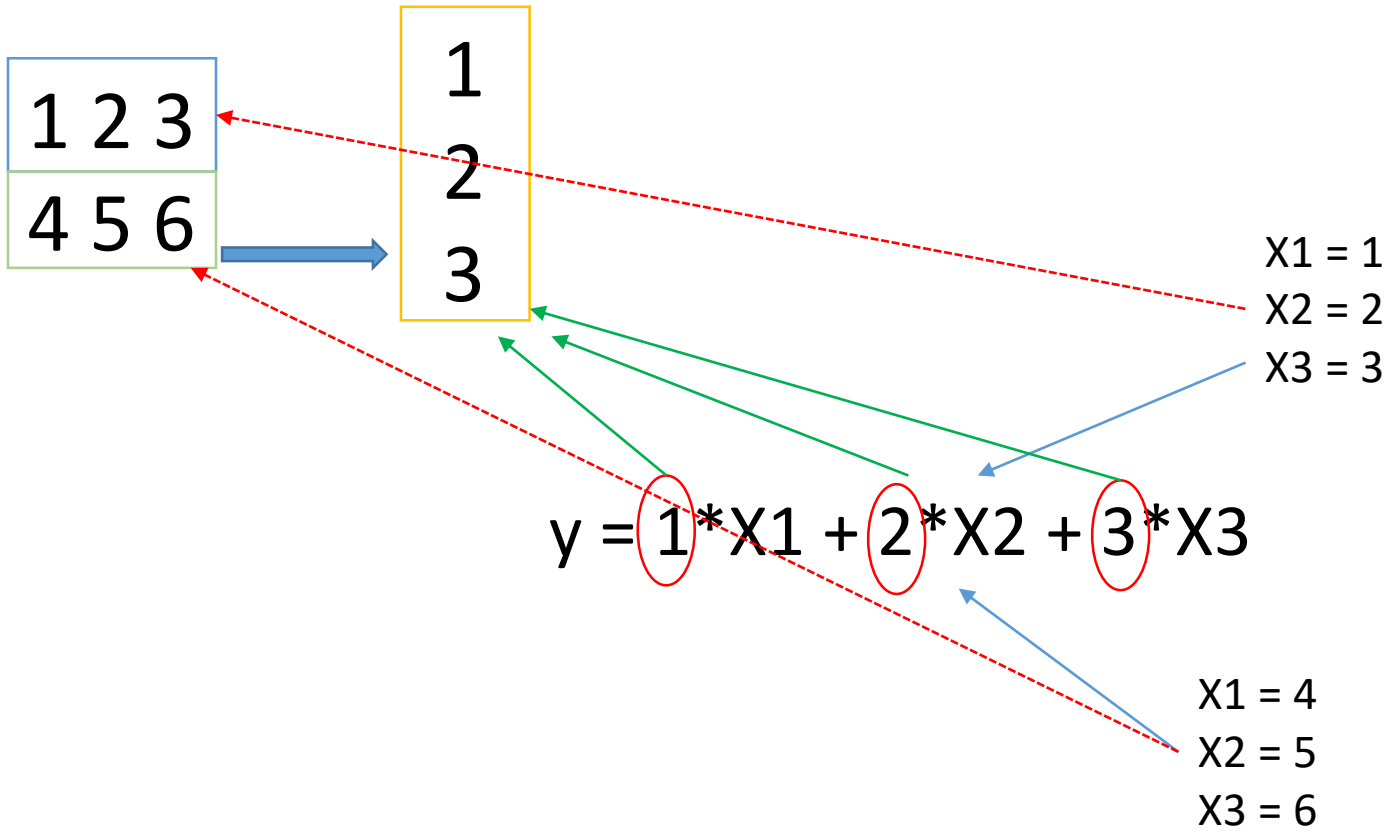
X1 = 1
X2 = 2
X3 = 3

$$y = 1*X1 + 2*X2 + 3*X3$$

X1 = 4
X2 = 5
X3 = 6



Weighted sum

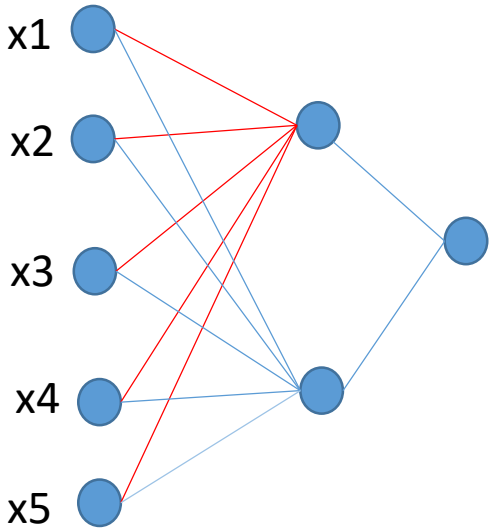


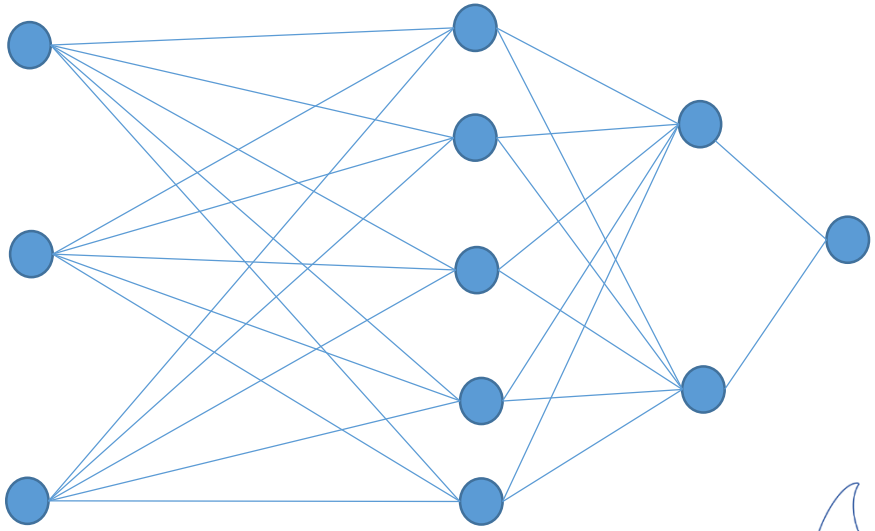
How weighted-sum used in neural network

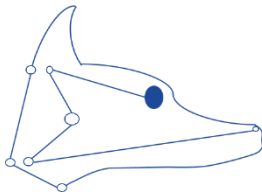
Assume we have a simple linear function:

$$F(x) = 2.5\,X_1 + 1.4\,X_2 + 0.2\,X_3 + 0.2\,X_4 + 4.5\,X_5$$

$$f(x) = 25\,X_1 + 14\,X_2 + 2\,X_3 + 2\,X_4 + 45\,X_5$$



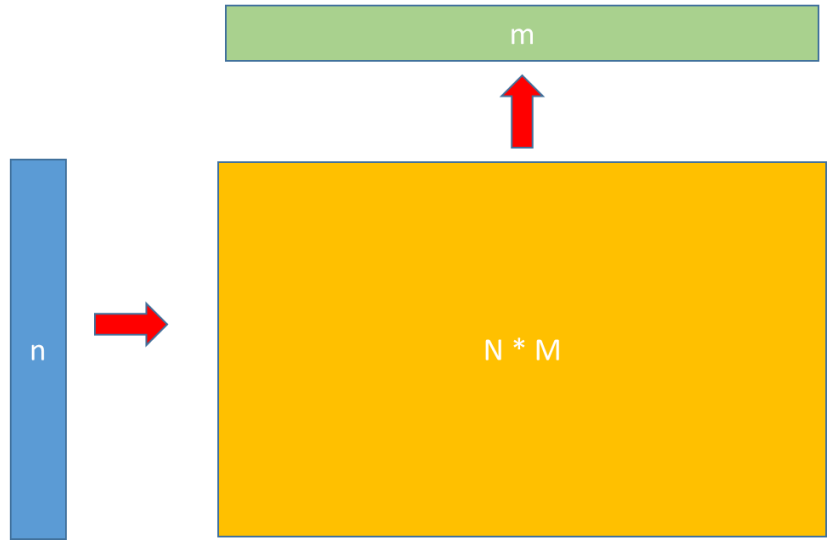


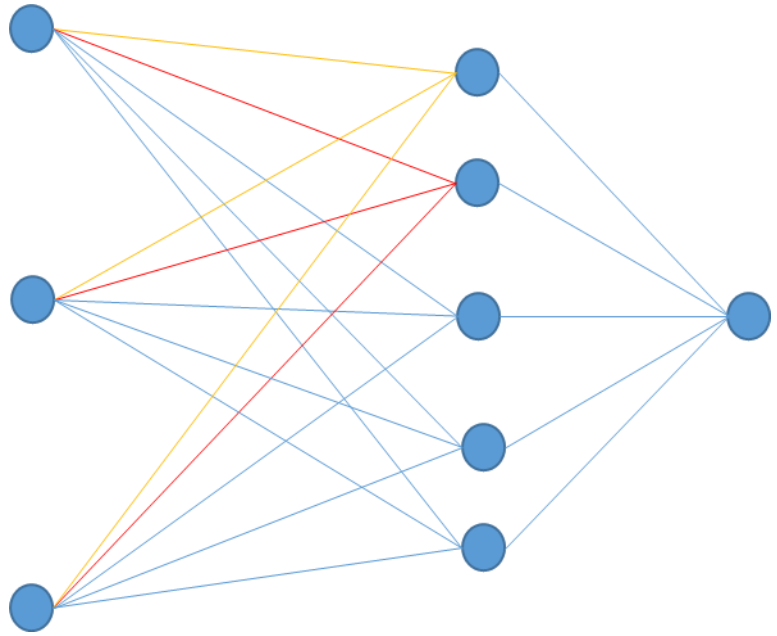


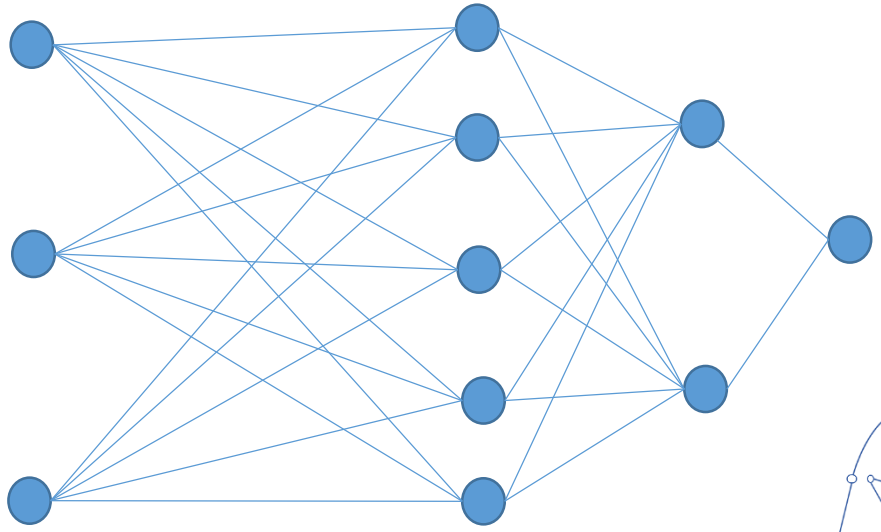
Another words

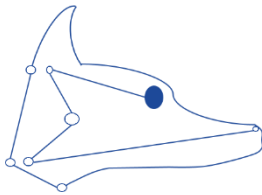
Use MNIST as example.

Make a MLP network and optimize it using gradient decent









But… How to train it??

I will try to explain it just use the essential math concept.

http://bimay.pixnet.net/blog/post/11410360-%E8%AA%AA%E8%A9%B1%E7%9A%84%E8%97%9D%E8%A1%93

Computational graph

- Considering the problem
  - How to adjust the weights?
    - We need to calculate the partial weights and partial loss for “gradient”

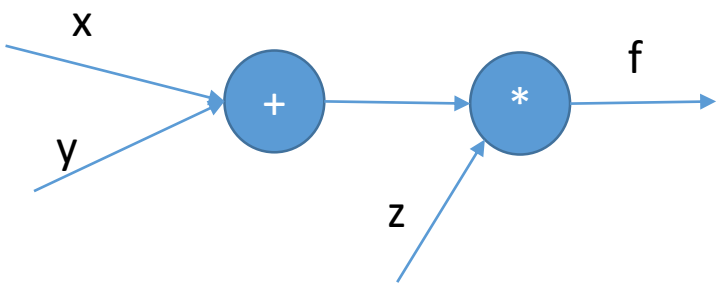
f(x,y,x) = (x+y)z

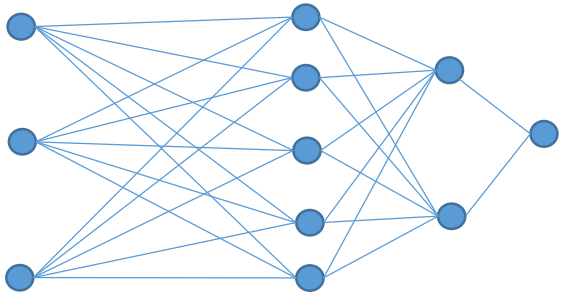
∂f/ ∂x = ?

∂f/ ∂y = ?

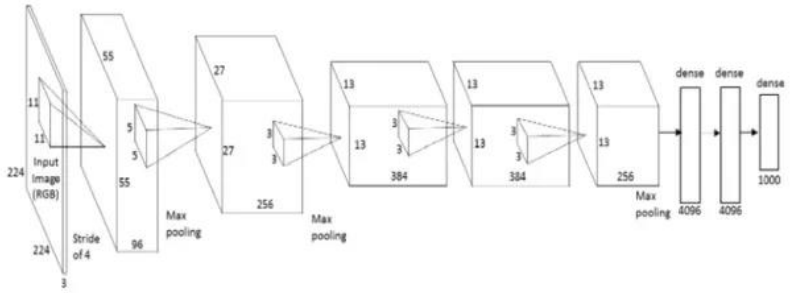
∂f/ ∂z = ?

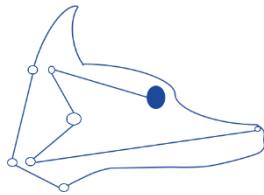
Piece of cake!!!





??





Gradient?

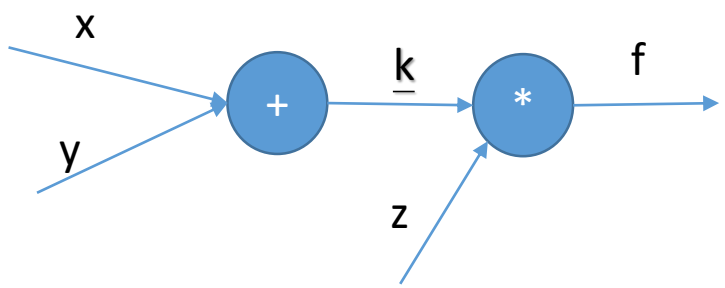
- 告訴我們“如果要讓下游結果有變化，那上游應該要要做多少調整”
- 用這概念來簡化計算圖，我們只需要知道節點的上下游，然後分開處理
- 一樣用簡單的例子來說明

f(x,y,x) = (x+y)z

∂f/ ∂x = ?

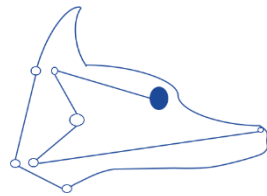
∂f/ ∂y = ?

∂f/ ∂z = ?

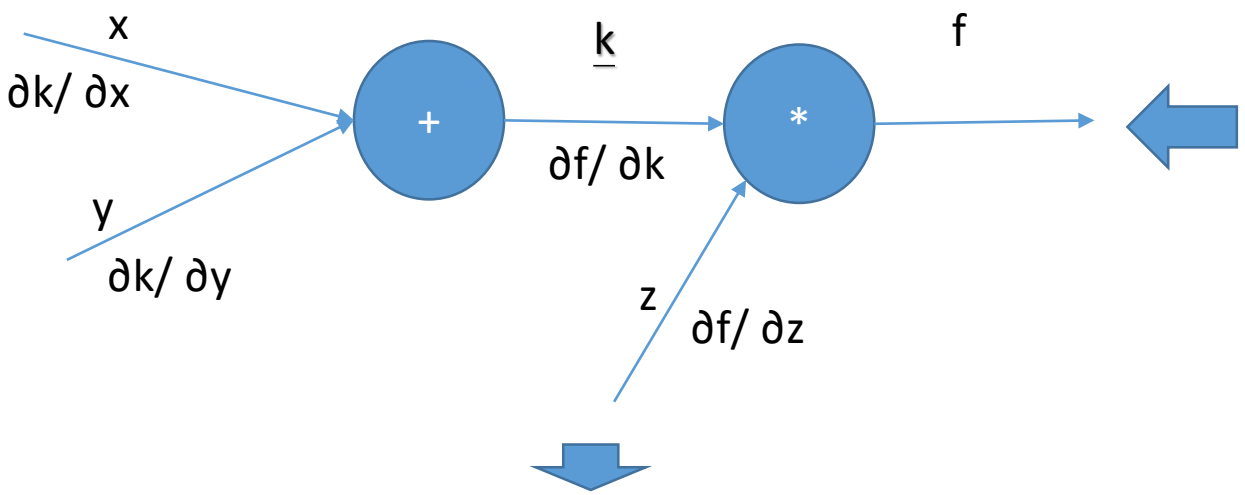


Chain rule

∂f/ ∂x = ∂f/ ∂y * ∂y/ ∂x



f(x,y,x) = (x+y)z

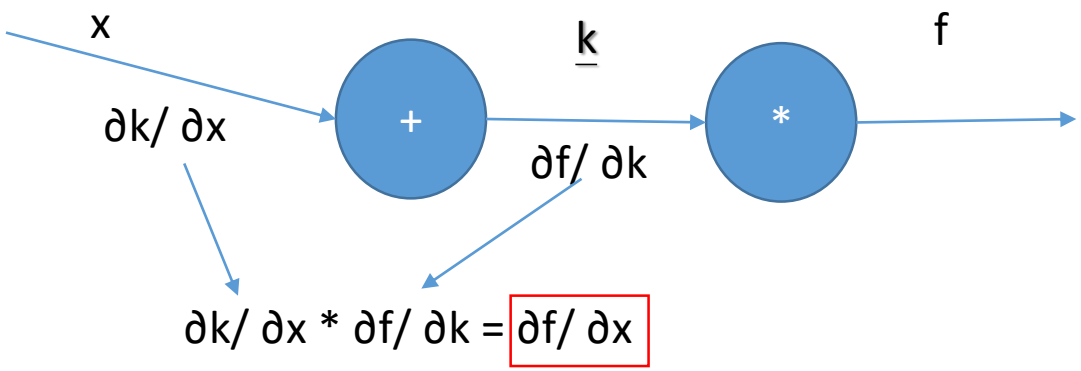


因為我們需要知道“下游結果”，所以最下游一定就是可以輕易知道結果的位點

∂f/ ∂x = ?

∂f/ ∂y = ?

∂f/ ∂z = ?



k = x + y

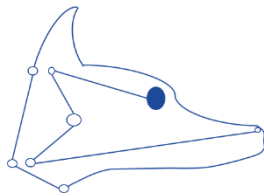
f = k * z

∂k/ ∂x = 1

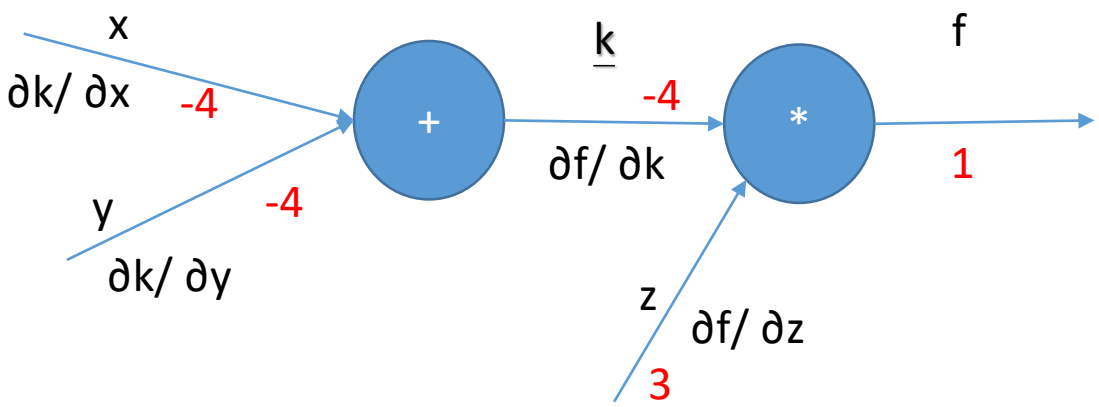
∂k/ ∂y = 1

∂f/ ∂k = z

∂f/ ∂z = k



f(x,y,x) = (x+y)z



k = x + y
f = k * z

∂k/ ∂x = 1
∂k/ ∂y = 1
∂f/ ∂k = z
∂f/ ∂z = k

∂f/ ∂x = ?

∂f/ ∂y = ?

∂f/ ∂z = ?

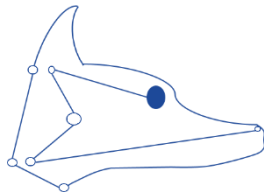
x = -2, y = 5, z = -4

The 1st small conclusion:

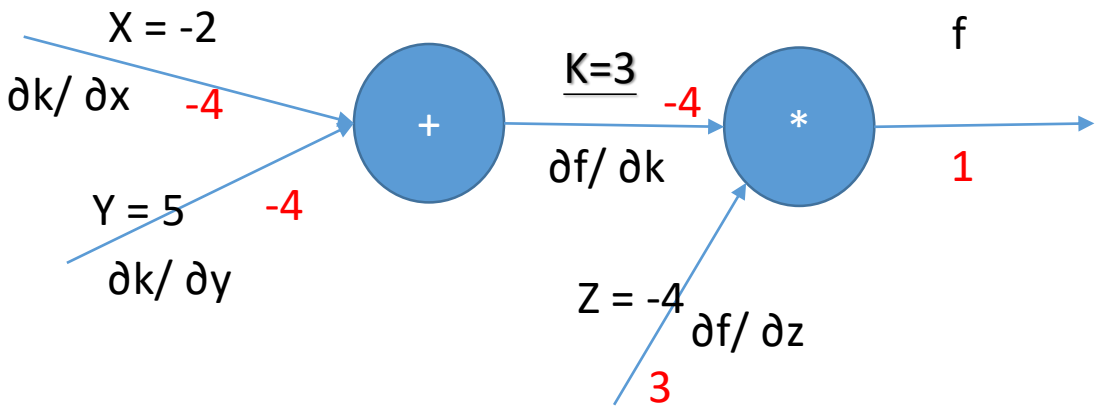
The computational graph can simplify the relationship between different functions

The 2nd small conclusion:

If we recoded the relations between variables, we can count the gradient easier



f(x,y,x) = (x+y)z



k = x + y
f = k * z

∂k/ ∂x = 1
∂k/ ∂y = 1
∂f/ ∂k = z
∂f/ ∂z = k

∂f/ ∂x = ?

∂f/ ∂y = ?

∂f/ ∂z = ?

x = -2, y = 5, z = -4

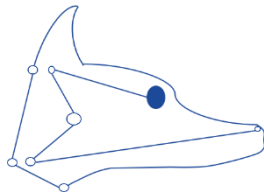
The 3rd small conclusion:

1. "ADD" means "gradient distributor"
2. "MULTUPLY" means "gradient switcher"

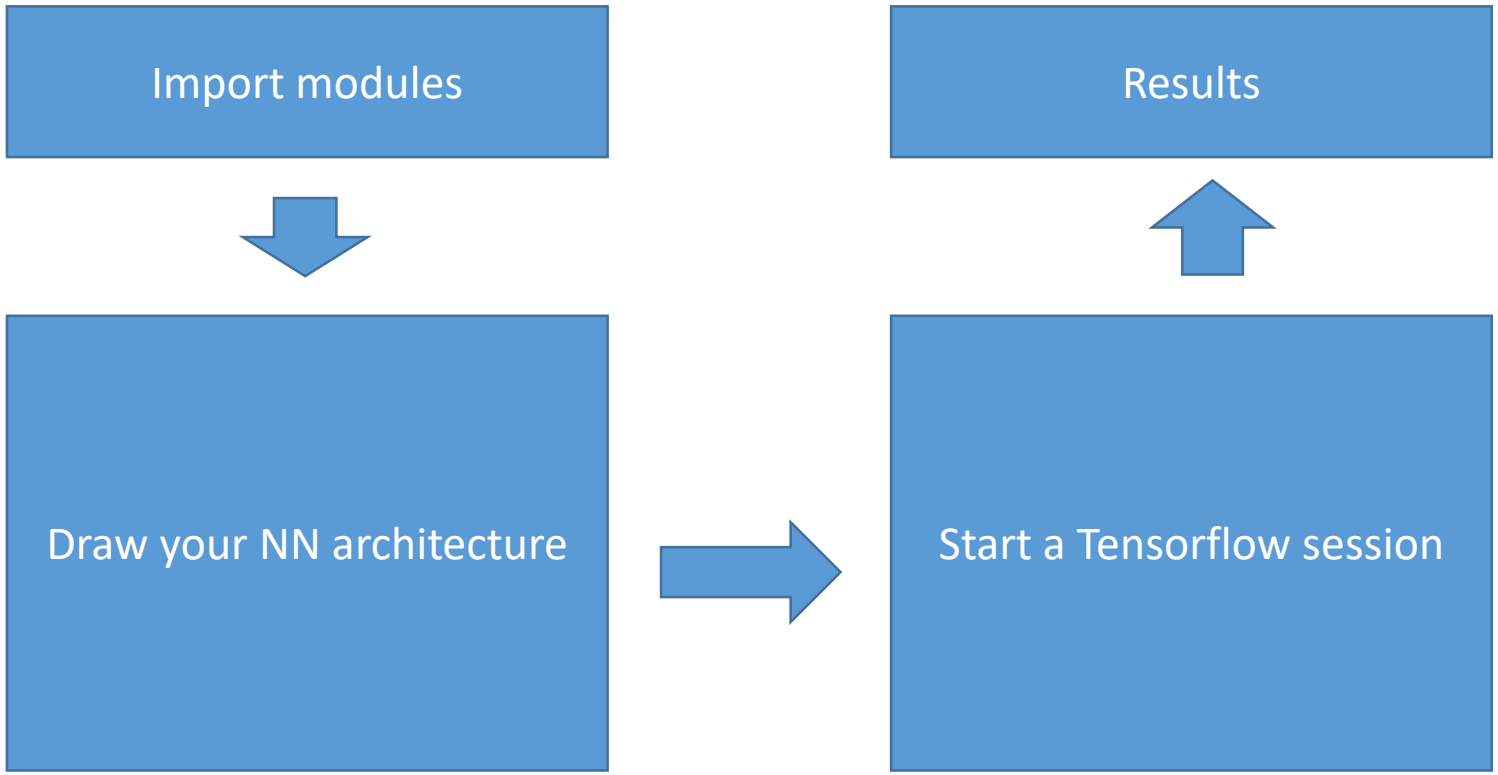
You can find more interesting characteristics from computational graph

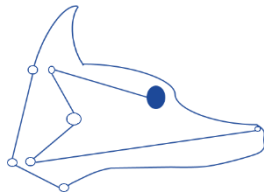
Now, you can write your code in the same space and remember the relationship between them.
Like:

```
Class ADD(obj)
    def forward(x,y)
        return x+y
    def backward(dz)
        dx = ……..
        dy = ……..
        return dx, dy
```



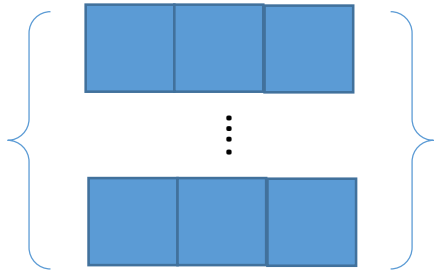
Process diagram





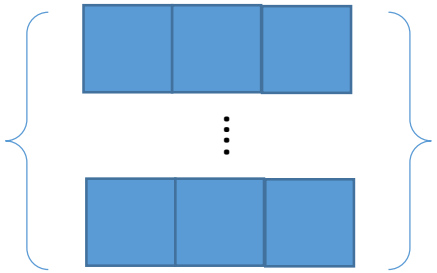
Placeholder and feed dictionaries

p2 = tf.placeholder(dtype=tf.float32, shape=[800,3])



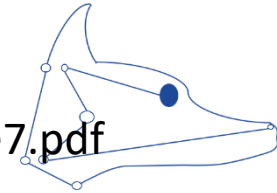
800

p1 = tf.placeholder(dtype=tf.float32, shape=[None,3])

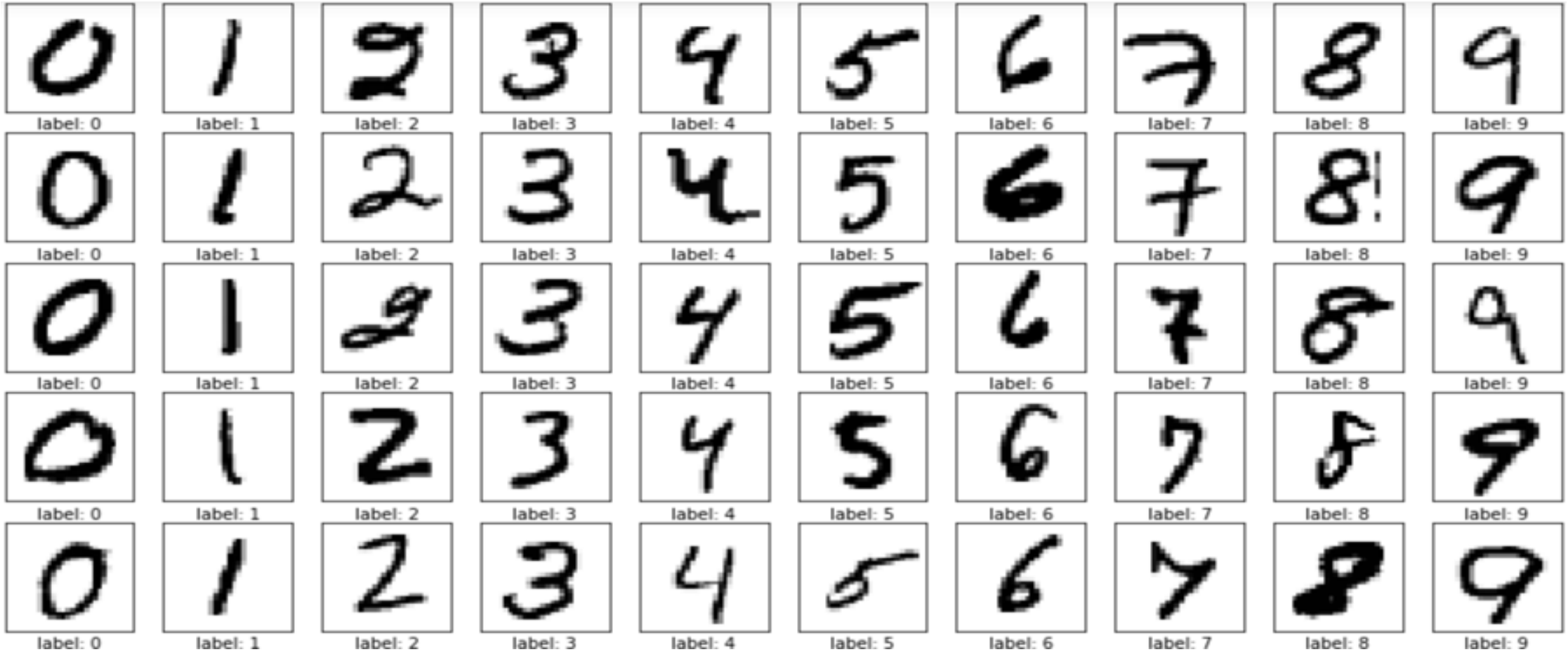


dynamic

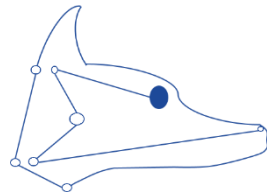
https://cs224d.stanford.edu/lectures/CS224d-Lecture7.pdf



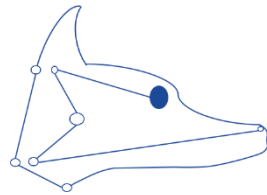
MNIST



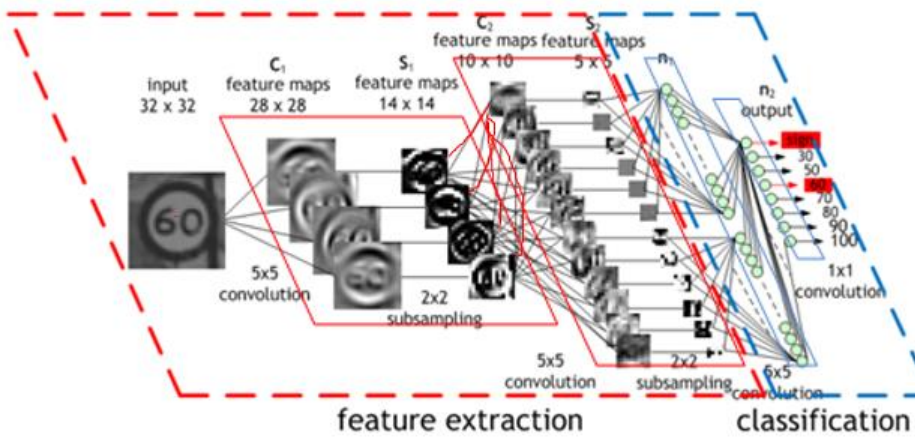
We use this dataset for further example code



Convolutional neural network

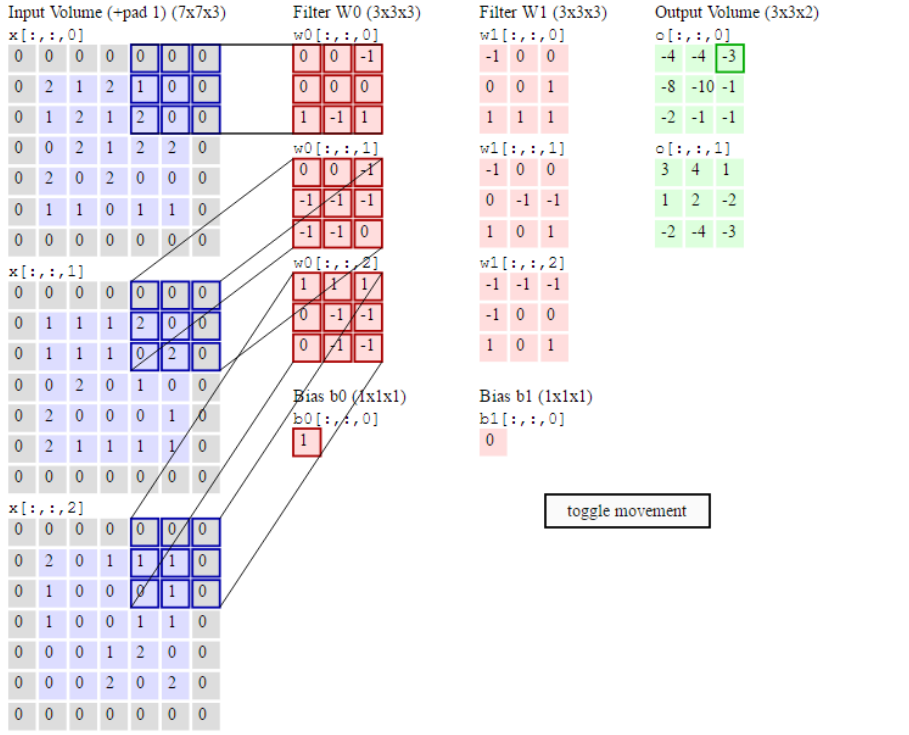


How about convolutional neural network



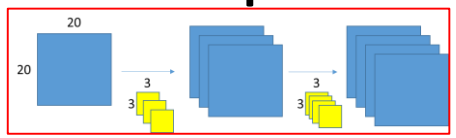
http://www.cnblogs.com/walccott/p/4957076.html

- CNN is also a kind of NNs, but the hidden nodes have their own receptive fields and sharing their weights
- Each “feature map” is a node.
- The ability for classification is still from the fully connected NN.
  - This also means, appending deeper layer here would cause gradient vanishment



https://www.quora.com/In-a-convolutional-neural-network-how-is-convolution-defined-with-an-RGB-image

The parameters – CNN as example in detail



original

20

20

3

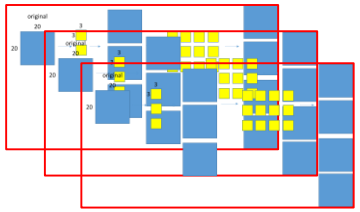
3

batch

100

??

accelerating



The precious counting miss these condition !

(This is also HMM)

Condition :

1. No pooling layer
2. Do not consider the activation function

Correction:

1. 20 * 20 = 400
2. 3 * 3 * 3 = 27
3. 400 *3 = 1200
4. 3 * 3 * 4 * 3 = 108
5. 400 * 4 = 1600

Total:

400 + 27 + 1200 + 108 + 1600 = 3335

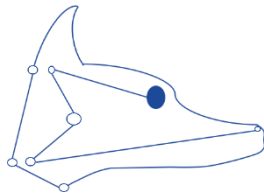
3335 * 4 (byte) = **13340**

(previous : 556 bytes)

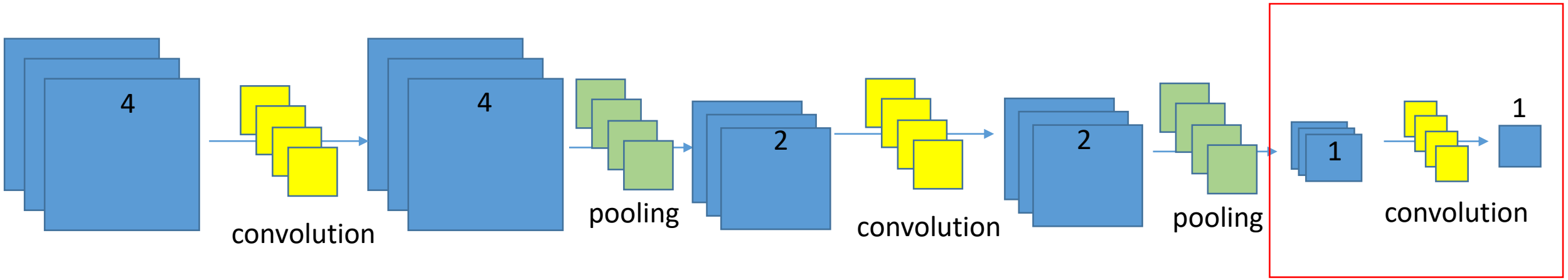
Convolutional neural network practice

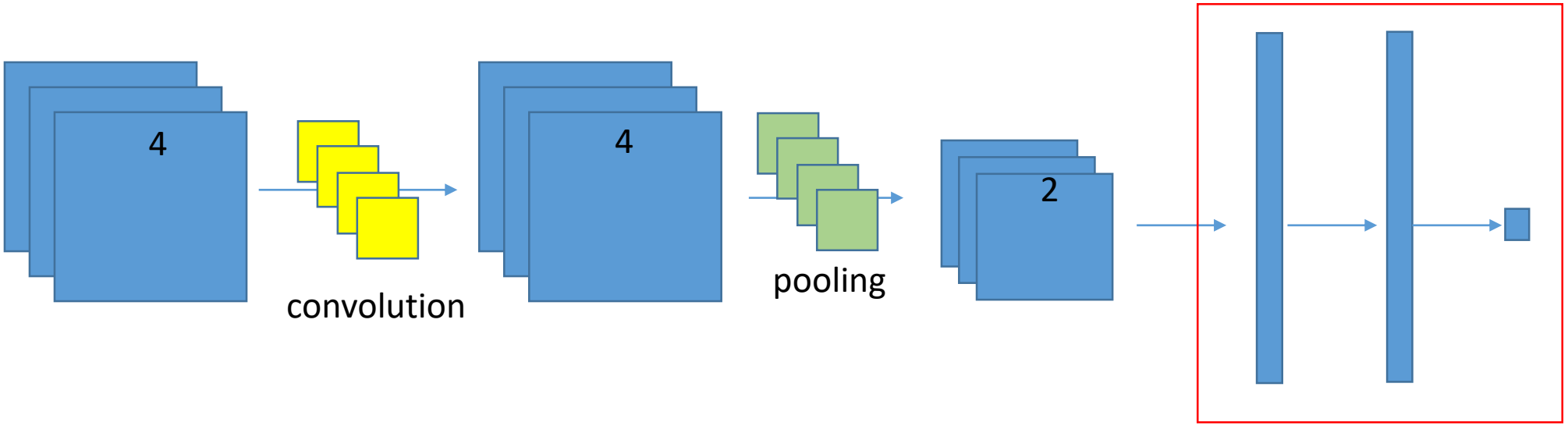
- Many convenient tools are already invented. But to better understanding how the tensors flow, we try to implement it as detailed as possible.

- convolutional_network_raw_Q.py



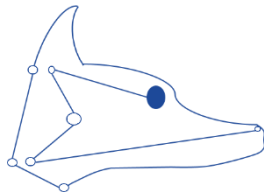
Fully convolutional neural network





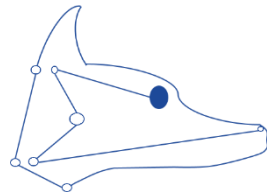
Fully convolutional neural network

- fully_convolutional_network_raw_Q.py



YOLO – you only look once

https://github.com/pjreddie/darknet



Objection

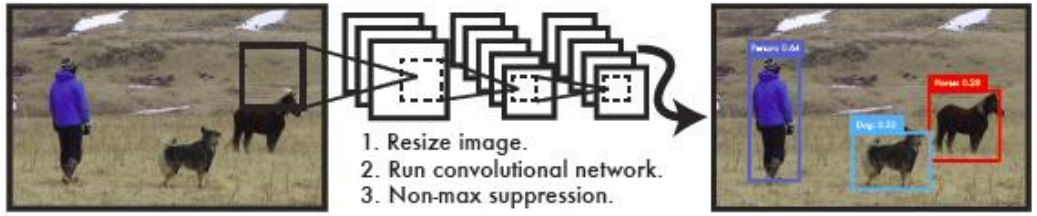
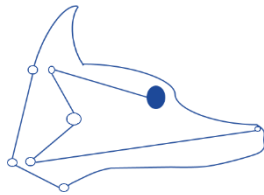


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

1. The original picture size is 224*224. The size here would influence the boundary center location. So they enlarge the picture size tp 448*448
2. The basic concept is use the same CNN to detect if there is an object in the region.
3. YOLO combine the object classification and region proposal.

Another good resource: http://christopher5106.github.io/object/detectors/2017/08/10/bounding-box-object-detectors-understanding-yolo.html



Main idea

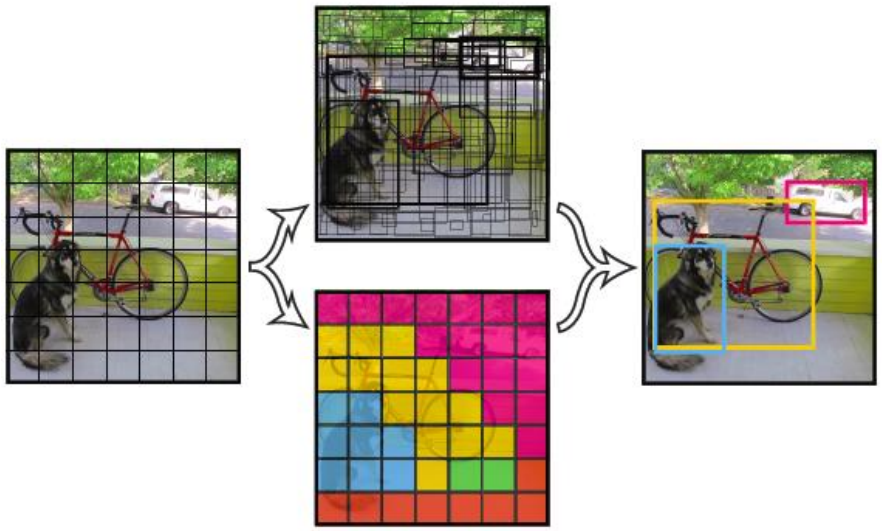
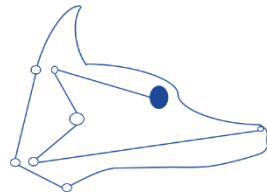


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an even grid and simultaneously predicts bounding boxes, confidence in those boxes, and class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

1. The image is separated in to S by S grids.
2. The bounding box is presented using 5 vectors:
   (x, y, H, W, C)
   x, y means the bounding box center. The box will be generated according to put the box center at (x, y). Then the box is seted to H by W.
3. This means each grid is responsible for a box. If the bonding box's center is located in certain box, this grid will be proposed.
4. C means the confidence. Confidence is calculated by counting the IOU*Pr(obj). If the IOU or obj. do not covered, this value will be zero.
5. This network can also give the "class label" at the same time.



The Convolution layers

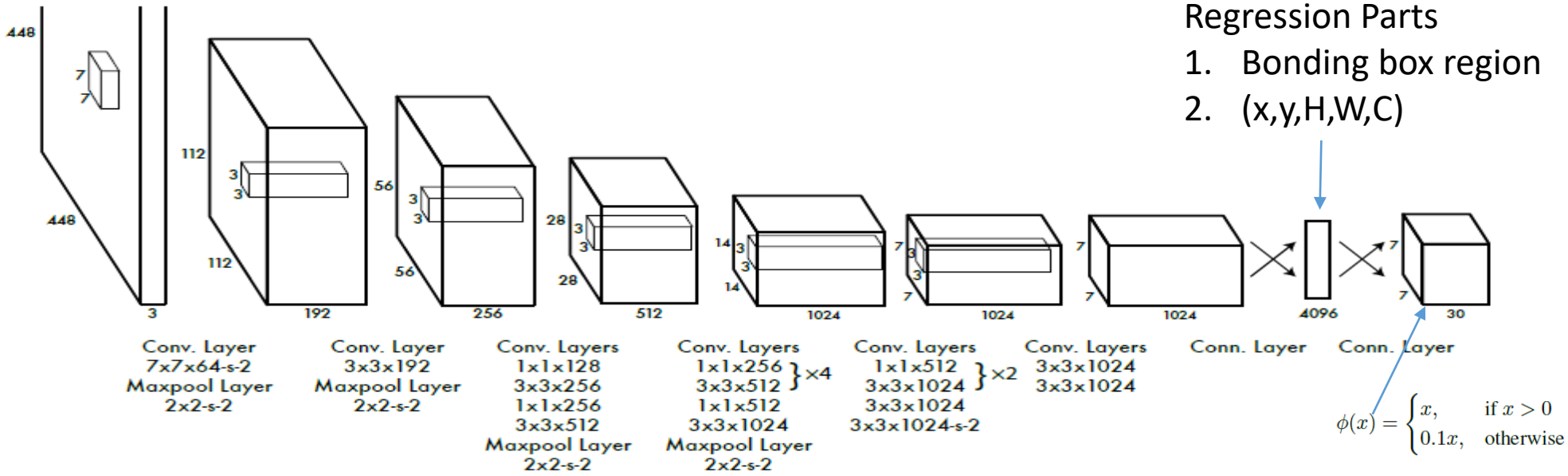
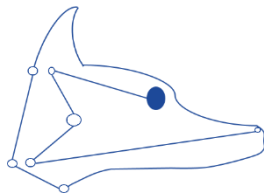
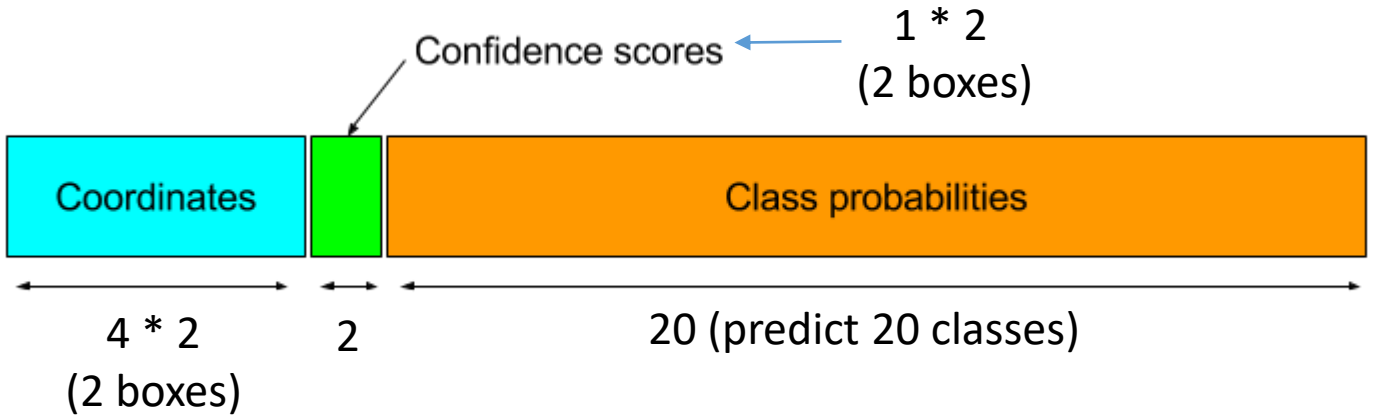


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

1. These layers are common convolutional layers.
2. They append the fully connected layer for regression.
3. The convolutional layers use the pre-train model
4. The final output use the leaky-rely with lambda=0.1



Final output tensors



1. The coordinates is (x, y, H, W)
2. The confidence score is mentioned before
3. Class probability means this bounding box have any class of it.

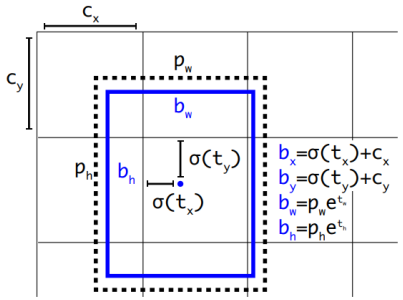
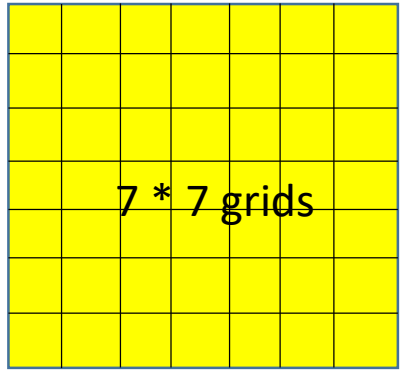
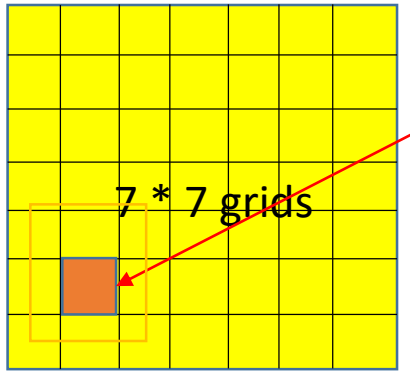
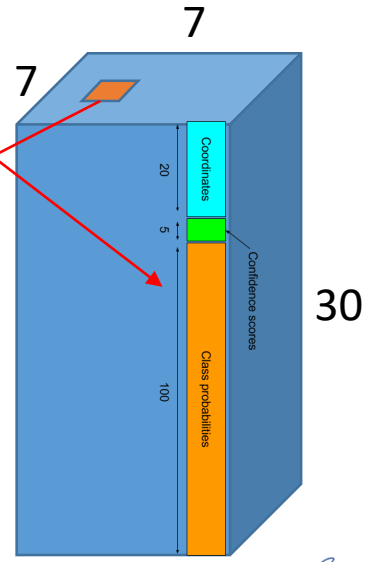


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

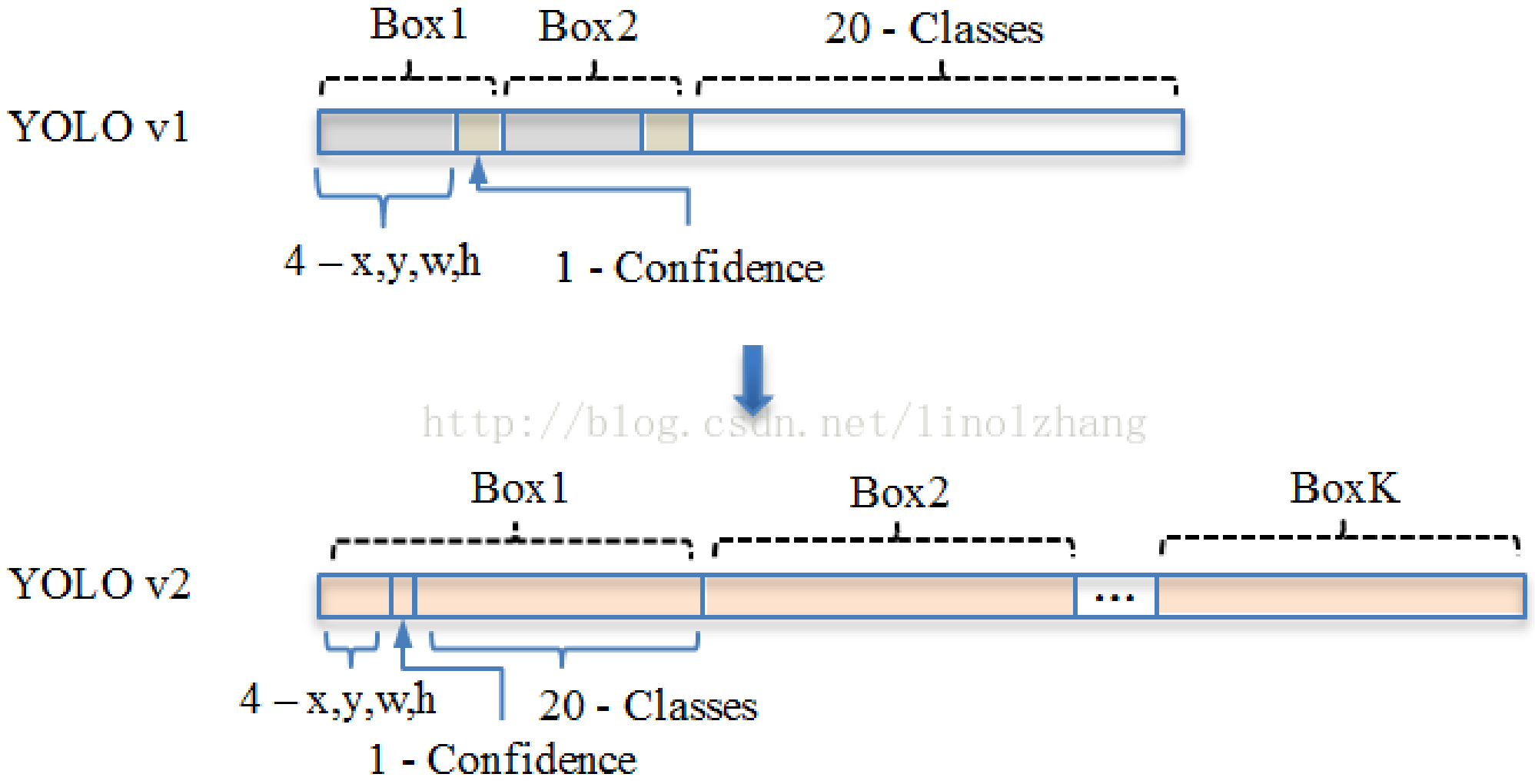




This tensor can make a region



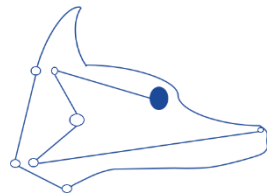
V1 vs V2


Box1
Box2
20 - Classes
YOLO v1
4 – x,y,w,h
1 - Confidence
http://blog.csdn.net/linolzhang
Box1
Box2
BoxK
YOLO v2
...
4 – x,y,w,h
20 - Classes
1 - Confidence

Darknet-19

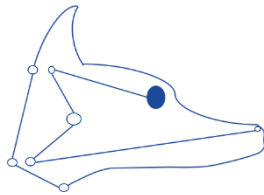
Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Table 6: Darknet-19.



V3

https://pjreddie.com/media/files/papers/YOLOv3.pdf



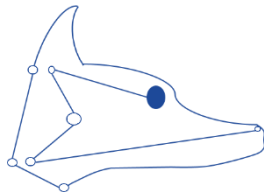
Architecture

This architecture has 53 convolutional layers

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 1. **Darknet-53.**

Residual network concept



Region prior

To solve the small object detecting problems, the prediction use masks for several stage.
The strategy is upsampling when YOLO ouput region results.

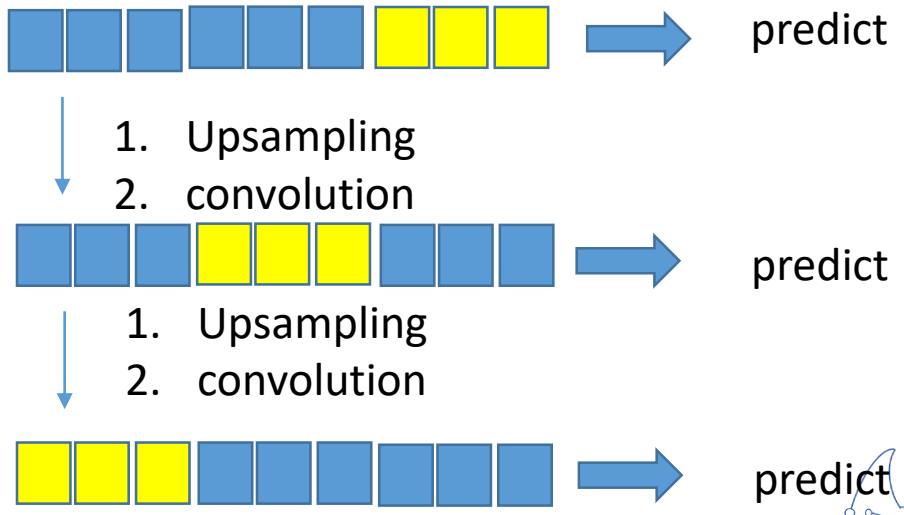
The clusters is set to 9:

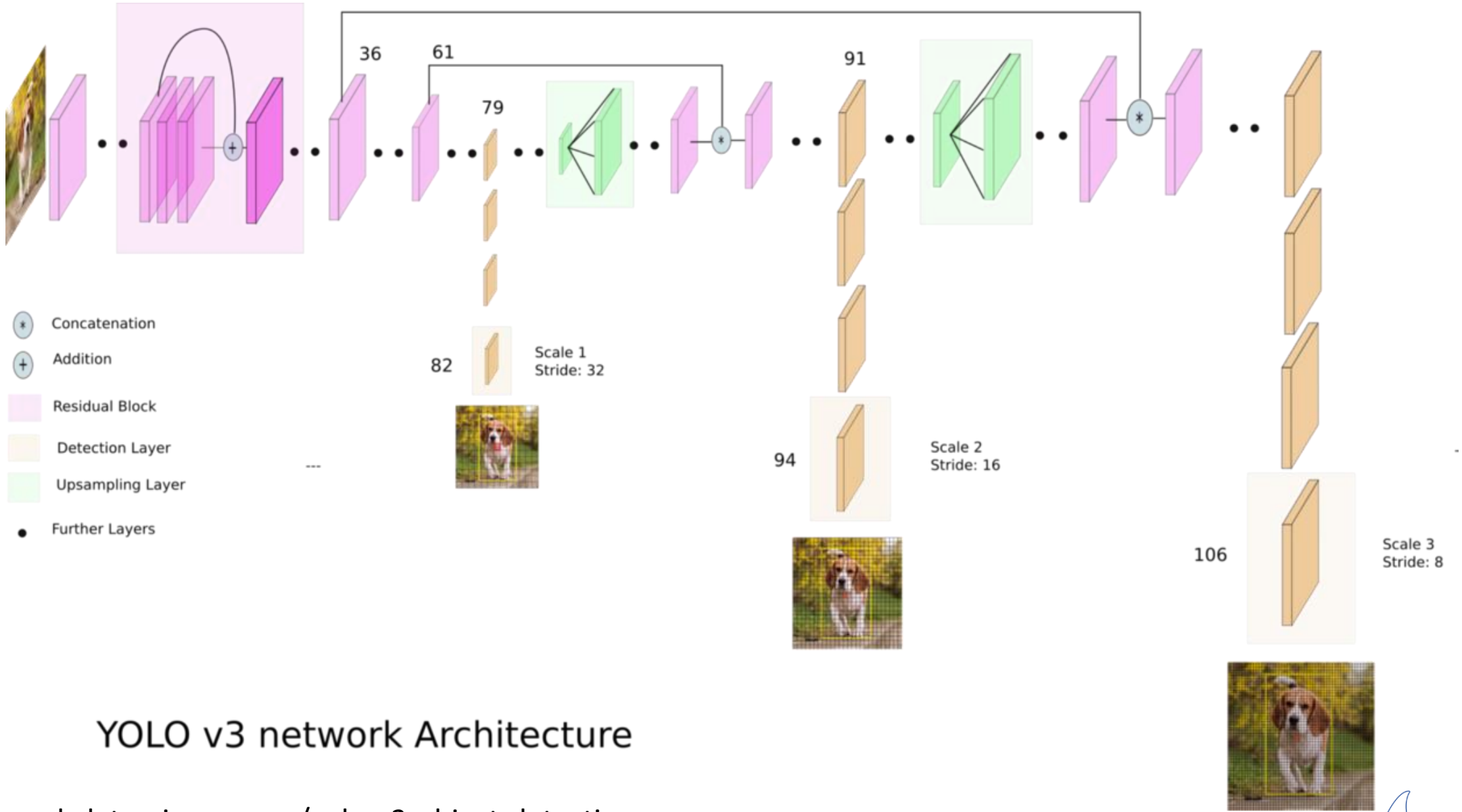
(10 X 13)
(16 X 30)
(33 X 23)
(30 X 61)
(62 X 45)
(59 X 119)
(116 X 90)
(156 X 198)
(373 X 326)

These information is not provide in YOLOv2

For example:

```
[yolo]
mask = 6,7,8
anchors = 10,13,  16,30,  33,23,  30,61,  62,45,  59,119,  116,90,  156,198,  373,326
classes=20
num=9
jitter=.3
ignore_thresh = .5
truth_thresh = 1
random=1
```

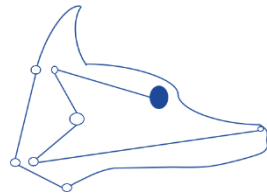




YOLO v3 network Architecture

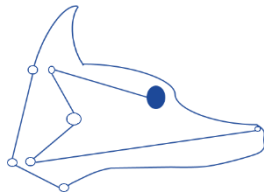
https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

How to run Darknet



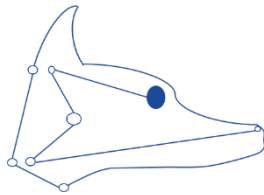
Pre-request

- Docker on Windows
  - Docker run -it --rm bcondux/python3-tensorflow:cpu



Play the YOLO models

- Start the Docker
- The latest version is YOLOv3
  - https://github.com/pjreddie/darknet.git
  - Download the YOLOv3 model
    - wget https://pjreddie.com/media/files/yolov3.weights
  - Run the
    - darknet detector test cfg/coco.name cfg/yolov3.cfg yolov3.weights [pics]
  - Reference
    - https://pjreddie.com/darknet/yolo/



How to train it?

- The dataset – using the existing
  - The COCO dataset
    - cp scripts/get_coco_dataset.sh data
    - cd data
    - sh get_coco_dataset.sh
    - Tips:
      - if you use the tensorflow image, the wget, unzip, paste should be install
      - If you use Cygwin, you also need to check the tools listed above. And the last two commands are need to execute by hand
  - Specify the path of the COCO dataset
    - vim cfg/coco.data
    - train = /workspace/darknet/data/coco/trainvalno5k.txt
    - backup = /workspace/darknet/backup
- Start to train
  - ./darknet partial cfg/yolov3.cfg yolov3.weights yolov3.weights4train 30
  - ./darknet detector train cfg/coco.data cfg/yolov3.cfg yolov3.weights