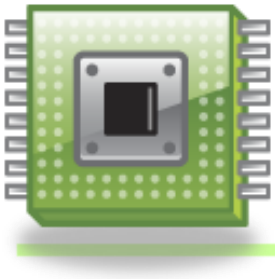


Using Yolov3 on server overview

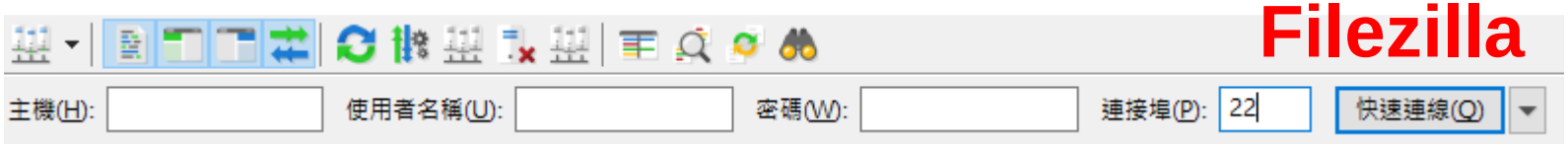
2020/10/06

Connecting the server(1/9)

- **What you need**
  - Putty
  - Xming
  - Filezilla
  - Notepad / PSpad

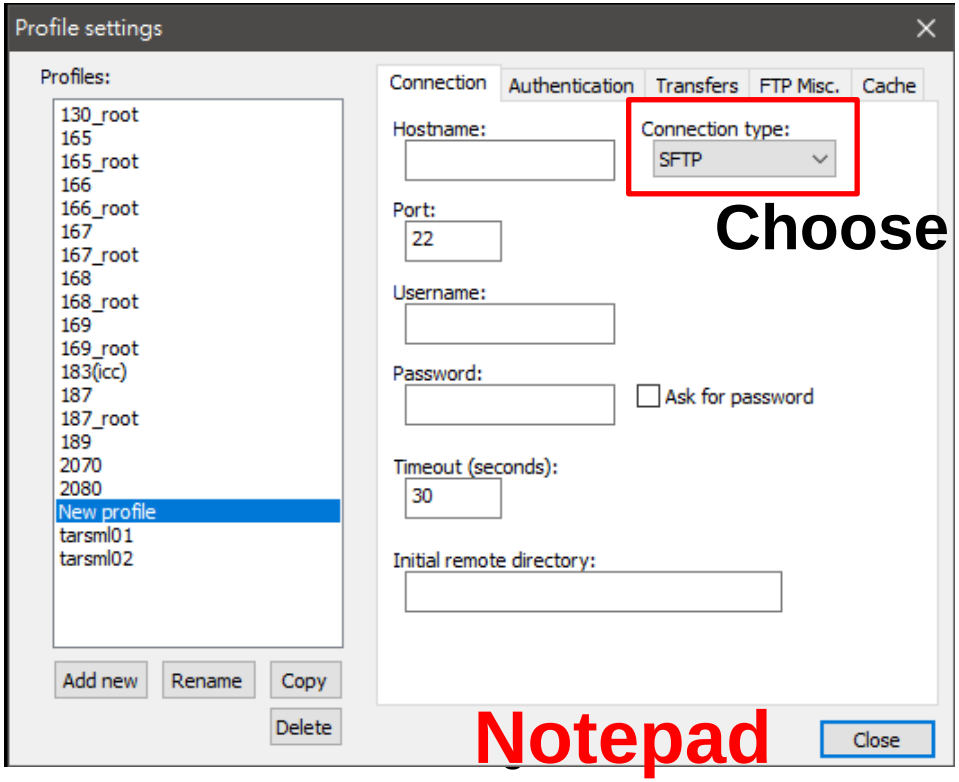
- **Account & Password**

Connecting the server(2/9)



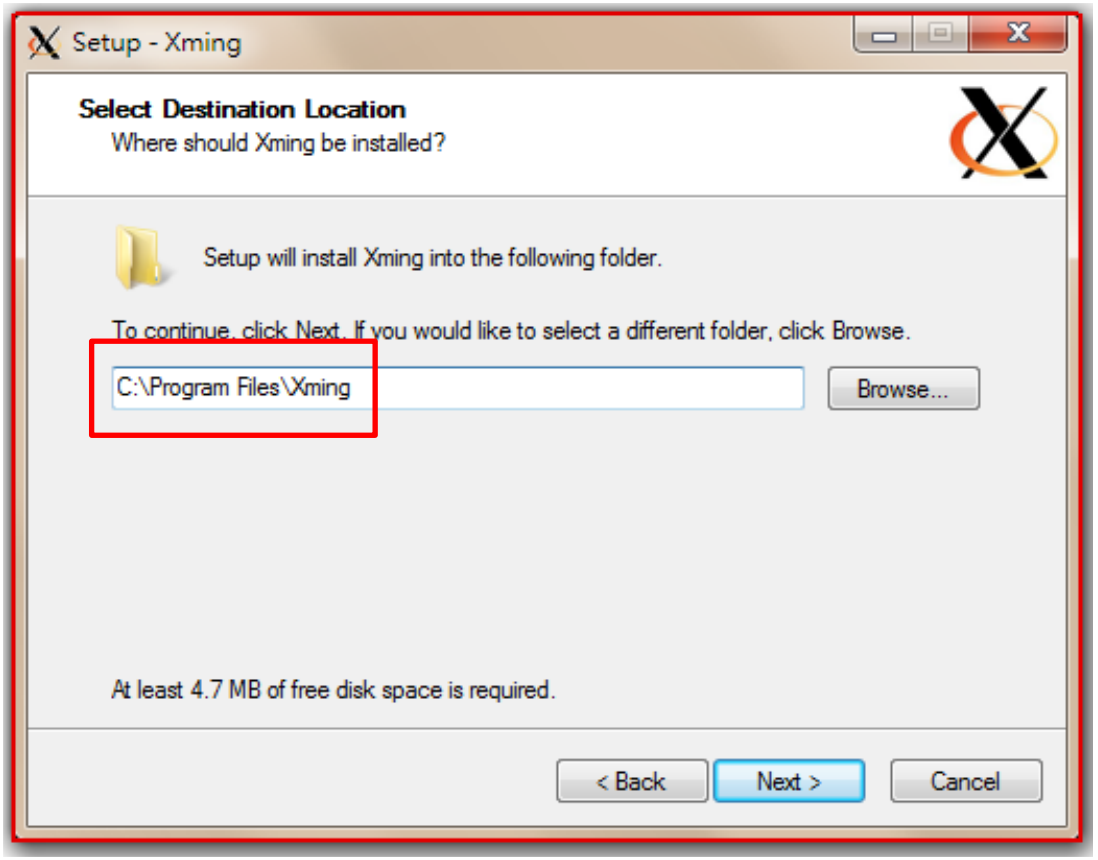
Enter IP / Account / Password

The client port of our server is 22.



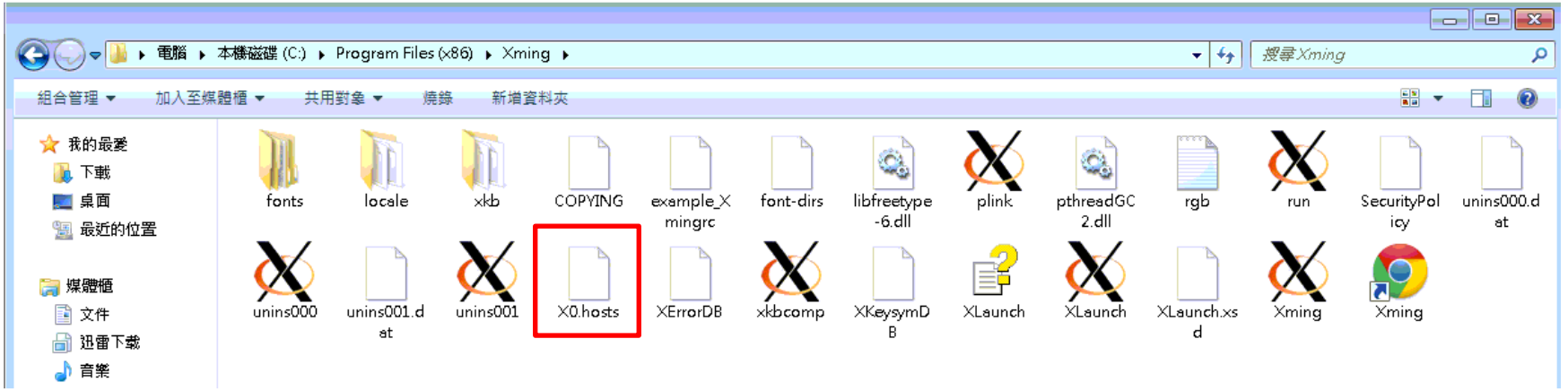
Choose SFTP here

Connecting the server(3/9)



Remember the path you install Xming

Connecting the server(4/9)

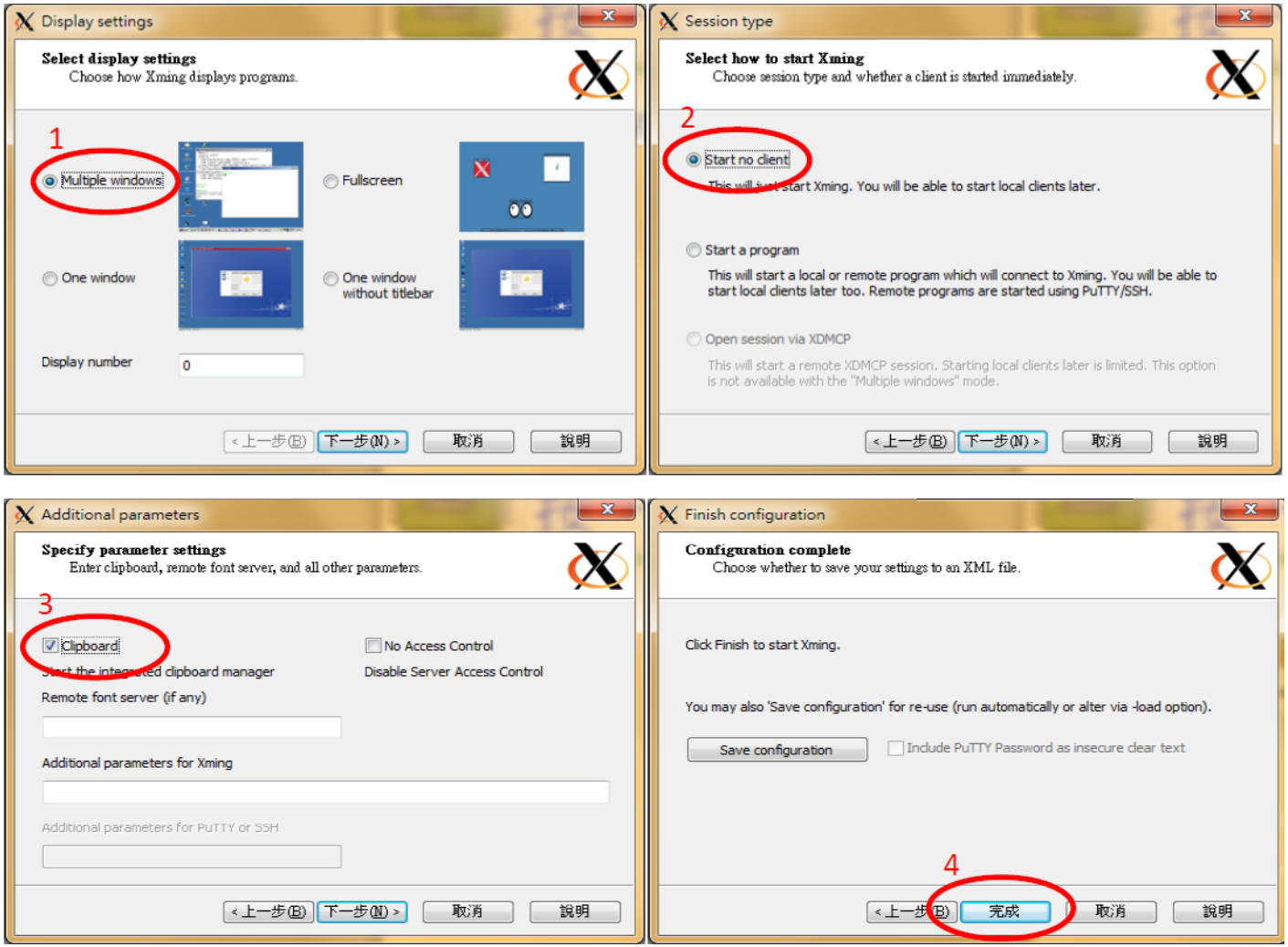


Open X0.host with administrator

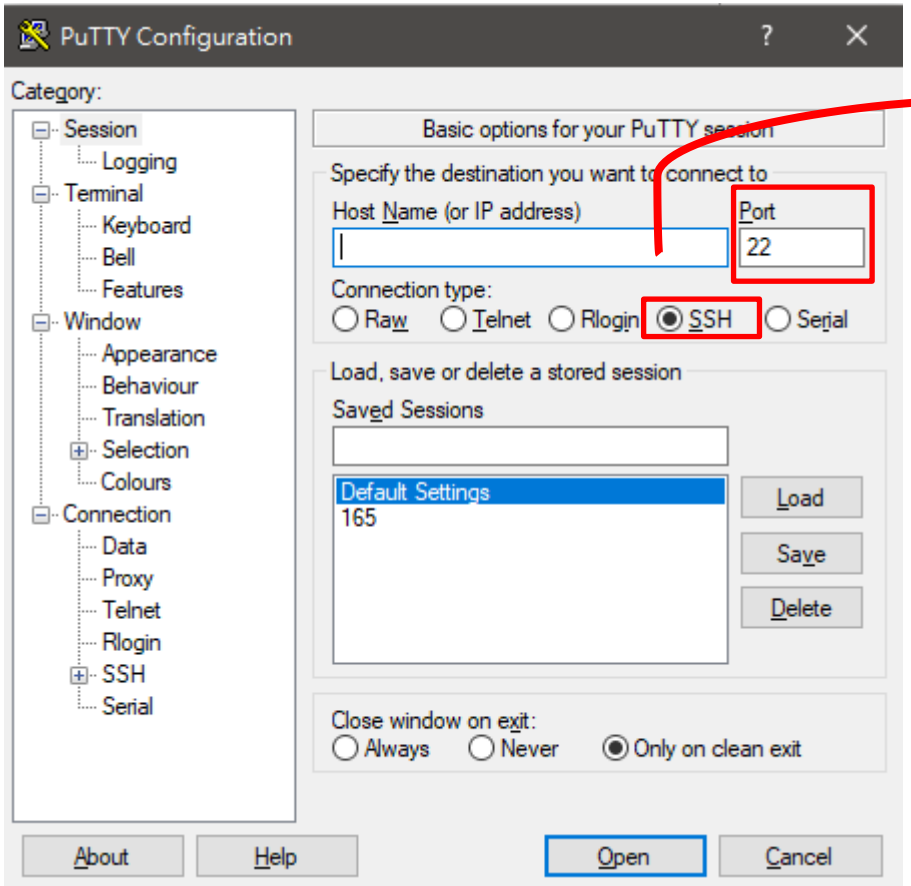
```
1  localhost
2  140.117.157.130
```

Add your IP address under localhost

Connecting the server(5/9)



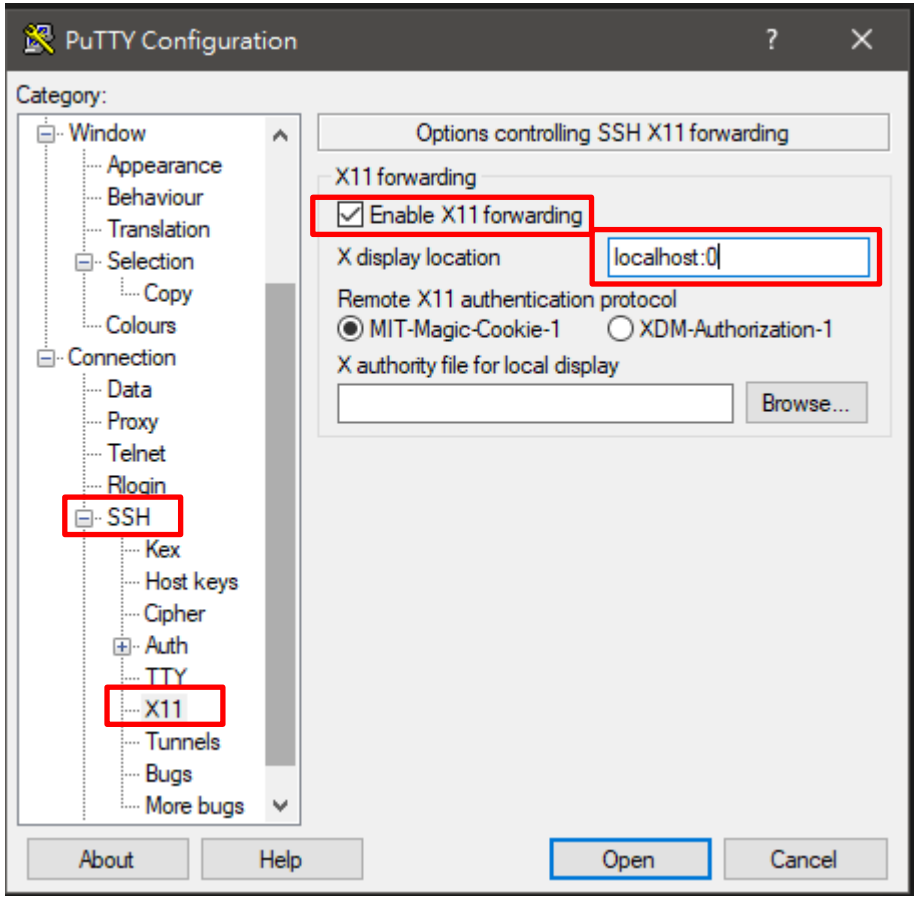
Connecting the server(6/9)



Enter your IP here

We use Port:22 and SSH to client

Connecting the server(7/9)



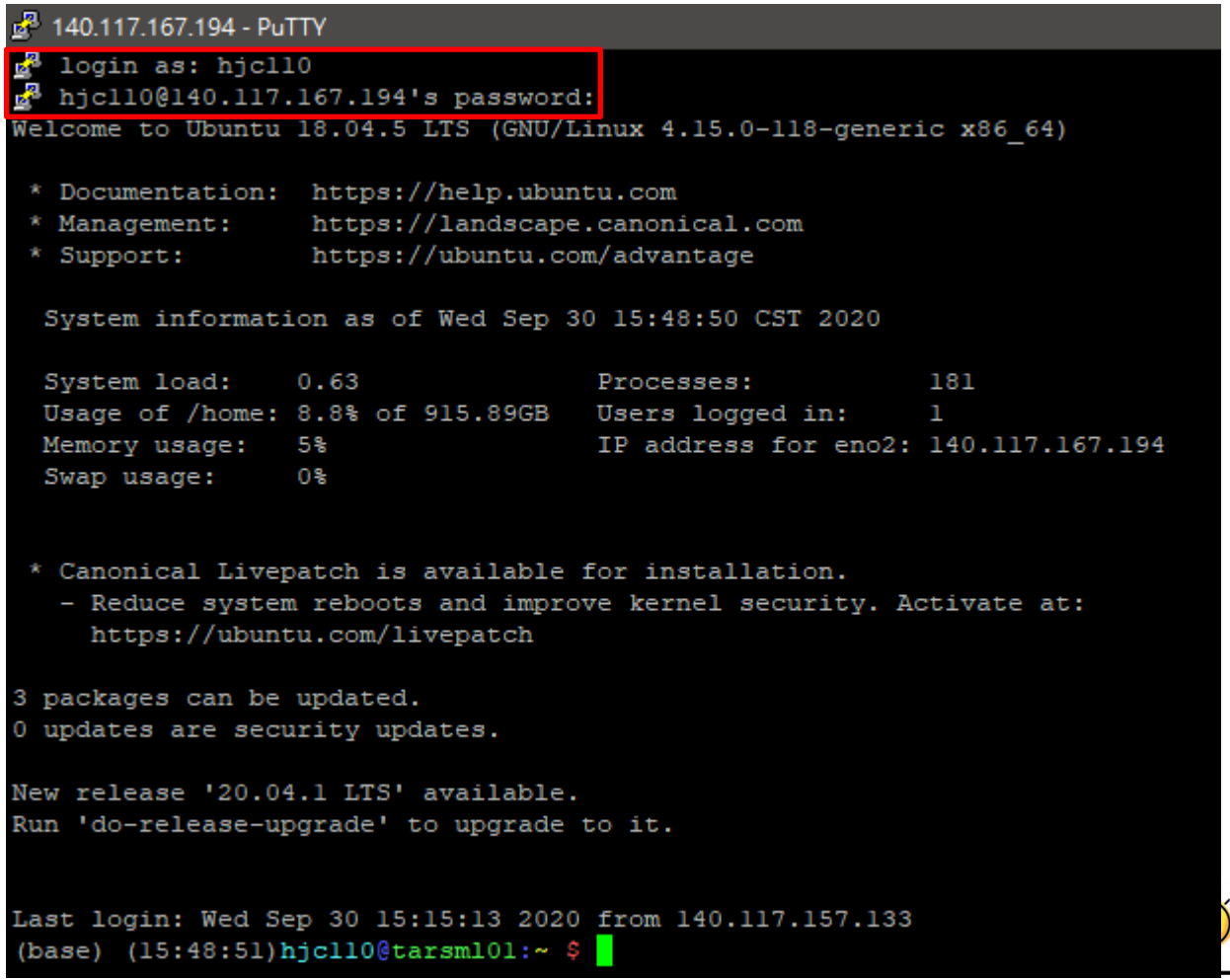
Enable X11 forwarding
&
Enter localhost:0

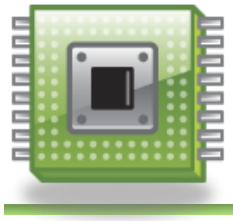
Connecting the server(8/9)

- **Without using NSYSU's static IP, you need to connect VPN.**
  - Even using NSYSU's Wi-Fi, you need to connect, too.
  - The appendix will teach you how to connect VPN.
- **Open Xming before putty.**
- **If you have Antivirus software or Firewall, remember to close it.**

Connecting the server(9/9)

- **Entering the correct account, you will see the window below.**





Basic Linux command

- Change dir
  - $ cd dir
- Show file
  - $ ls
  - $ ls -a (Show hidden file)
- Copy file
  - $ cp fileA ThePathYouWant
  - $ cp -r folderA folder (add -r for copy folder)

Basic Linux command

- **Remove file**
  - $ rm filename (add **-r** for folder)
- **Create folder**
  - $ mkdir folder_name
- **Move file**
  - $ mv pathA pathB

Environment & Requirement(1/2)

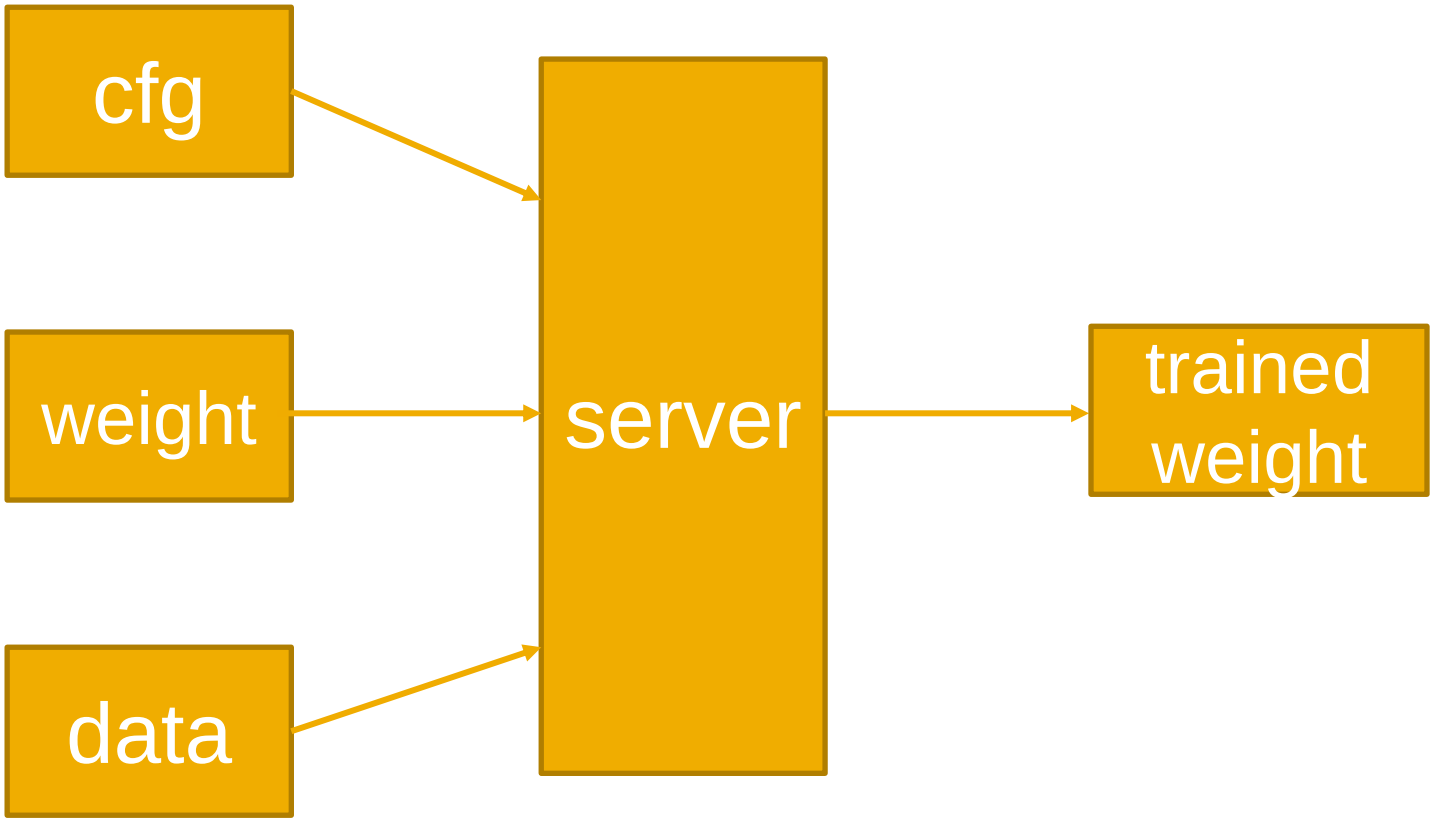
- **Download source code (GitHub or anywhere)**
  - Copy from our server : $ cp -r /home/tars/yolo/ .

- **Read the README.md & requirement.txt**
  - How to build the environment and run code
  - Remember **never use pip**, using **conda** instead.
  - The package name won't be same, check before install.

Environment & Requirement(2/2)

- **Create virtual environment**
  - $ conda create --name env_name python=3.X
- **Activate the environment you create**
  - We have already created an environment for this class
    - $ conda activate yolo
- **Install requirement**
  - Open requirenment.txt and copy the command to install the tool.

Yolov3 training overview



Data overview

- **cfg**
  - Store different configure of yolo.
- **weights**
  - Store the pre_weight or the trained weight.
- **data**
  - Different dataset document.

- **Call different *.py to do different things.**
  - train.py
  - test.py
  - detect.py

Cfg

- **The main difference of different cfg is the number of filters of the last convolution.**

yolov3.cfg	yolov3_IVS_3cls.cfg
80 classes	3 classes
[convolutional] size=1 stride=1 pad=1 filters=255 activation=linear	[convolutional] size=1 stride=1 pad=1 filters=24 activation=linear

- **# of filters = [# of classes + 4(bbox) + 1(confidence score)] * (# of classes)**

Weight

- After training, you will get best.pt, this is the best weight of your training.
- Remember to change the name of best.pt, or it will be overwritten.
- We can use this weight to detect objects.

Data

- We can use different dataset to train.
- The document of dataset is in this folder.
- Yolov3 default use coco dataset.
  - This dataset has 80 classes.
- The chapter below will tell you how to use different dataset.

Darknet-Visualizer

- **Create a figure with different cfg**
  - **$ python3 darknet_visualize.py The_path_of_cfg**
  - **You can see the figure after running the code, or you can download it and open.**

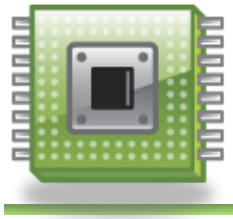
- **Example :**
  - **$ python3 darknet_visualize.py ../cfg/yolov3_IVS_3cls.cfg**
  - **$ python3 darknet_visualize.py ../cfg/yolov3.cfg**

Screen(1/2)

- This command will create a terminal in the server, so we don't need to worry about disconnect.

- What you do in screen is the same as the thing you do at your terminal.
  - After creating the screen, remember to activate the virtual environment.

```
GNU Screen version 4.06.02 (GNU) 23-Oct-17

Copyright (c) 2015-2017 Juergen Weigert, Alexander Naumov, Amadeusz Slawinski
Copyright (c) 2010-2014 Juergen Weigert, Sadrul Habib Chowdhury
Copyright (c) 2008-2009 Juergen Weigert, Michael Schroeder, Micah Cowan, Sadrul Habib Chowdhury
Copyright (c) 1993-2007 Juergen Weigert, Michael Schroeder
Copyright (c) 1987 Oliver Laumann

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 3, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program (see the file COPYING); if not, see http://www.gnu.org/licenses/, or contact Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA  02111-1301  USA.

Send bugreports, fixes, enhancements, t-shirts, money, beer & pizza to screen-devel@gnu.org


Capabilities:
+copy +remote-detach +power-detach +multi-attach +multi-user +font +color-256 +utf8 +rxvt +builtin-telnet
```



Screen(2/2)

- Create a screen
  - $ screen
- Detach screen
  - Ctrl + a, and press d
- Show all the screen you have opened
  - $ screen -ls
- Reconnect to screen
  - $ screen -r id_of_your_screen
- Kill the screen after using
  - When no code running, enter exit to kill screen.

Training with yolov3

- **$ python3 train.py --cfg cfg/xxx.cfg --weights weights/xxx.weights --data data/xxx.data --batch-size x --epochs x --device x**
  - **You should modify the x yourself.**

- **Example :**
  - **$ python3 train.py --cfg cfg/yolov3.cfg --weights weights/yolov3.weights --data data/coco2017.data --batch-size 6 --epochs 2 --device 0**
    - **It will take 1.5hr for one epoch.**

Parameter

- **Batch-size**
  - The bigger batch-size, the higher running speed.
  - It will affect the memory usage of our training.
    - yolov3 : 6
    - yolov3-spp : 4
    - yolov3_IVS : 4
- **Epochs**
  - Number of training times.
  - We need more epochs to upgrade our model's accuracy.

Evaluate the model

- **Run test.py to evaluate the mAP of the model.**
  - **$ python3 test.py --cfg cfg/xxx.cfg --weights weights/best.pt --data data/xxx.data --batch-size x --device x**
- **Example :**
  - **$ python3 test.py --cfg cfg/yolov3_IVS_3cls.cfg --weights weights/IVS.pt --batch-size 4 --data data/IVS_3cls.data --device 0**

Class	Images	Targets	P	R	mAP@0.5	F1
all	8.9e+03	7.17e+04	0.738	0.884	0.881	0.804
Motor-vehicles	8.9e+03	4.79e+04	0.812	0.931	0.933	0.867
Cycles-vehicles	8.9e+03	1.64e+04	0.763	0.876	0.878	0.816
Adult	8.9e+03	7.39e+03	0.639	0.845	0.831	0.727

Object Detection

- **Use the weight you trained to detect object**
  - $ python3 detect.py --cfg cfg/**xxx.cfg** --weights weights/**best.pt** --names data/xxx.**names** --source **xxx**
  - You can see the result in the folder output.

- **Example :**
  - $ python3 detect.py --cfg cfg/yolov3_IVS_3cls.cfg --weights weights/IVS.pt --names data/IVS_3cls.names --source gpt.mp4

How to use different dataset(1/2)

- Create xxx.data
  - class= # of classes of your dataset
  - train= path of your training images' .txt
  - valid= path of your valid images' .txt
  - names= path of your class data
- Create xxx.names
  - Motor-vehicles
  - Cycles-vehicles
  - Adult
    - Writing all the classes the dataset has into the file.

How to use different dataset(2/2)

- Create **train.txt**
  - This txt contain the path of all your train data.
- Create **val.txt**
  - This txt contain the path of all your valid data.
- Create **test.txt**
  - This txt contain the path of all your test data.

Create your own dataset(1/2)

- Images

- Labels

```xml
<?xml version="1.0" encoding="UTF-8"?>
<annotation>
  <filename>1_40_00_00001</filename>
  <size>
    <width>1920</width>
    <height>1080</height>
  </size>
  <object>
    <category>Transportation</category>
    <type>Motor-vehicles</type>
    <attribution>Sport Utility Vehicle</attribution>
    <bndbox>
      <xmin>371</xmin>
      <ymin>509</ymin>
      <xmax>592</xmax>
      <ymax>698</ymax>
    </bndbox>
  </object>
  <object>
    <category>Transportation</category>
    <type>Motor-vehicles</type>
    <attribution>Sport Utility Vehicle</attribution>
    <bndbox>
      <xmin>302</xmin>
      <ymin>544</ymin>
      <xmax>342</xmax>
      <ymax>584</ymax>
    </bndbox>
  </object>
  <object>
```

*See appendix for the use of labelimg

Create your own dataset (2/2)

- **The dataset's folder name architecture should be like this :**
  - Dataset name (IVS_3cls)
    - images
      - train
      - val
      - test
    - labels
      - train
      - val
      - test

Flow overview

- Divide the images to 3 folders
  - train
  - test
  - val
- Divide the XML to the same 3 folders
- Transfer the XML files to txt files

Example : IVS_3cls(1/6)

- **This dataset only has three classes**
  - **Motor-vehicles**
  - **Cycles-vehicles**
  - **Adult**

- **We need to divide the data ourself.**
  - **It totally has about 89000 pictures.**

Example : IVS_3cls(2/6)

- Copy IVS_3cls from server
  - $ cp -r /home/tars/IVS_3cls /home/Account_name
  - There are two folders in IVS_3cls folder.
- Create images & labels folder
  - $ cd ~/IVS_3cls/
  - $ mkdir images
  - $ mkdir labels
- Open yolo/pre_process/IMG_div.py
  - We use this code to divide our images.
  - The rate of train : test : valid preset here is 8:1:1
    - python3 IMG_div.py

Example : IVS_3cls(3/6)

- Open yolo/pre_process/get_dir.py
  - This code will create a txt file with all data's path .
  - Modify the data path and the output file name.
    - $ python3 get_dir.py

- Open yolo/pre_process/XML_div.py
  - This code will divide the XML files according to the images you divided.
  - Modify the *.txt file yourself.
    - $ python3 XML_div.py

Example : IVS_3cls(4/6)

- **Transfer XML files to txt type.**
  - **$ cd XmlToTxt/**
  - **$ python3 xmltotxt.py -xml ~/IVS_3cls/train/ -out ~/IVS_3cls/labels/train**
- **Make sure the type images and labels all conformed to the shape below.**
  - **EX :**
  - **/hjc110/IVS_3cls**
    - **images**
    - **labels**

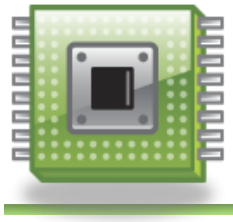
Example : IVS_3cls(5/6)

- **Start training with IVS_3cls**
  - **$ python3 train.py --cfg cfg/yolov3_IVS_3cls.cfg --weights weights/yolov3.weights --data data/IVS_3cls.data --batch-size 4 --epochs 5 --device 0**
  - **It will take about an hour for one epochs.**
- **Go to data/ and create IVS_3cls_test.data**
  - **classes = 3**
  - **valid = data/test1.txt**
  - **names = data/IVS_3cls.names**

Example : IVS_3cls(6/6)

- **Evaluate model**
  - **$ python3 test.py --cfg cfg/yolov3_IVS_3cls.cfg --weights weights/best.pt –batch-size 4 --data data/IVS_3cls_test.data --device 0**

Class	Images	Targets	P	R	mAP@0.5	F1:
all	8.9e+03	7.16e+04	0.745	0.884	0.882	0.808
Motor-vehicles	8.9e+03	4.77e+04	0.816	0.927	0.931	0.868
Cycles-vehicles	8.9e+03	1.63e+04	0.768	0.879	0.882	0.819
Adult	8.9e+03	7.59e+03	0.653	0.845	0.834	0.737



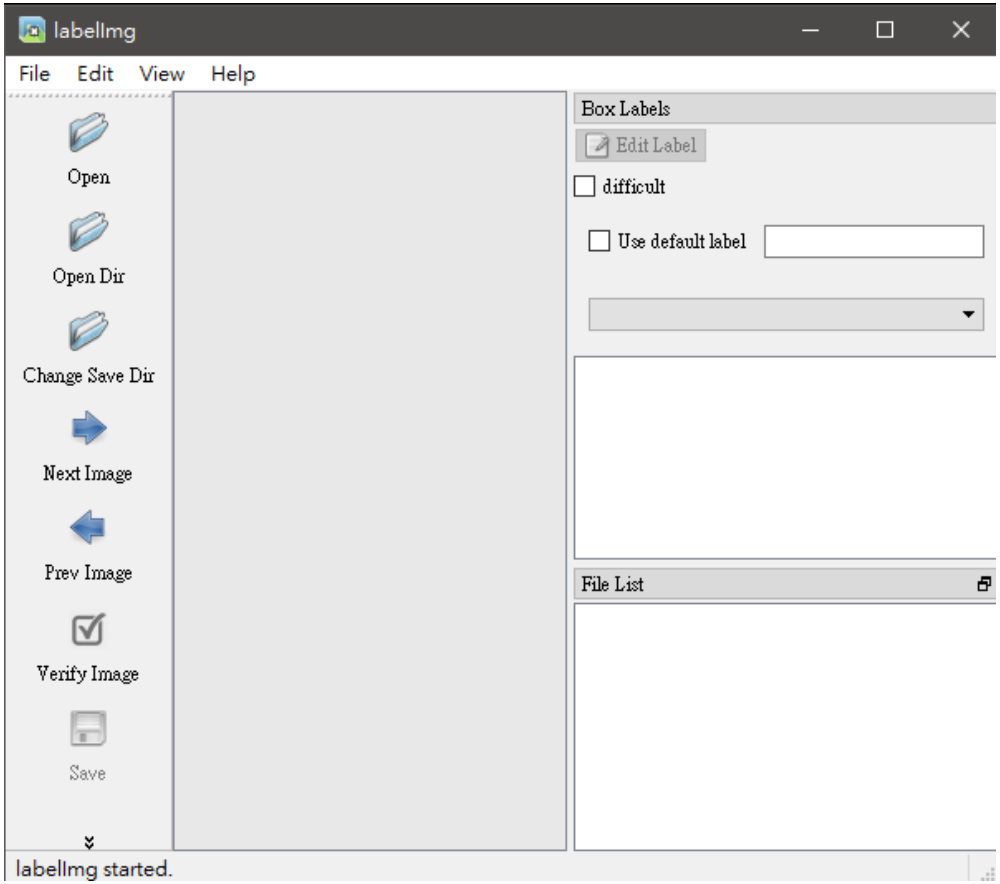
APPENDIX

Label_Img(1/3)

- Un-zip labelImg.rar
- Open Anaconda Prompt
- Change dir to the labelImg folder
- Activate your virtual environment
- Install requirement
  - $ conda install -y pyqt=5
  - $ conda install -y -c anaconda lxml
  - $ pyrcc5 -o libs/resources.py resources.qrc

Label_Img(2/3)

- Open the tool
  - $ python labelImg.py

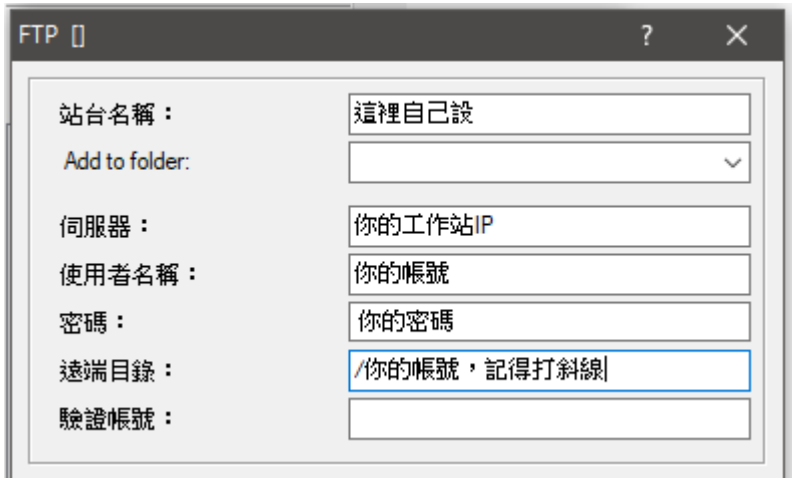


Label_Img (3/3)

Ctrl + u	Load all of the images from a directory
Ctrl + r	Change the default annotation target dir
Ctrl + s	Save
Ctrl + d	Copy the current label and rect box
Space	Flag the current image as verified
w	Create a rect box
d	Next image
a	Previous image
del	Delete the selected rect box
Ctrl++	Zoom in
Ctrl--	Zoom out
↑→↓←	Keyboard arrows to move selected rect box

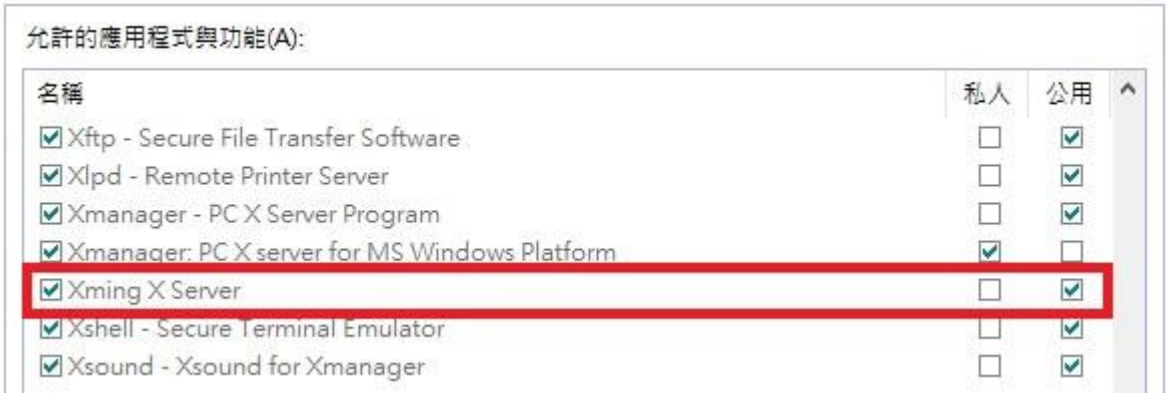
工作站基礎Debug(1/2)

- VPN使用教學連結
- Xming使用前記得在X0.host的localhost下面加入自己的IP(140.117.XXX.XXX)
- Putty連線前記得在X11把Enable打勾以及輸入localhost:0
- 開啟順序為Xming→Putty
- PsPad設定如圖



工作站基礎Debug(2/2)

- 防火牆設定：Windows搜尋防火牆→允許應用程式通過防火牆→Xming X Server



- 有使用防毒軟體的同學，建議在連線工作站時關閉