fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

PART1 | Vision and YOLO In practice perspective

Introduction – TK Profile

Chapter 1. AI

Chapter 2. Vision

Chapter 3. Object Recognition

Chapter 4. Object Detection

Chapter 5. YOLO

Chapter 6. YOLO in practice

Chapter 7. AI Vision Limitation and Next

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

TK Profile

Blog
Profile
Specialty

Deep Learning Study Methods

- Start small and grow
- Learning by doing
- Principle
- Don’t give up

Contents

AI
- Intro, History
- Preparations
- Tools. Development environment (GPU, Pip, Anaconda, Windows or Ubuntu)
- Python introduction (numpy, shape ...) 실습
- AI framework introduction (Tensorflow, Keras, PyTorch)
- NN forward, backward propagation
- Activation Function, Overfitting solution, Training Performance solution

Vision
- History
- Image, Convolution Filter.

Object Recognition
- History
- LeNet, AlexNet, VGG16 실습
- GoogleLeNet, ResNet 실습
- Transfer Learning. 실습

Object Detection
- RoI, NMS, mAP, Selective Search, Bounding-box regressor 실습
- R-CNN, Fast R-CNN, Faster R-CNN, SDD, YOLOv1 실습

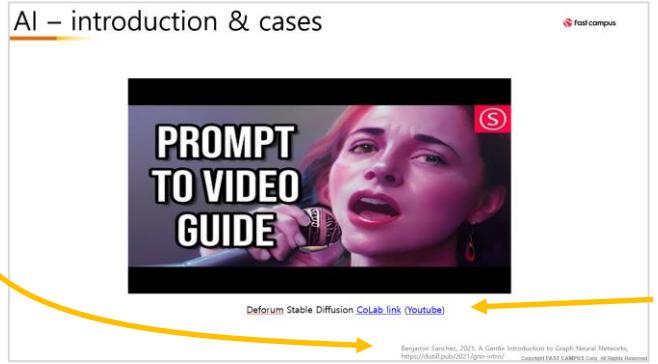
YOLO
- History
- Paper review
- Network architecture
- YOLO

YOLO in practice
- Introduction, Installation
- Analysis YOLO core in source code
- Prepare dataset (Construction in google image) 실습
- Labeling dataset 실습
- Train dataset
- Predict data
- Hyperparameter Tuning
- Serving custom model using Flask 실습

AI Vision Limitation and Next
- Paper review (A Path Towards Autonomous Machine Intelligence)
- Conclusion

User Skill Minimum Level = {
"os": "windows, ubuntu basic",
"python": "array, dict, func, file, pip, numpy concept",
"AI framework": "keras, pytorch, tf, basic concept",
"dataset": "mnist, coco, VOC format",
"math": "linear algebra, diff, numerical analysis",
"Tools": "git clone, colab, ubuntu, anaconda, docker",
"English": "basic",
"Research": "basic"
}

사용된 Reference 표시 = {Author, date, title, company, URL}



Link

Diving pathway

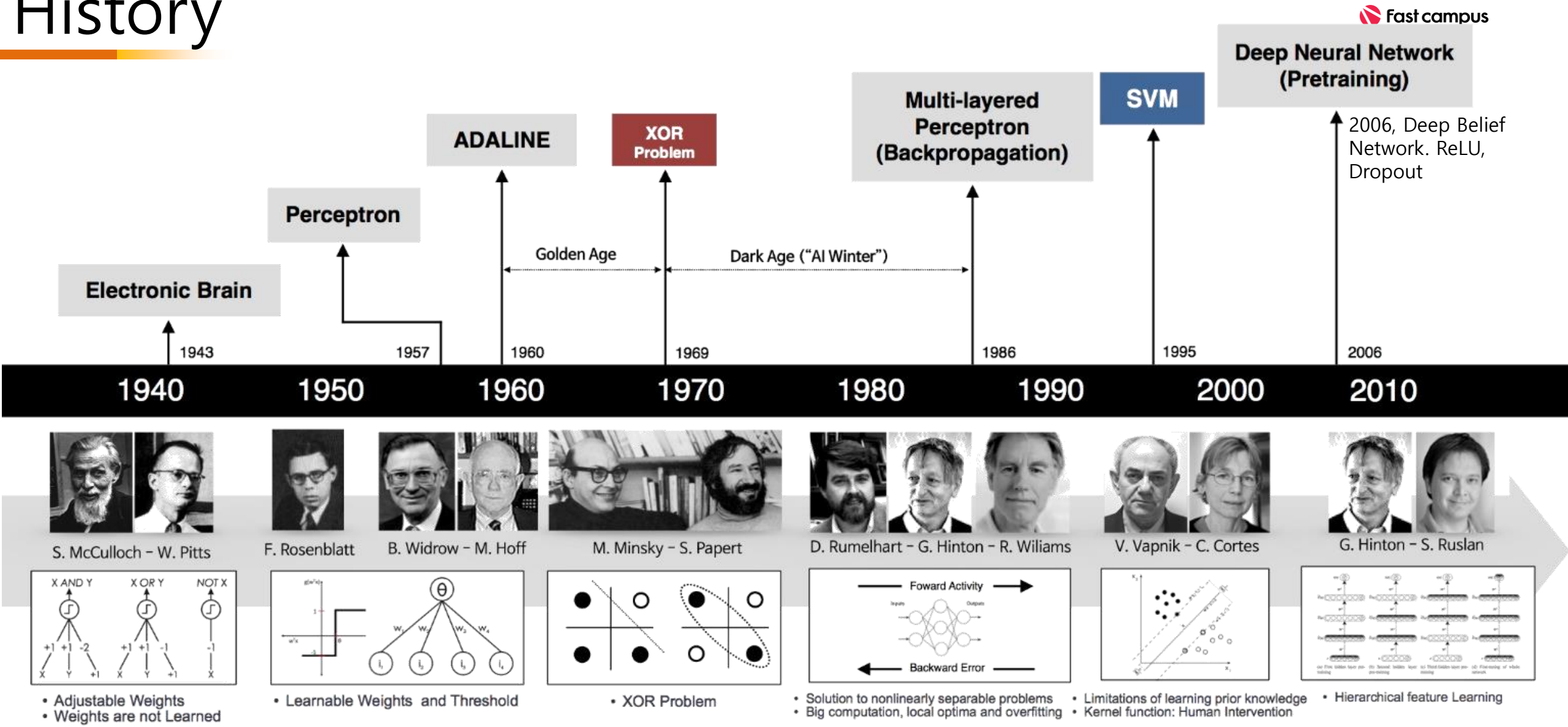
- Concept & Background
- YOLO Mechanism
- Practice

Copyright FAST CAMPUS Corp. All Rights Reserved

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

AI

History



Link

Yann LeCun. LeNet5 using CNN

Copyright FAST CAMPUS Corp. All Rights Reserved

History

Deep Belief Networks (DBN) Hinton, G. E, Osindero, S., and Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18:1527-1554.

Convolutional neural networks running on GPUs (2012) Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton, Advances in Neural Information Processing Systems 2012

Deep CNN, GPU, ReLU, Dropout, Deep Belief Network

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Link

2013. Google acquired DNN

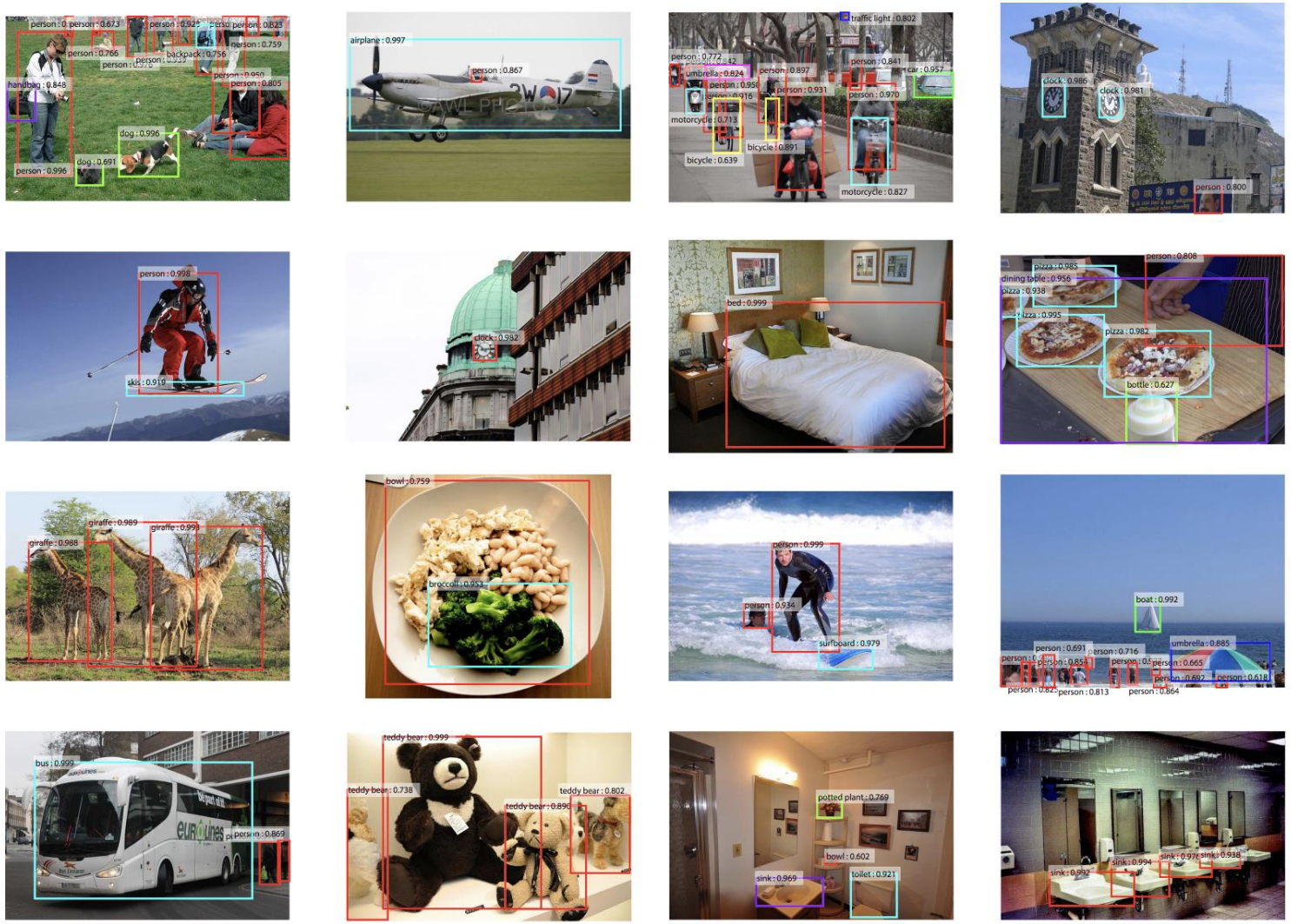
Google images

Yann LeCun.
Chief AI Scientist for Facebook AI Research (FAIR)

Facebook AI Research

Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases



Jason Brownlee, 2019, 9 Applications of Deep Learning for Computer Vision, machine learning mastery

Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases

Link

Image Colorization with U-Net and GAN

Jason Brownlee, 2019, 9 Applications of Deep Learning for Computer Vision, machine learning mastery

Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases

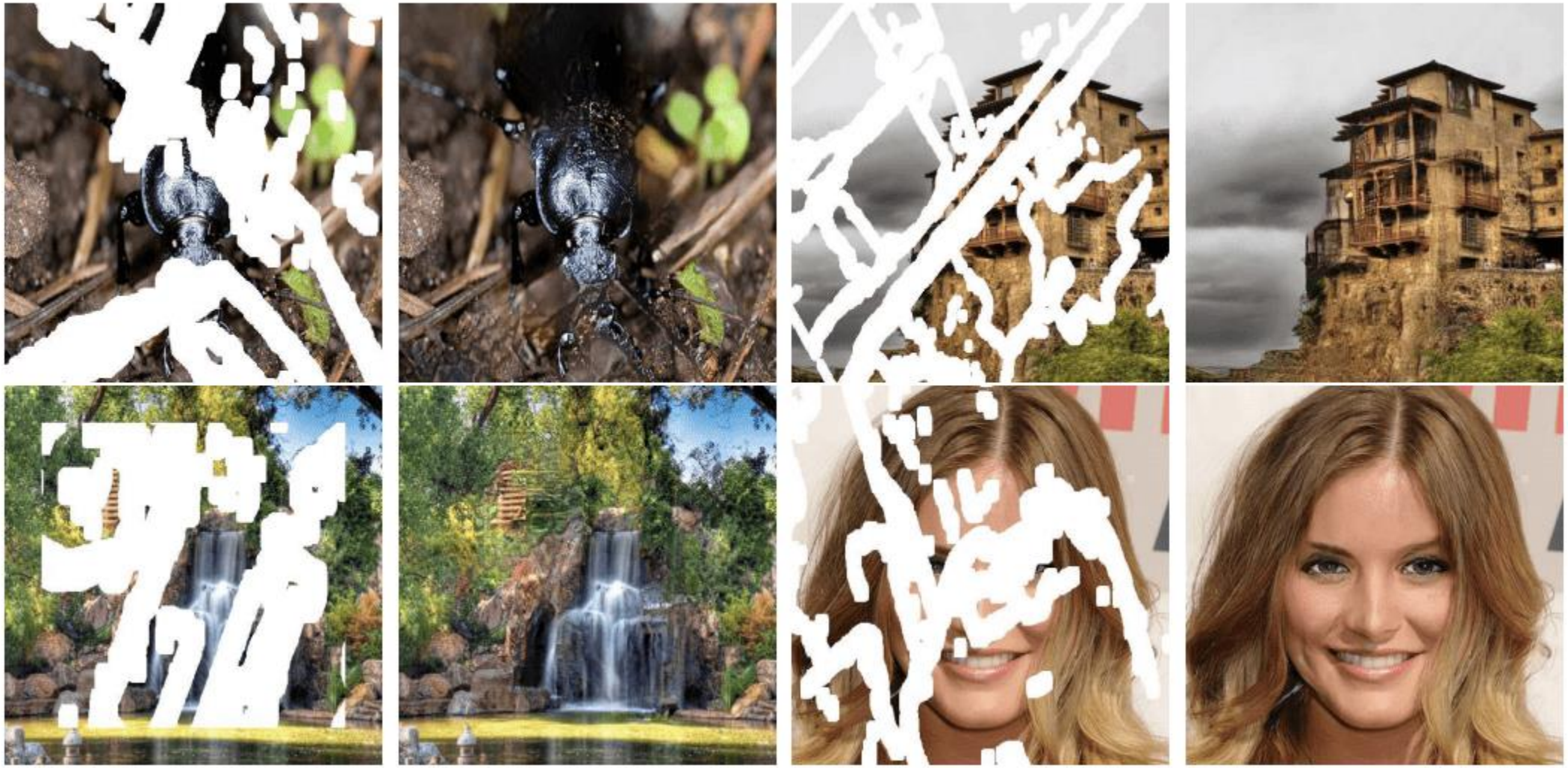
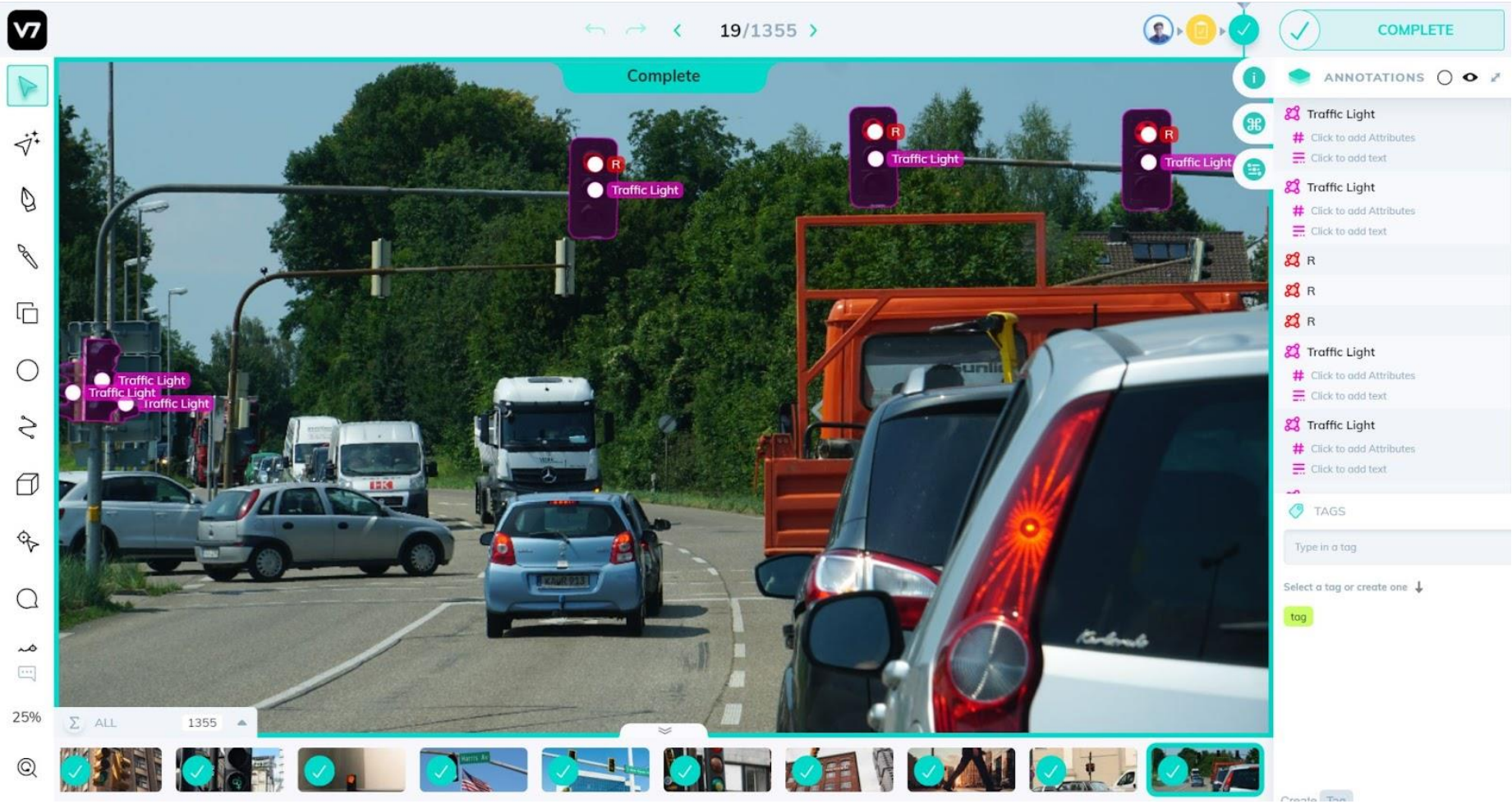


Image inpainting

Jason Brownlee, 2019, 9 Applications of Deep Learning for Computer Vision, machine learning mastery

Copyright FAST CAMPUS Corp. All Rights Reserved

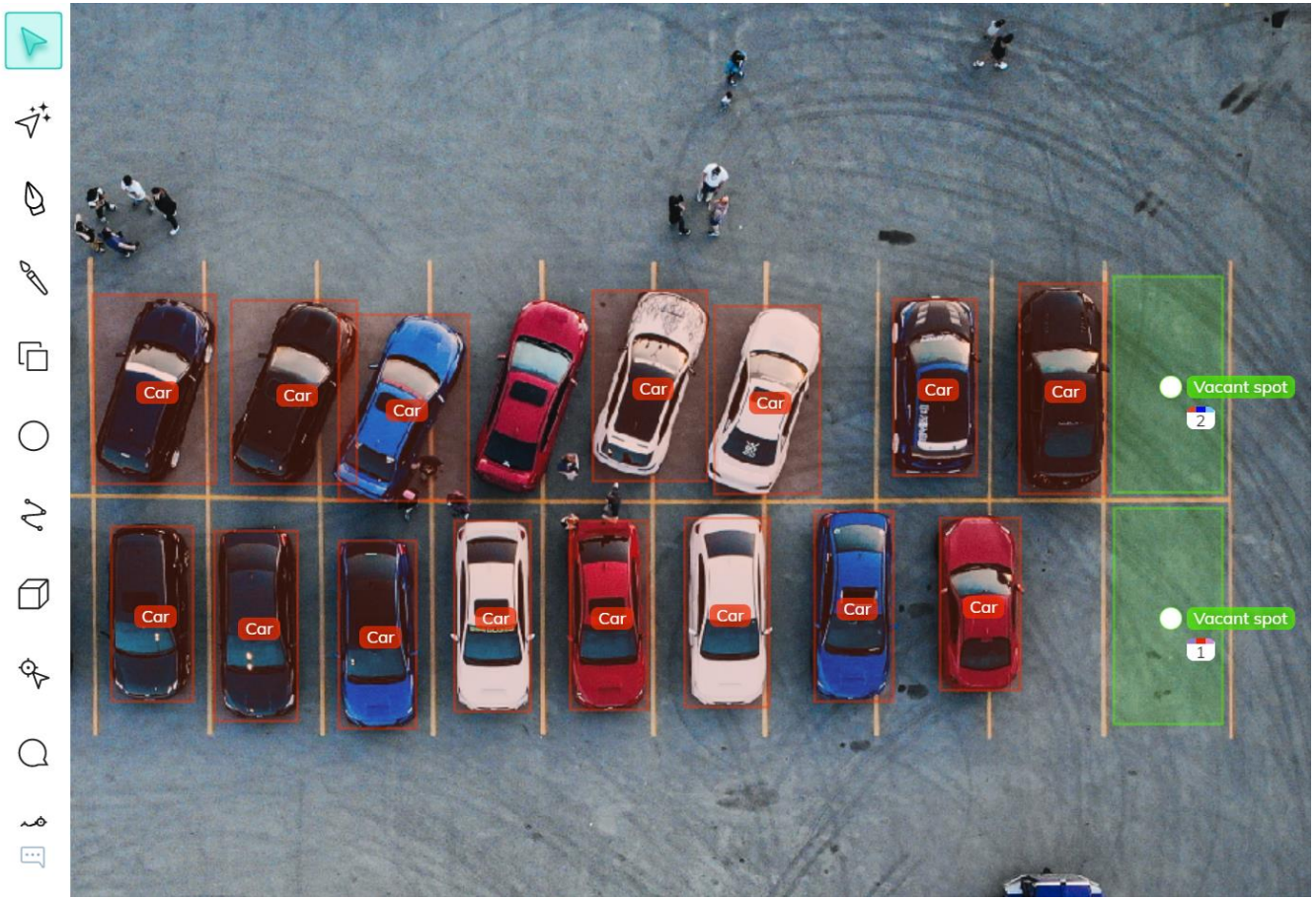
AI – introduction & cases



Alberto Rizzoli, 2022, 27+ Most Popular Computer Vision Applications and Use Cases in 2022, V7

Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases



Alberto Rizzoli, 2022, 27+ Most Popular Computer Vision Applications and Use Cases in 2022, V7

Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases

```
280 FI
New Bleecker St.
York, NY 10014
Tel: (212) 9 Elizabet
Tbl 2/1
Chk 2334
Jul03' 12 02:34PM
Gst 1
1 Crab Special
1 $2.5 PBR
21.95
2.50
1 $4 Stella
4.00
food
21.95
liquor
6.50
Tax
2.53
04:02PM Total Due
30.98
Please be advised: unless noted
above, gratuity is NOT included
Thank you for dining with us!
```

[Use Google Cloud Vision API to process invoices and receipts]

[Tesseract-OCR]

Alberto Rizzoli, 2022, 27+ Most Popular Computer Vision Applications and Use Cases in 2022, V7

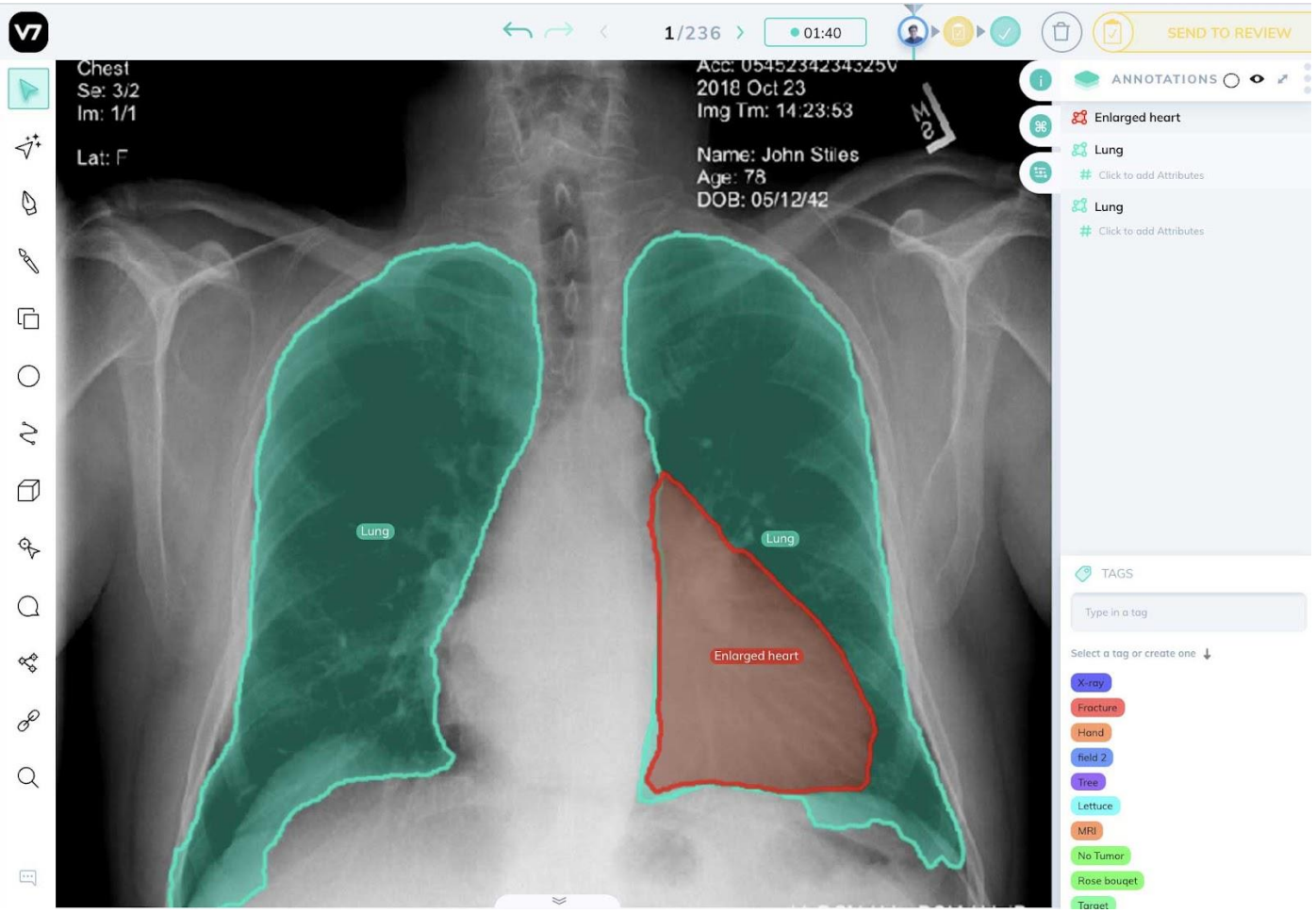
Copyright FAST CAMPUS Corp. All Rights Reserved

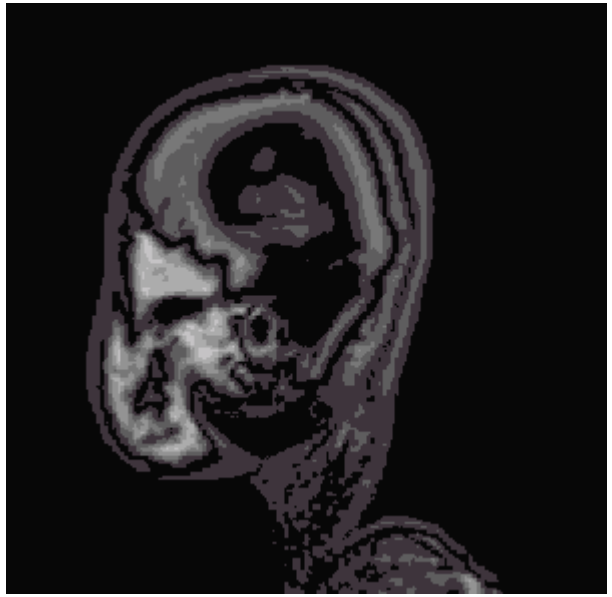
AI – introduction & cases

Alberto Rizzoli, 2022, 27+ Most Popular Computer Vision Applications and Use Cases in 2022, V7

Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases





[CRF-RNN for Semantic Image Segmentation]

Conditional Random Fields as Recurrent Neural Networks.

[Detectron2] [CoLab]

Alberto Rizzoli, 2022, 27+ Most Popular Computer Vision Applications and Use Cases in 2022, V7

Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases

YOLACT (You Only Look At CoefficienTs)
Detectron

Alberto Rizzoli, 2022, 27+ Most Popular Computer Vision Applications and Use Cases in 2022, V7

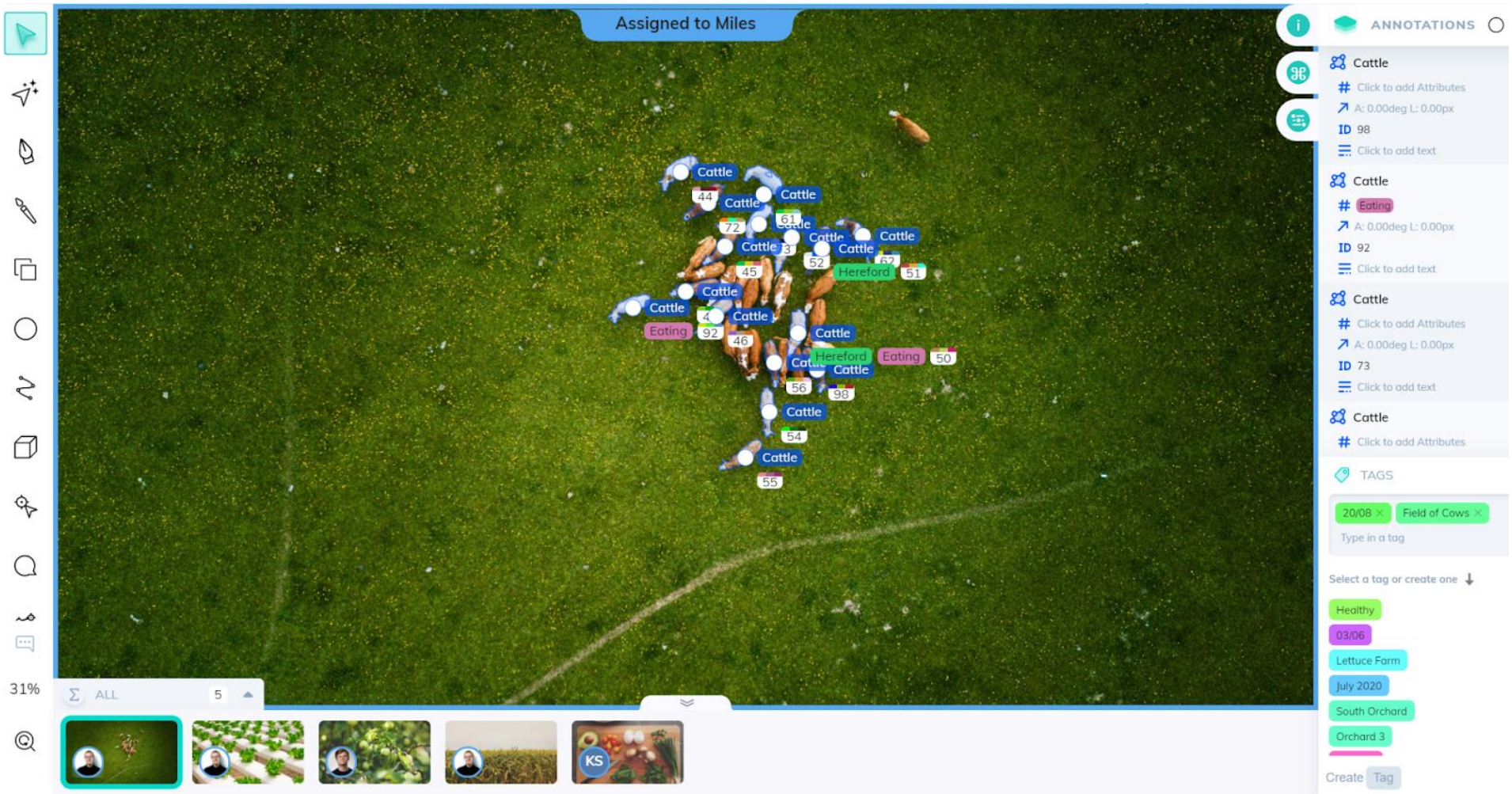
Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases

Alberto Rizzoli, 2022, 27+ Most Popular Computer Vision Applications and Use Cases in 2022, V7

Copyright FAST CAMPUS Corp. All Rights Reserved

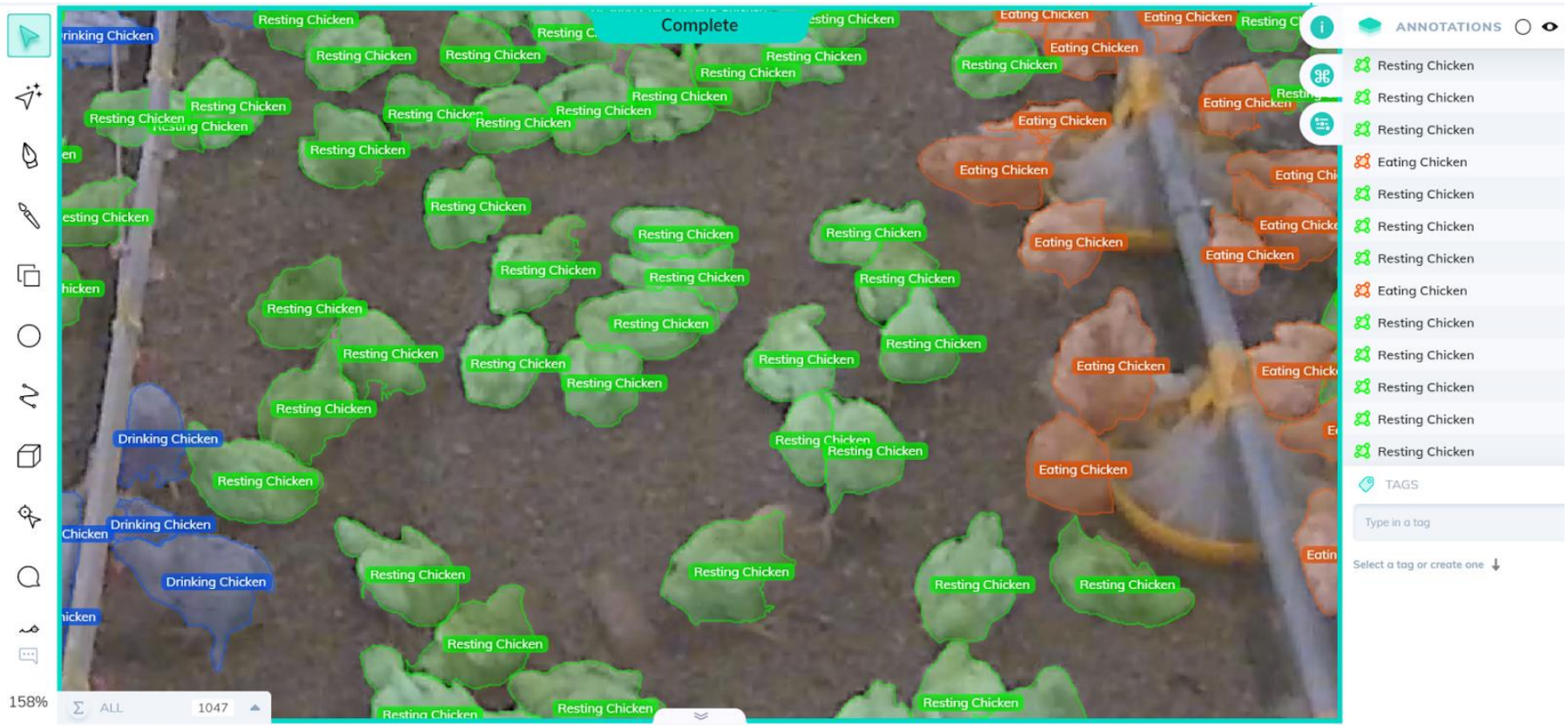
AI – introduction & cases



Alberto Rizzoli, 2022, 27+ Most Popular Computer Vision Applications and Use Cases in 2022, V7

Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases

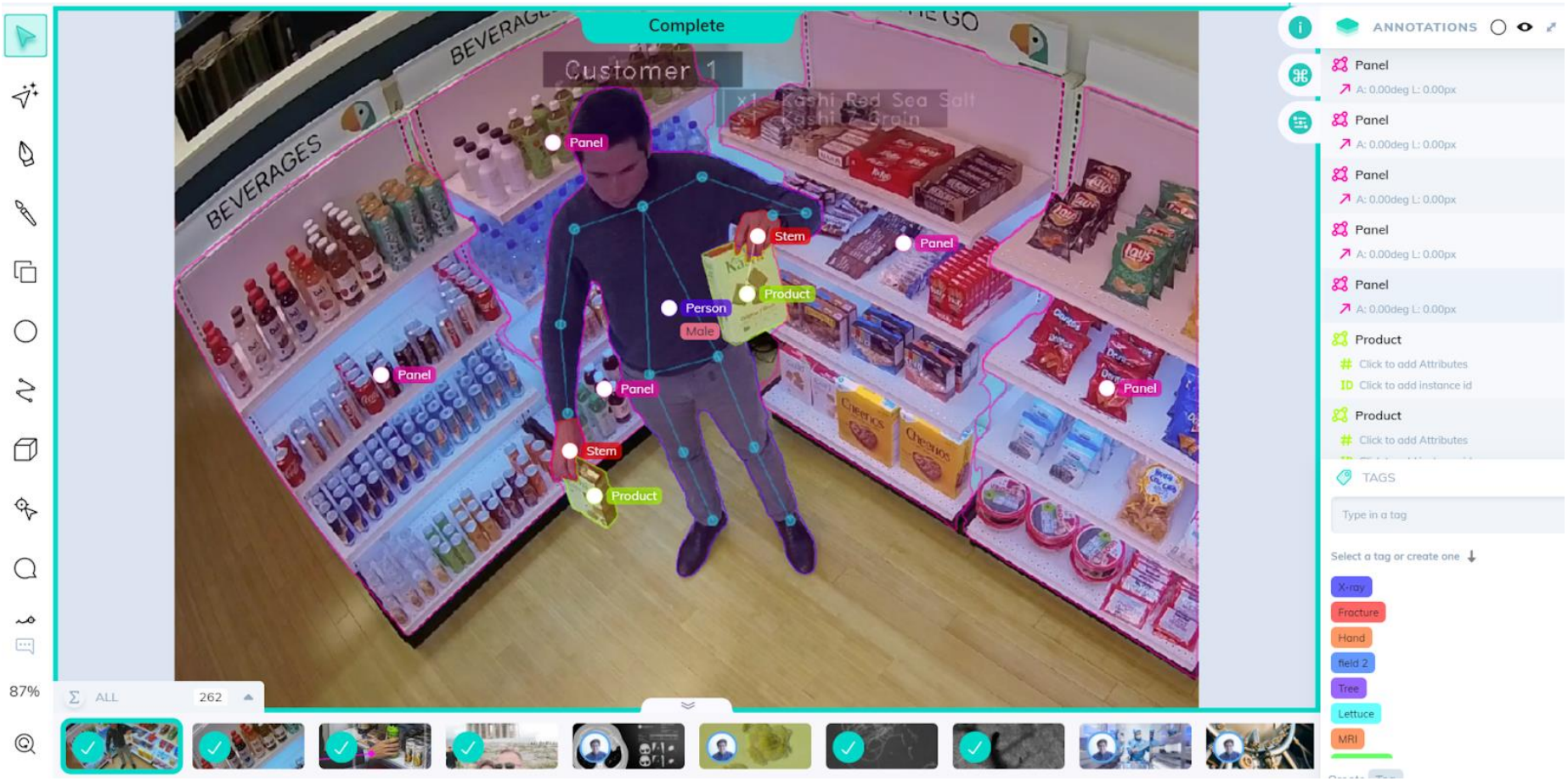


Vision Transformer (ViT)
[Link]

Alberto Rizzoli, 2022, 27+ Most Popular Computer Vision Applications and Use Cases in 2022, V7

Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases

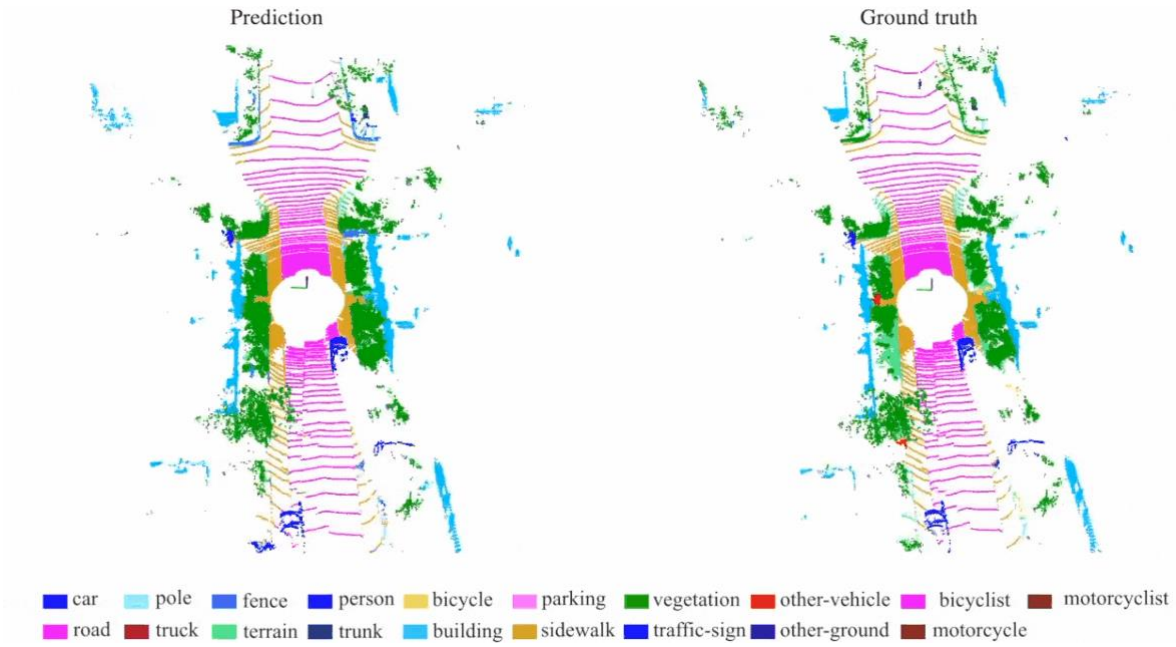


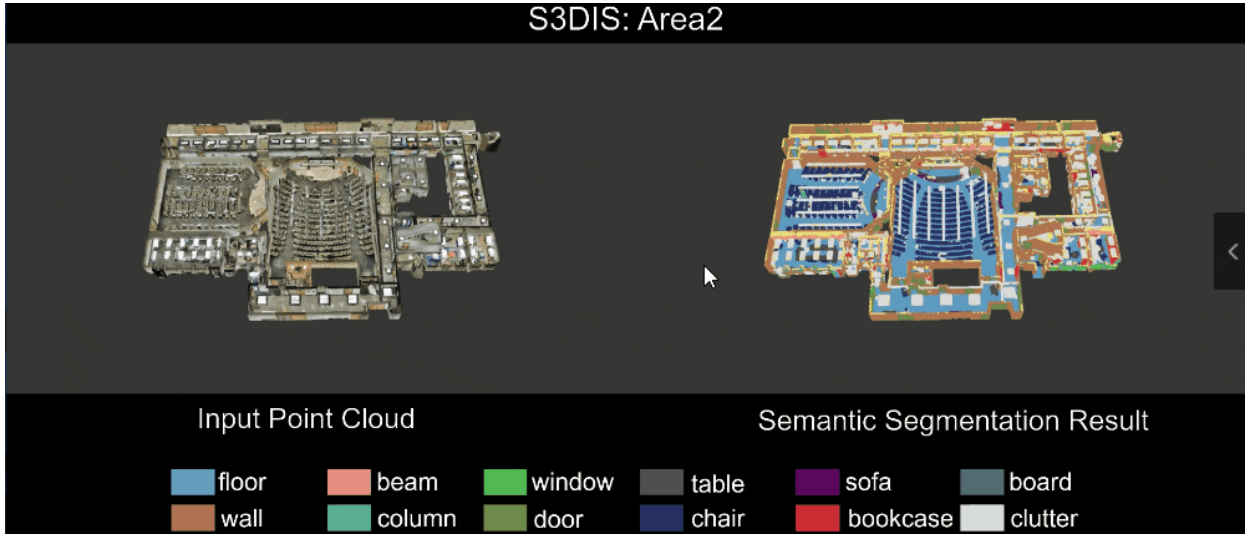
[OpenPose]

Alberto Rizzoli, 2022, 27+ Most Popular Computer Vision Applications and Use Cases in 2022, V7

Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases

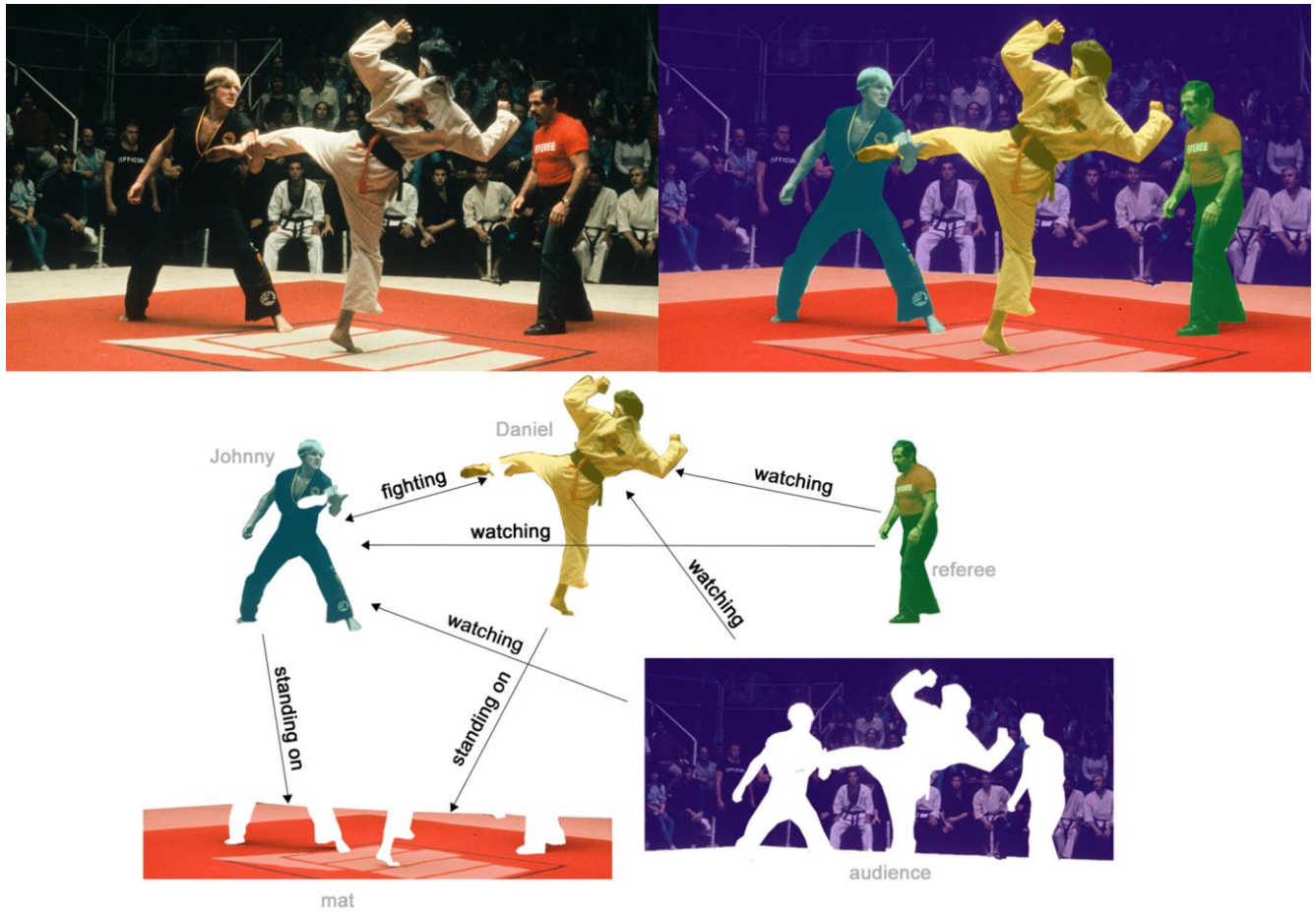




3D semantic segmentation (Random sampling and an effective local feature aggregator). link

Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases



GCN (Graph Convolutional Network) CoLab link

Benjamin Sanchez, 2021, A Gentle Introduction to Graph Neural Networks, https://distill.pub/2021/gnn-intro/

Copyright FAST CAMPUS Corp. All Rights Reserved

AI – introduction & cases

Deforum Stable Diffusion CoLab link (Youtube)

Benjamin Sanchez, 2021, A Gentle Introduction to Graph Neural Networks, https://distill.pub/2021/gnn-intro/

Copyright FAST CAMPUS Corp. All Rights Reserved

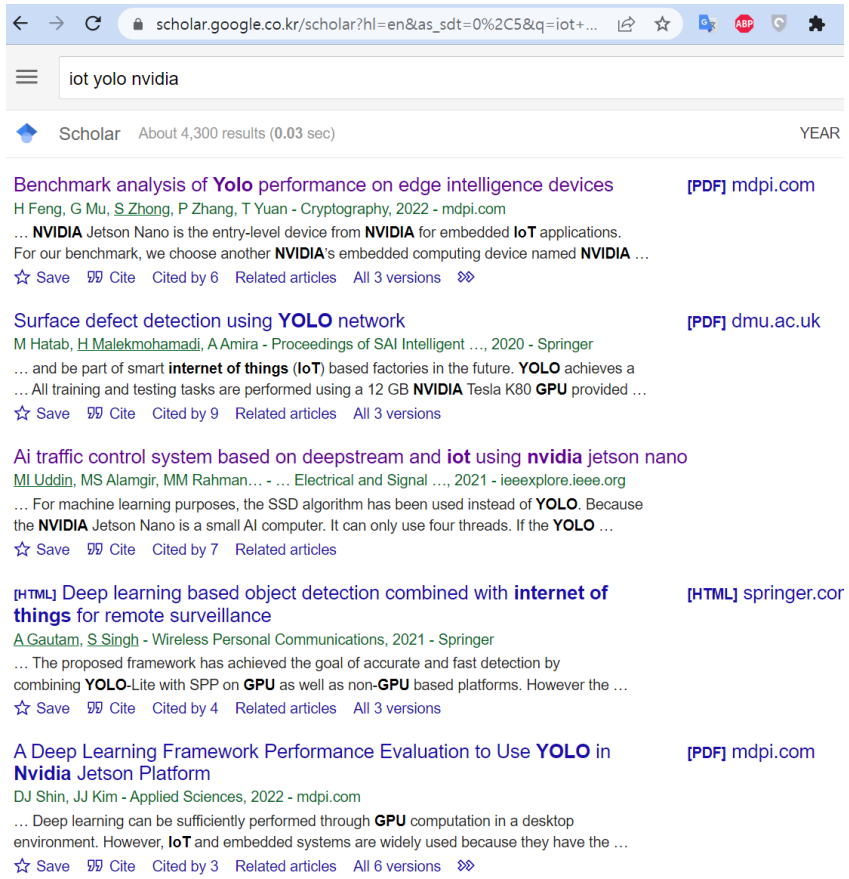
AI – introduction & cases

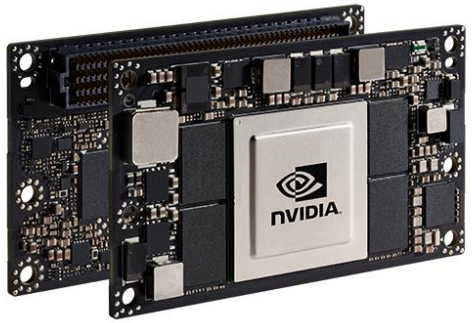
Fast campus

[DL papers in Scholar]

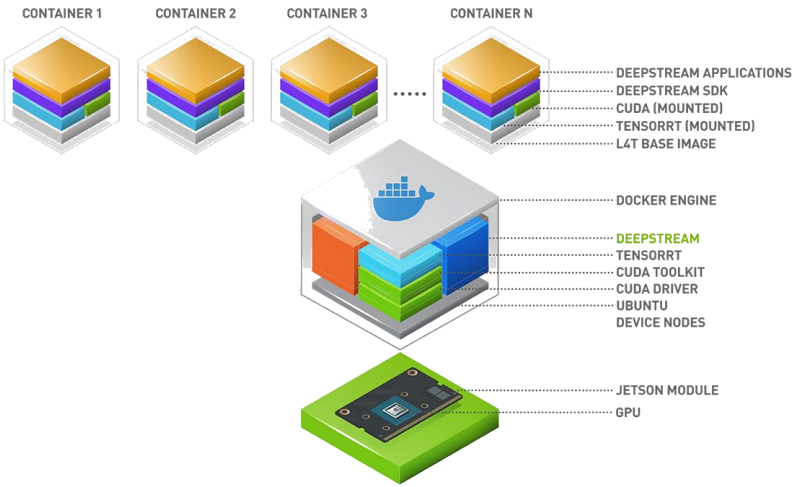
[Example]

[Example]





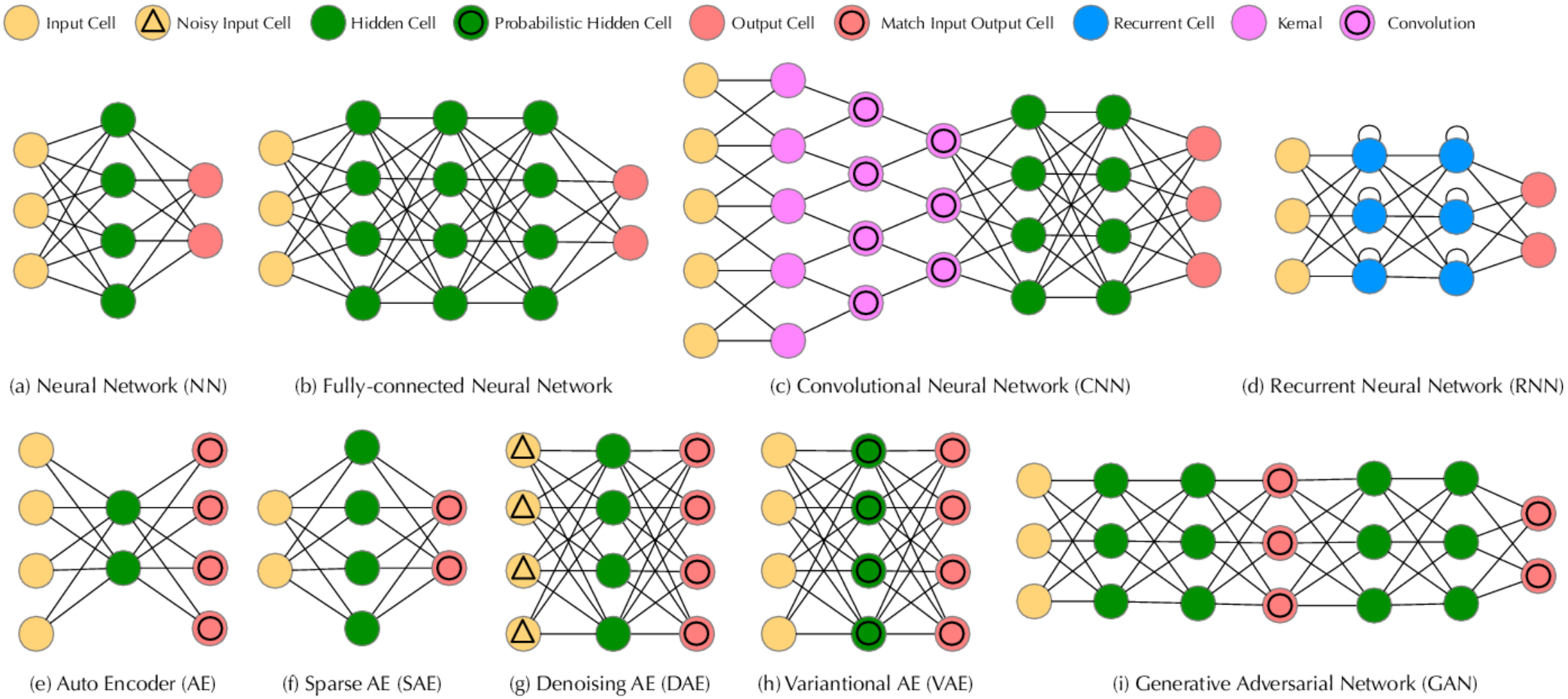
[DeepStream]



Alberto Rizzoli, 2022, 27+ Most Popular Computer Vision Applications and Use Cases in 2022, V7

Copyright FAST CAMPUS Corp. All Rights Reserved

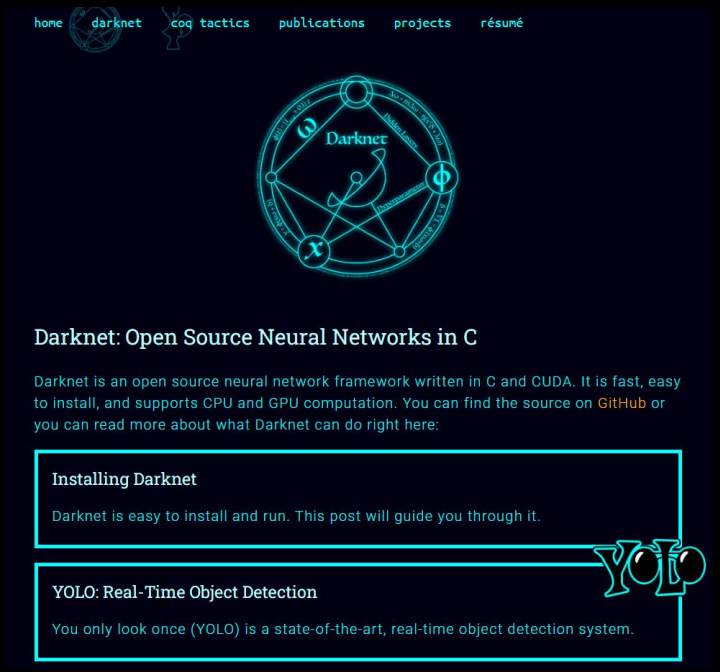
AI – deep learning family



Saravanan Thirumuruganathan, N. Tang, M. Ouzzani, 2018, Data Curation with Deep Learning [Vision]: Towards Self Driving Data Curation, ArXiv

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO



DarkNet

link

Link

Tools

Dive

1. Install NVIDIA Driver
2. Install CUDA 11.3
3. Install cuDNN 11.x and path setting
4. Install python (3.9.12)
5. cmd *관리자 권한으로 실행해야 함
6. pip install tensorflow (v2)
7. pip3 install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu113
8. pip install virtualenv
9. Install vscode
10. Install sublime
11. https://github.com/ultralytics/yolov5
12. pip install jupyterlab
13. pip install ipywidgets

PyTorch Build	Stable (1.12.1)		Preview (Nightly)		LTS (1.8.2)
Your OS	Linux		Mac		Windows
Package	Conda	Pip	LibTorch		Source
Language	Python		C++ / Java		
Compute Platform	CUDA 10.2	CUDA 11.3	CUDA 11.6	~~ROCm 5.1.1~~	CPU
Run this Command:	pip3 install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu113				

pip install torch==1.11.0+cu113 torchvision==0.12.0+cu113 torchaudio==0.11.0 --extra-index-upytorch.org/whl/cu113rl https://download.

How To Set Up A Python Virtual Environment On Windows 10

https://www.liquidweb.com/kb/how-to-setup-a-python-virtual-environment-on-windows-10/

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO tools - implementation

linux

```
git clone https://github.com/pjreddie/darknet.git
cd darknet
make
```

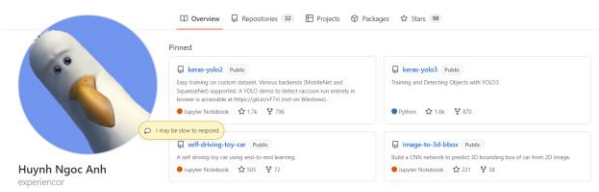
Windows

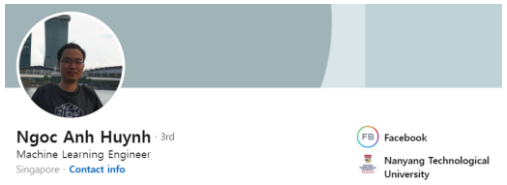
https://github.com/AlexeyAB/darknet

Keras

https://github.com/experiencor/keras-yolo2

https://github.com/experiencor/keras-yolo3



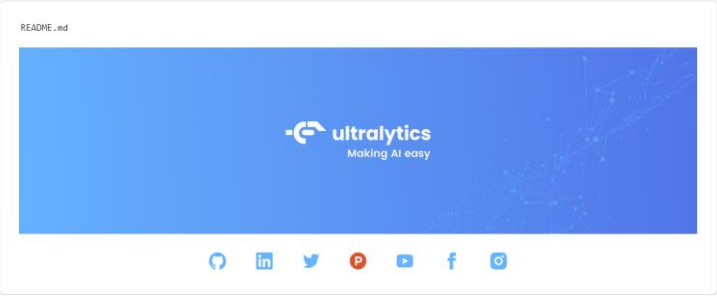


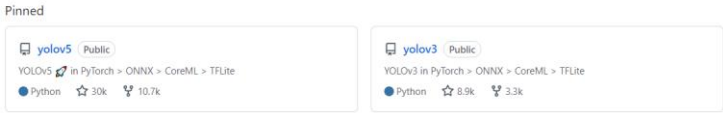
PyTorch

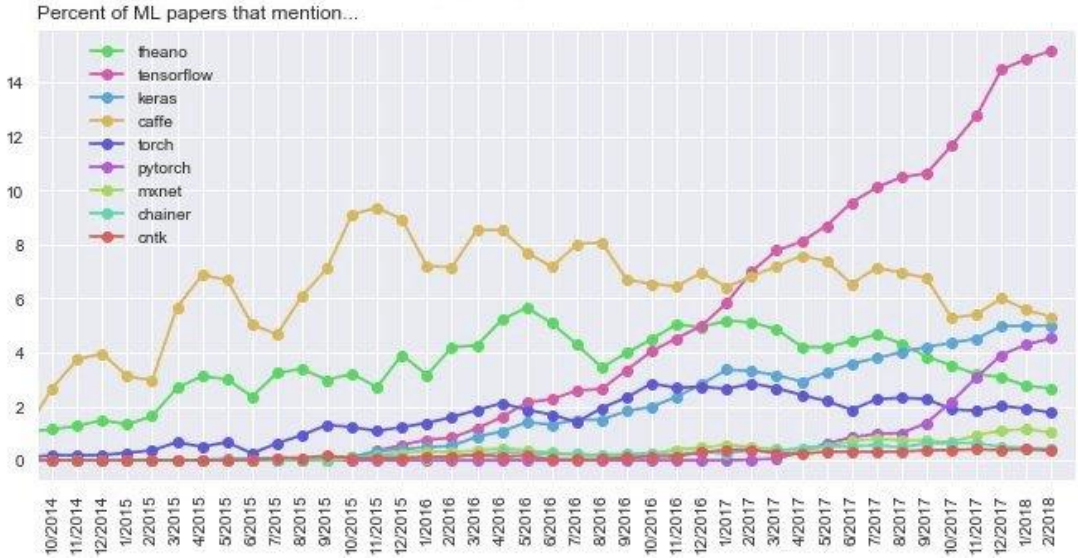
https://github.com/ultralytics/yolov3

https://github.com/ultralytics/yolov5

By [Glenn Jocher]







Copyright FAST CAMPUS Corp. All Rights Reserved

TOOL – development environment

Fast campus

NVIDIA driver ver

https://www.nvidia.com/download/index.aspx

CUDA ver 11.3

https://developer.nvidia.com/rdp/cudnn-archive

cuDNN ver

https://developer.nvidia.com/cuda-11.3.0-download-archive

```
(venv_pytorch) E:₩Projects₩yolov3_pytorch_ultralytics>nvcc -V
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2021 NVIDIA Corporation
Built on Sun_Mar_21_19:24:09_Pacific_Daylight_Time_2021
Cuda compilation tools, release 11.3, V11.3.58
Build cuda_11.3.r11.3/compiler.29745058_0

(venv_pytorch) E:₩Projects₩yolov3_pytorch_ultralytics>nvidia-smi
Mon Aug 22 14:25:13 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 465.89       Driver Version: 465.89       CUDA Version: 11.3     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ... WDDM  | 00000000:01:00.0  On |                  N/A |
| N/A   46C    P8     7W /  N/A |    239MiB /  8192MiB |      3%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      1516    C+G   Insufficient Permissions        N/A      |
|    0   N/A  N/A     16784    C+G   ...5n1h2txyewy₩SearchApp.exe    N/A      |
|    0   N/A  N/A     17124    C+G   ...nputApp₩TextInputHost.exe    N/A      |
+-----------------------------------------------------------------------------+
```

```
(venv_pytorch) E:₩Projects₩yolov3_pytorch_ultralytics>python
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from tensorflow.python.client import device_lib
>>> device_lib.list_local_devices()
2022-08-22 15:24:15.961747: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized
with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operat
ions:  AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-08-22 15:24:17.422549: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1532] Created device /device:GPU:0 wit
h 6636 MB memory:  -> device: 0, name: NVIDIA GeForce GTX 1070, pci bus id: 0000:01:00.0, compute capability: 6.1
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 1492384573320559791
xla_global_id: -1
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 6958415872
locality {
  bus_id: 1
  links {
  }
}
incarnation: 6653115814557146904
physical_device_desc: "device: 0, name: NVIDIA GeForce GTX 1070, pci bus id: 0000:01:00.0, compute capability: 6.1"
xla_global_id: 416903419
]
```

Tensorflow and Tensorflow-gpu ver

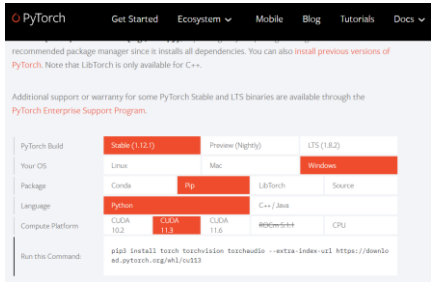
pip install tensorflow
pip install tensorflow-gpu
pip install gast==0.3.3

```
from tensorflow.python.client import device_lib
device_lib.list_local_devices()
```

Keras and PyTorch ver 11.3

https://pytorch.org/

https://pytorch.org/get-started/previous-versions/



pip install torch torchvision torchaudio --extra-index-url https://download.pytorch.org/whl/cu113

```
python
import torch
print(torch.__version__)
```

Copyright FAST CAMPUS Corp. All Rights Reserved

TOOL – development environment

Fast campus

PIP

https://www.python.org/downloads/

Anaconda

https://www.anaconda.com/

Jupyter notebook

https://jupyter.org/

virtualenv

pip install virtualenv
pip3 install virtualenv

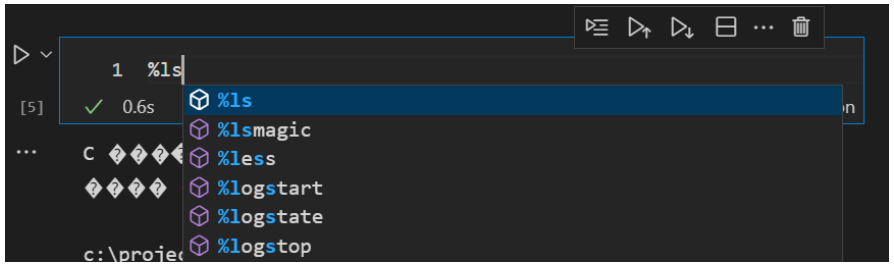
python virtualenv

3.7, 3.9

.₩venv₩Scripts₩activate
source ./venv/bin/activate

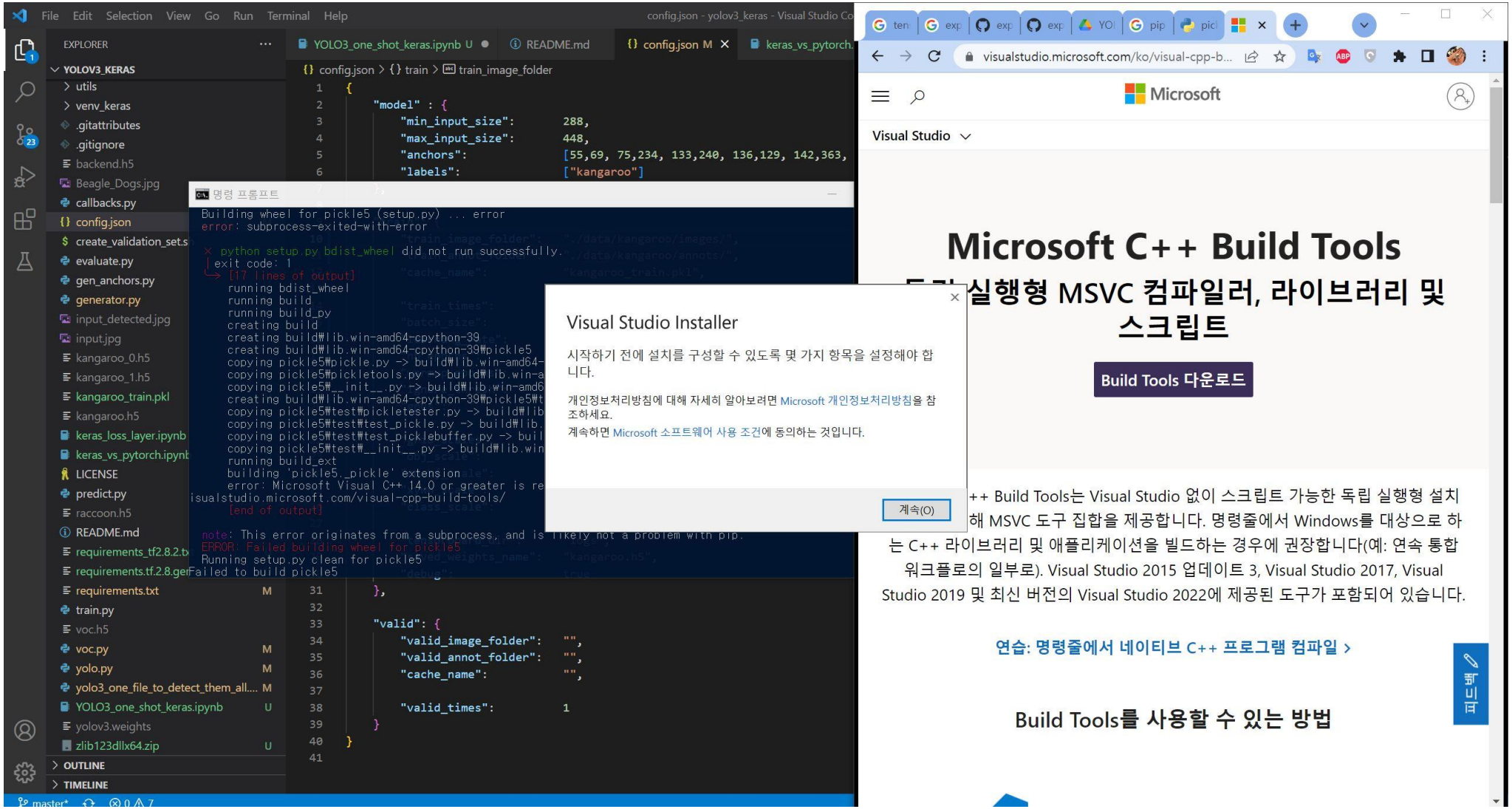
Jupyter/IPython Notebook Quick Start Guide
https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/index.html

28 Jupyter Notebook Tips, Tricks, and Shortcuts
https://www.dataquest.io/blog/jupyter-notebook-tips-tricks-shortcuts/



Copyright FAST CAMPUS Corp. All Rights Reserved

TOOL – development environment



https://visualstudio.microsoft.com/ko/visual-cpp-build-tools/

Copyright FAST CAMPUS Corp. All Rights Reserved

TOOL – development environment

명령 프롬프트

```
E:₩Projects₩keras-yolo3>virtualenv venv
created virtual environment CPython3.8.3.final.0-64 in 2352ms
  creator CPython3Windows(dest=E:₩Projects₩keras-yolo3₩venv, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:₩Users₩KTW₩Ap
pData₩Local₩pypa₩virtualenv)
    added seed packages: pip==22.2.2, setuptools==63.4.1, wheel==0.37.1
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
```

```
E:₩Projects₩keras-yolo3>.₩venv₩Scripts₩activate

(venv) E:₩Projects₩keras-yolo3>
```

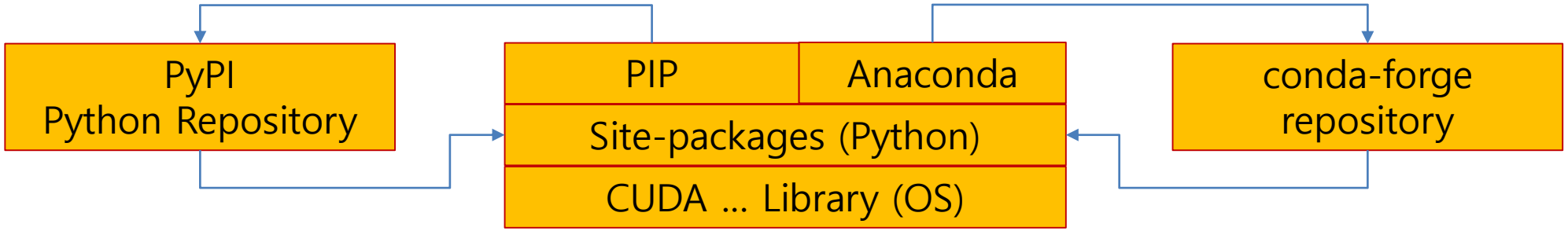
pip install ipywidgets

명령 프롬프트 - python

```
  --use-deprecated <feature>  Enable deprecated functionality, that will be removed in the future
.

(venv_tf2) C:₩projects>python
Python 3.9.12 (tags/v3.9.12:b28265d, Mar 23 2022, 23:52:46) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Copyright FAST CAMPUS Corp. All Rights Reserved

TOOL – development environment



pip conda package dependencies conflict problem

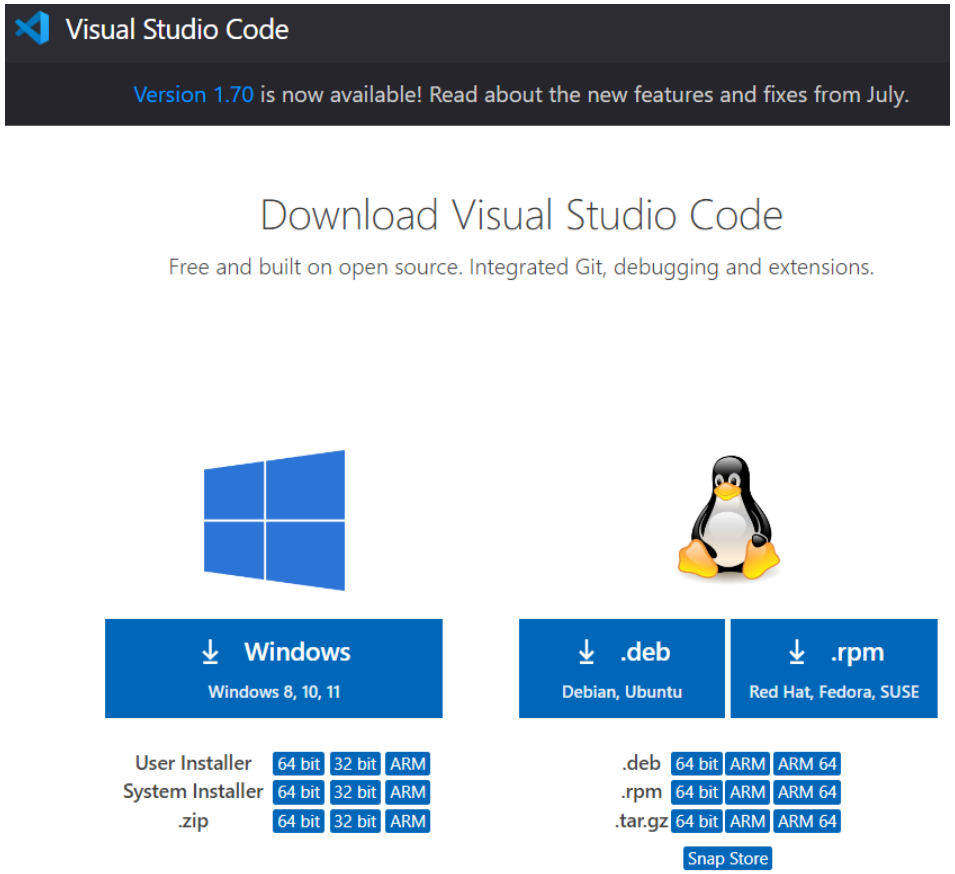
Copyright FAST CAMPUS Corp. All Rights Reserved

TOOL – development environment

Fast campus

vscode

https://code.visualstudio.com/download



Copyright FAST CAMPUS Corp. All Rights Reserved

TOOL – development environment

Fast campus

unzip

http://stahlworks.com/dev/?tool=zipunzip

Products | DView | SFK | Buy Now | Downloads | Support | Company | Embedded | Mobile

FREEWARE DOWNLOADS:

Swiss File Knife

a command line multi function tool.

remove tabs
list dir sizes
find text
filter lines
find in path
collect text
instant ftp or

How to get zip and unzip on the windows command line for quick creation and extraction of zip compressed files.
Download the free **Windows executables**: (for Linux, see below)

- to **extract zip files on the command line**, download unzip.exe here.

  this file is a copy of the original Info-ZIP unzip.exe version 5.52, provided for free under the Info-ZIP license. Alternatively, you may download it via the original Info-ZIP website.

  How to use unzip? Read the examples below.

- to **create zip files on the command line**, download zip.exe here.

ZLIB

NVIDIA. ACCELERATED COMPUTING

NVIDIA CUDNN DOCUMENTATION

Search

Getting Started
Installation Guide
1. Overview
▷ 2. Installing cuDNN on Linux
▽ 3. Installing cuDNN on Windows
▽ 3.1. Prerequisites

3.1.3. Installing Zlib

Zlib is a data compression software library that is needed by cuDNN.

Procedure

1. Download and extract the zlib package from ZLIB DLL. Users with a 32-bit machine should download the 32-bit ZLIB DLL.

WGET

https://eternallybored.org/misc/wget/

Copyright FAST CAMPUS Corp. All Rights Reserved

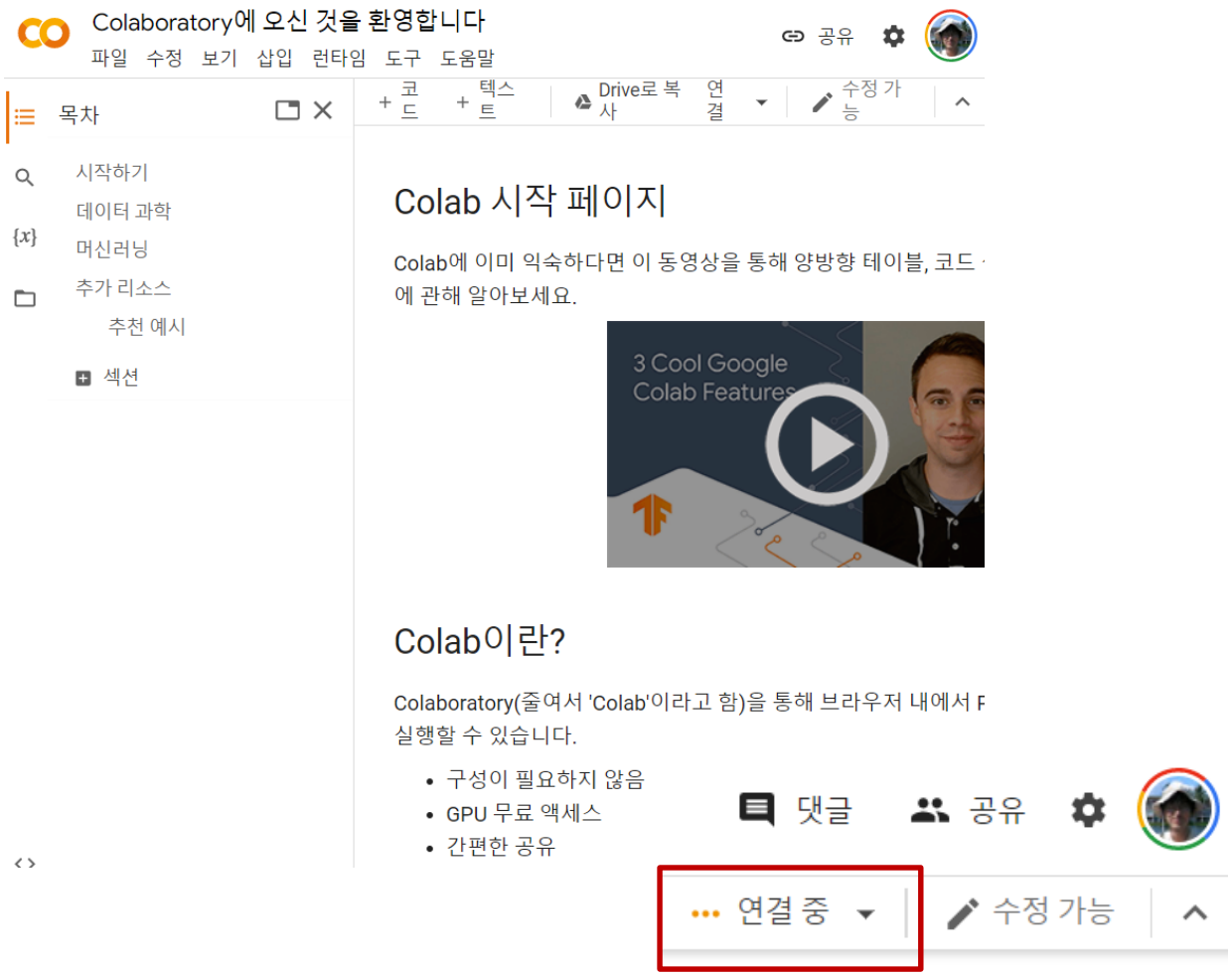
TOOL – development environment

Dive

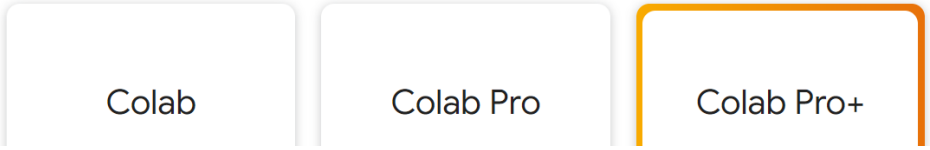
Fast campus

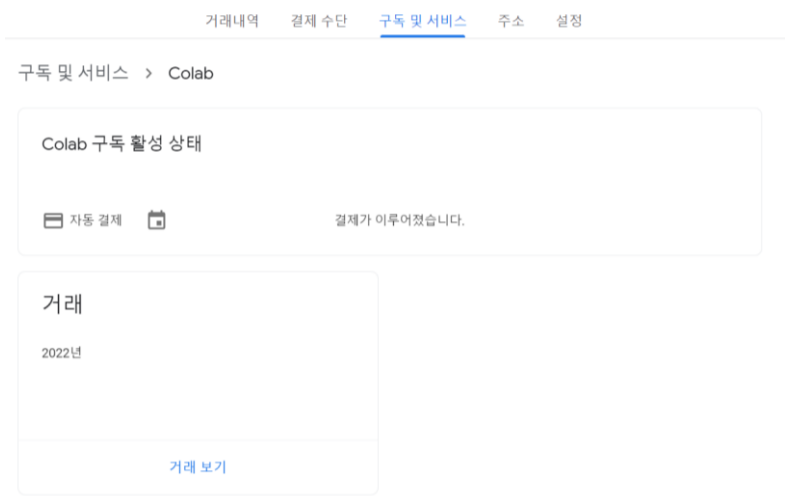
colab

https://colab.research.google.com/



https://colab.research.google.com/signup





Copyright FAST CAMPUS Corp. All Rights Reserved

TOOL – development environment

https://sourceforge.net/projects/staruml/

Copyright FAST CAMPUS Corp. All Rights Reserved

TOOL – development environment

Fast campus

Open source license

https://www.olis.or.kr/license/Detailselect.do?lId=1072

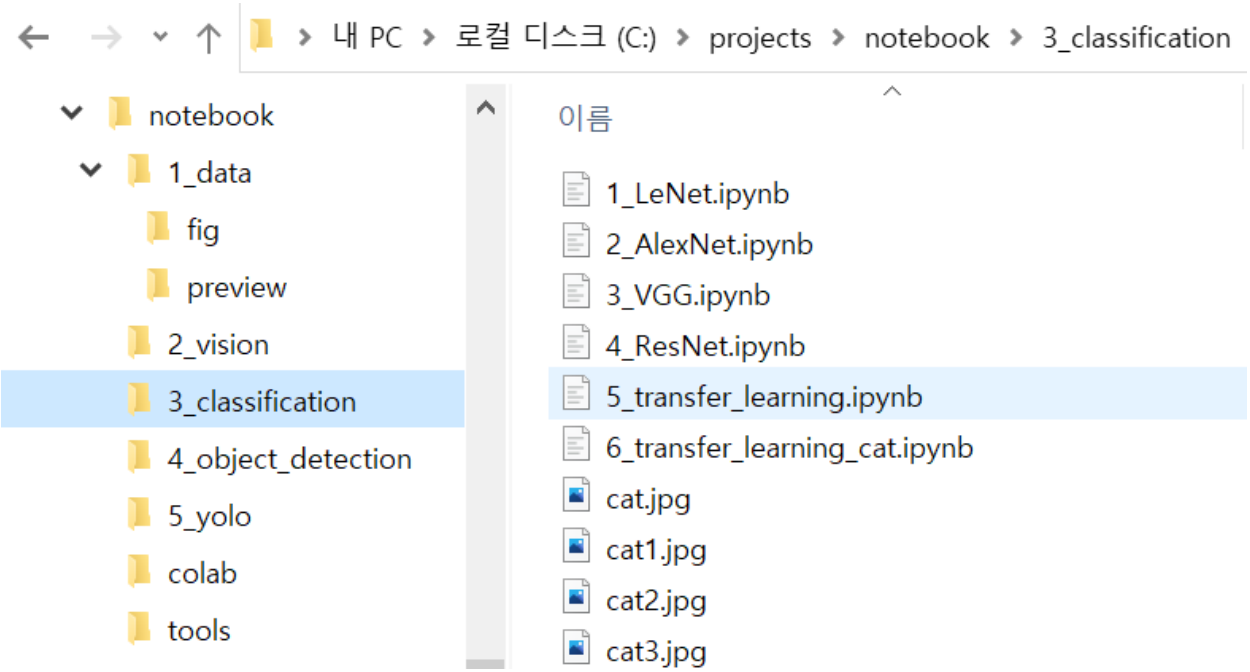
	APACHE	BSD	MIT	GPL Free as in Freedom	LGPL Free as in Freedom	AGPL Free as in Freedom
Type	Permissive	Permissive	Permissive	Copyleft	Copyleft	Copyleft
Provides copyright protection	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE
Can be used in commercial applications	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE
Provides an explicit patent license	✓ TRUE	✗ FALSE	✗ FALSE	✗ FALSE	✗ FALSE	✗ FALSE
Can be used in proprietary (closed source) projects	✓ TRUE	✓ TRUE	✓ TRUE	✗ FALSE	✗ FALSE partially	✗ FALSE for web
Popular open-source and free projects	Kubernetes Swift Firebase	Django React Flutter	Angular.js JQuery, .NET Core Laravel	Joomla Notepad++ MySQL	Qt SharpDevelop	SugarCRM Launchpad

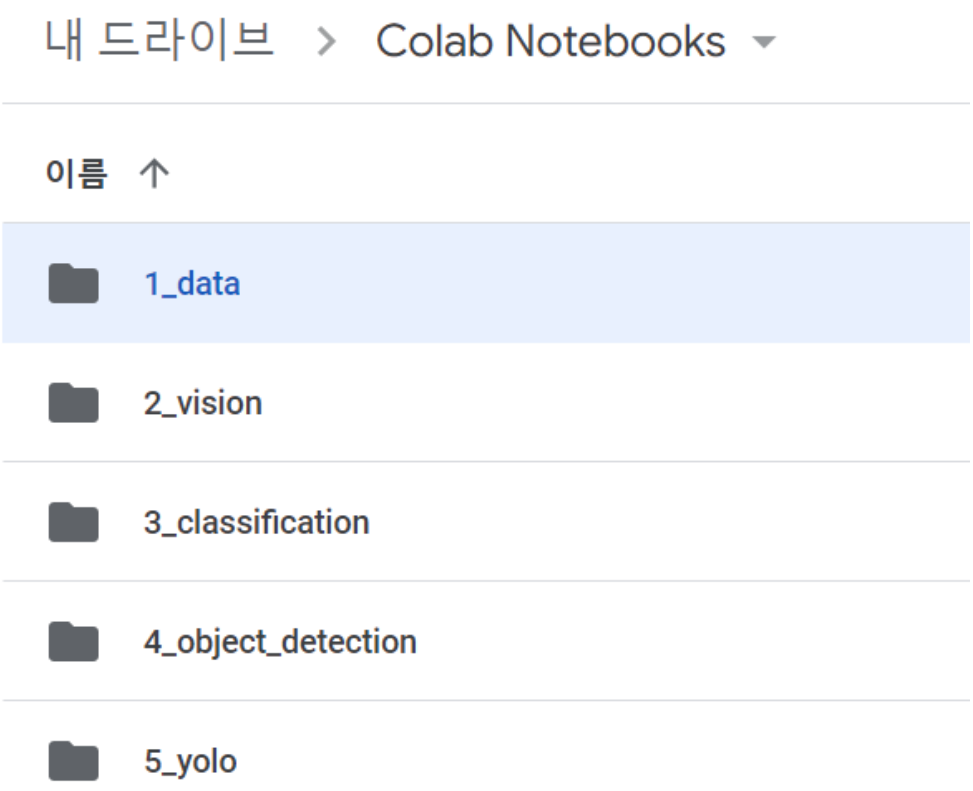
Moqod, 2019, Understanding open-source and free software licensing, https://moqod-software.medium.com/understanding-open-source-and-free-software-licensing-c0fa600106c9

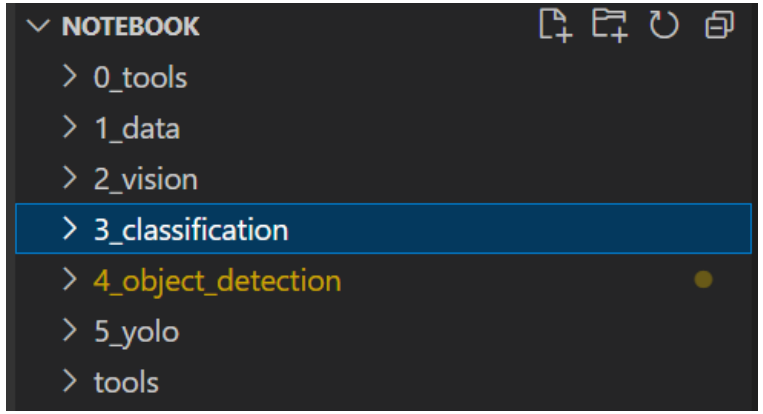
Copyright FAST CAMPUS Corp. All Rights Reserved

Source code & Dataset

Fast campus

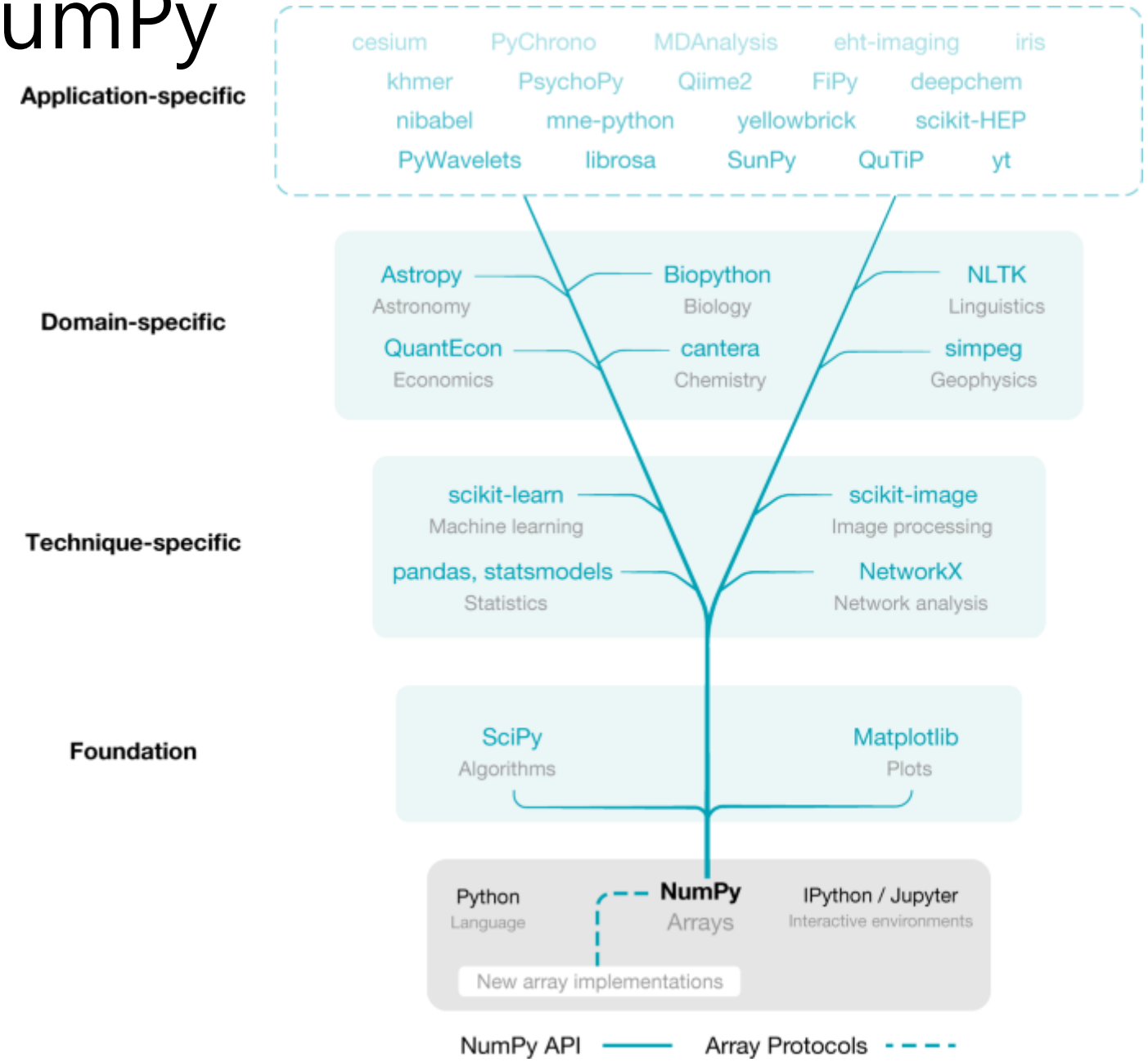






Copyright FAST CAMPUS Corp. All Rights Reserved

Python & NumPy



Charles R. Harris, K. Jarrod Millman, 2020, Array programming with NumPy, nature, https://www.nature.com/articles/s41586-020-2649-2

Copyright **FAST CAMPUS** Corp. All Rights Reserved

Tensorflow, Keras

Fast campus

Keras

Star 55,957

- About Keras
- Getting started
  - Introduction to Keras for engineers
  - Introduction to Keras for researchers
  - The Keras ecosystem
  - Learning resources
  - Frequently Asked Questions
- Developer guides
- Keras API reference
- Code examples
- Why choose Keras?
- Community & governance
- Contributing to Keras
- KerasTuner
- KerasCV
- KerasNLP

Search Keras documentation...

» Getting started

Getting started

Are you an engineer or data scientist? Do you ship reliable and performant applied machine learning solutions? Check out our **Introduction to Keras for engineers**.

Are you a machine learning researcher? Do you publish at NeurIPS and push the state-of-the-art in CV and NLP? Check out our **Introduction to Keras for researchers**.

Are you a beginner looking for both an introduction to machine learning and an introduction to Keras and TensorFlow? You're going to need more than a one-pager. And you're in luck: **we've got just the book for you**.

Further starter resources

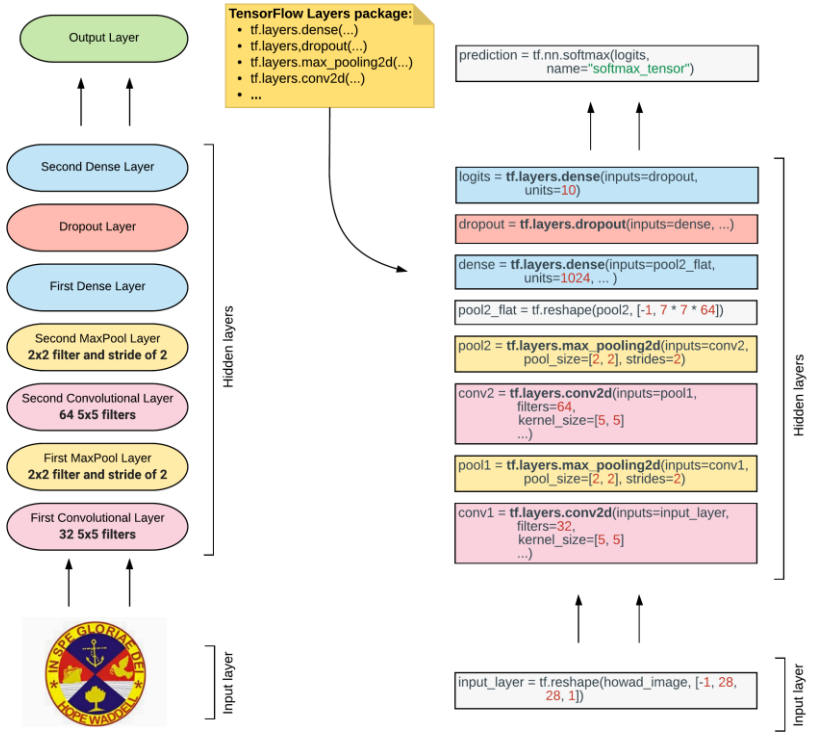
- The Keras ecosystem
- Learning resources
- Frequently Asked Questions

Installing Keras

To use Keras, will need to have the TensorFlow package installed. See detailed instructions.

Once TensorFlow is installed, just import Keras via:

```
from tensorflow import keras
```



https://ekababisong.org/gcp-ml-seminar/tensorflow/

https://keras.io/getting_started/

Copyright FAST CAMPUS Corp. All Rights Reserved

PyTorch

Fast campus

PyTorch

Get Started | Ecosystem | Mobile | Blog | Tutorials | Docs | Resources | GitHub

1.12.1+cu102

Search Tutorials

PyTorch Recipes [+]

Introduction to PyTorch [-]

- Learn the Basics
- Quickstart
- Tensors
- Datasets & DataLoaders
- Transforms
- Build the Neural Network
- Automatic Differentiation with `torch.autograd`
- Optimizing Model Parameters
- Save and Load the Model

Introduction to PyTorch on YouTube [-]

- Introduction to PyTorch - YouTube Series
- Introduction to PyTorch
- Introduction to PyTorch Tensors

Tutorials > Quickstart

Shortcuts

Run in Microsoft Learn | Run in Google Colab | Download Notebook | View on GitHub

Learn the Basics || **Quickstart** || Tensors || Datasets & DataLoaders || Transforms || Build Model || Autograd || Optimization || Save & Load Model

QUICKSTART

This section runs through the API for common tasks in machine learning. Refer to the links in each section to dive deeper.

Working with data

PyTorch has two primitives to work with data: `torch.utils.data.DataLoader` and `torch.utils.data.Dataset`. `Dataset` stores the samples and their corresponding labels, and `DataLoader` wraps an iterable around the `Dataset`.

```
import torch
from torch import nn
from torch.utils.data import DataLoader
from torchvision import datasets
from torchvision.transforms import ToTensor
```

Quickstart
- Working with data
- Creating Models
- Optimizing the Model Parameters
- Saving Models
- Loading Models

https://pytorch.org/tutorials/beginner/basics/quickstart_tutorial.html

Copyright FAST CAMPUS Corp. All Rights Reserved

keras vs pytorch

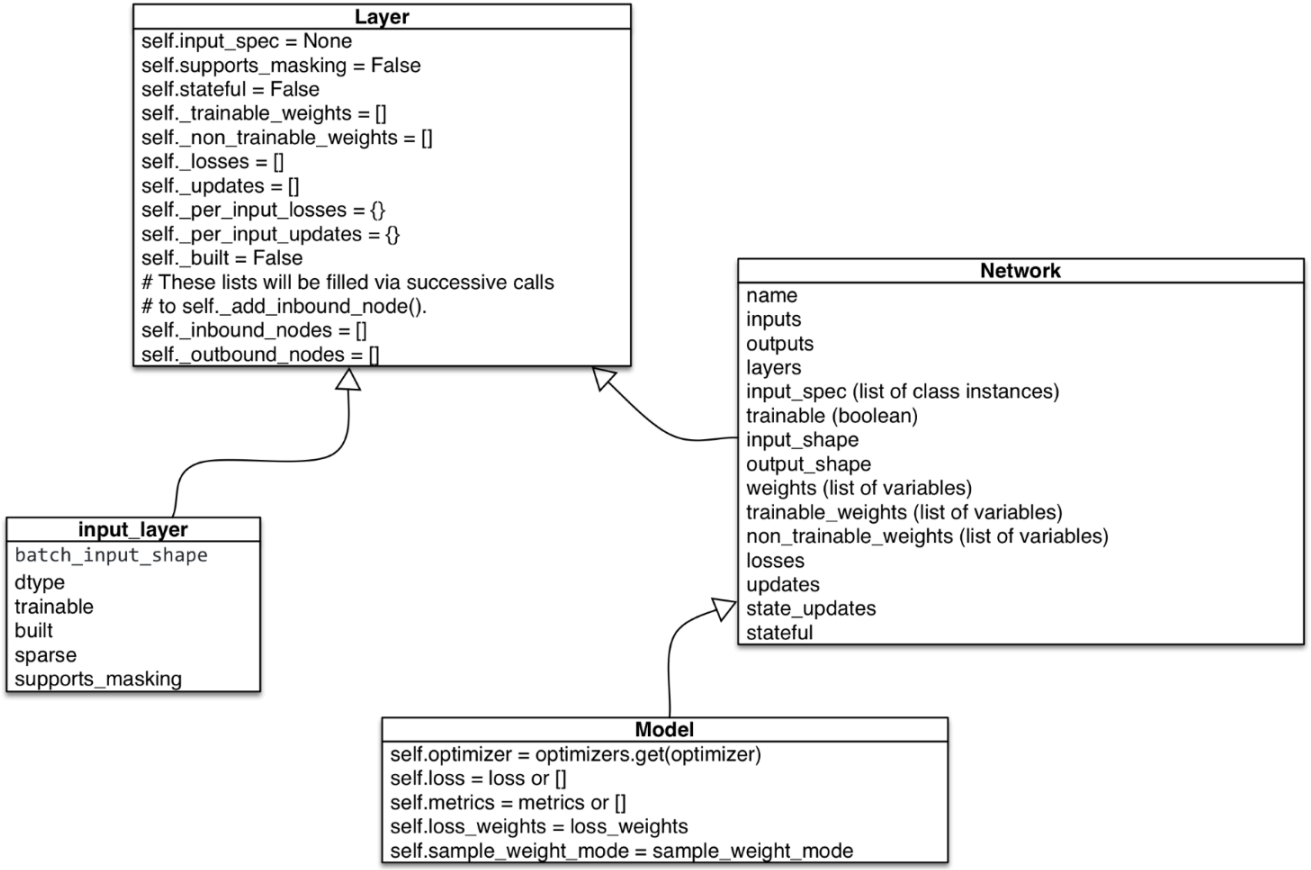
	Keras	PyTorch	TensorFlow
API Level	High	Low	High and Low
Architecture	Simple, concise, readable	Complex, less readable	Not easy to use
Datasets	Smaller datasets	Large datasets, high perf ormance	Large datasets, high perf ormance
Debugging	Simple network, so debugging is not oft en needed	Good debugging capabil ities	Difficult to conduct deb ugging
Does It Have Trained Mo dels?	Yes	Yes	Yes
Popularity	Most popular	Third most popular	Second most popular
Speed	Slow, low performance	Fast, high-performance	Fast, high-performance
Written In	Python	Lua	C++, CUDA, Python

John Terra, 2022.7, Keras vs Tensorflow vs Pytorch: Key Differences Among the Deep Learning Framework, simplilearn

Copyright FAST CAMPUS Corp. All Rights Reserved

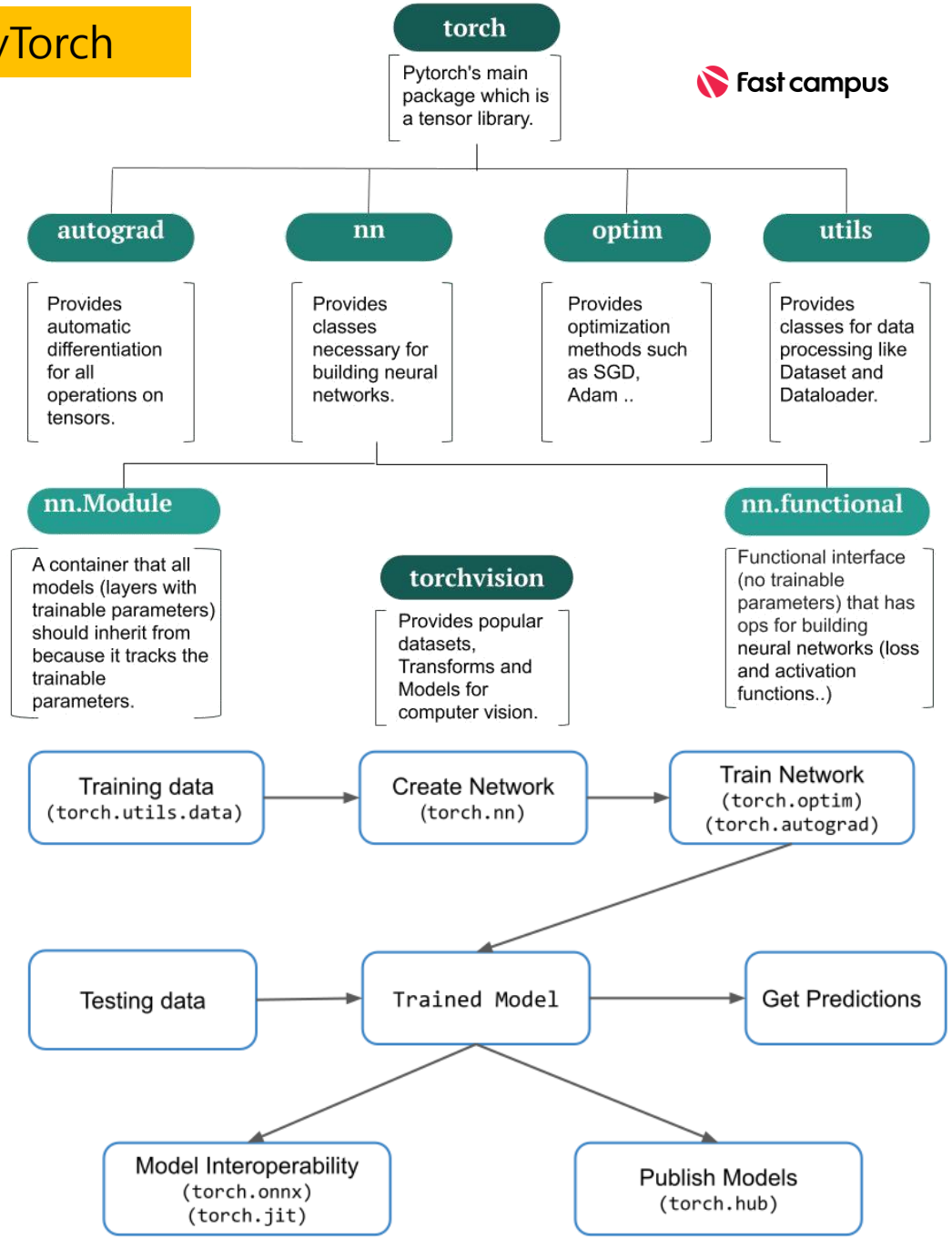
keras vs pytorch

Keras



Models in Keras – getting started, https://subscription.packtpub.com/book/big-data-and-business-intelligence/9781788621755/2/ch02lvl1sec20/models-in-keras-getting-started

PyTorch



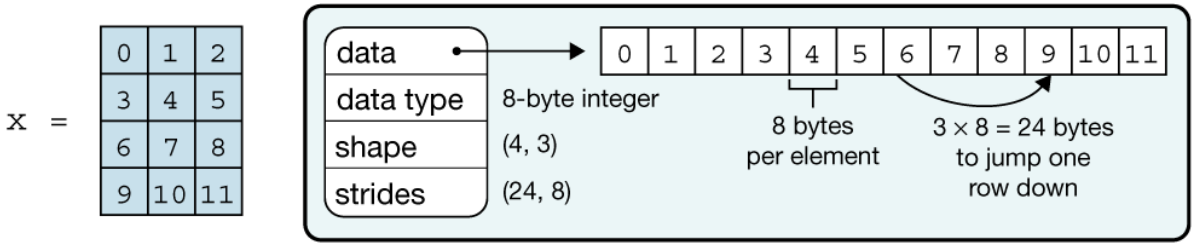
Shrimali 2019, PyTorch, https://devopedia.org/pytorch

Copyright FAST CAMPUS Corp. All Rights Reserved

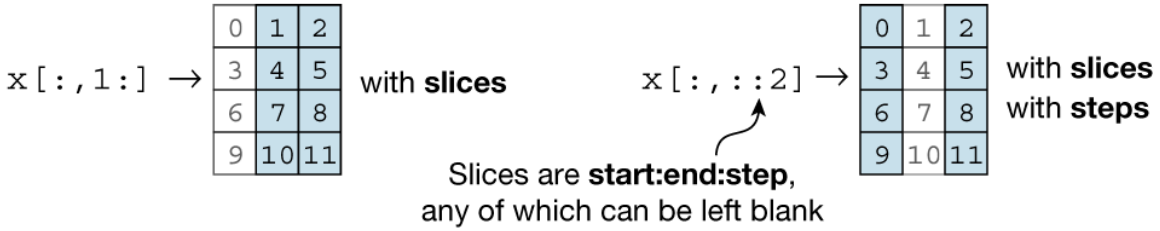
Python & NumPy

Fast campus

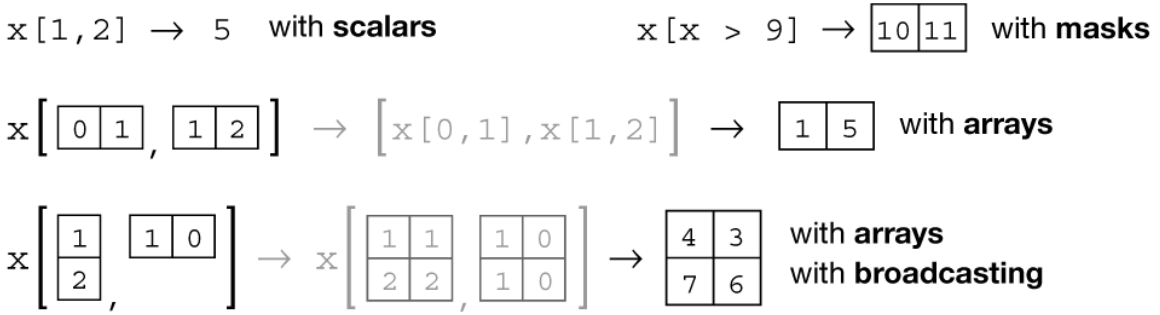
a Data structure



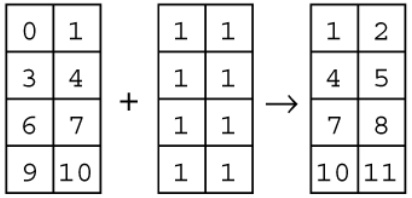
b Indexing (view)



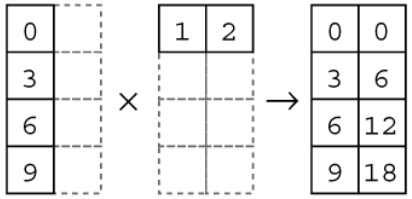
c Indexing (copy)



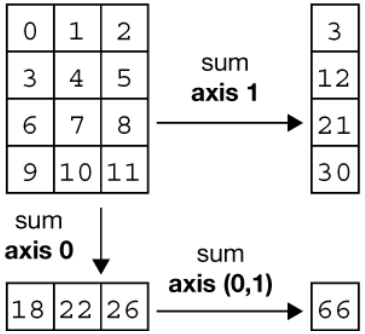
d Vectorization



e Broadcasting



f Reduction



g Example

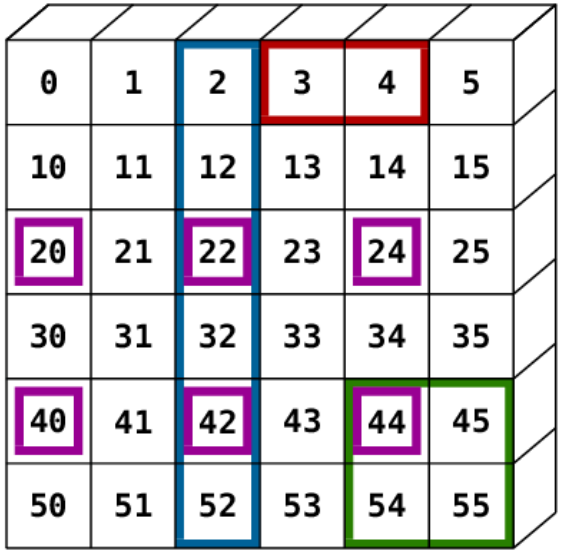
```
In [1]: import numpy as np

In [2]: x = np.arange(12)

In [3]: x = x.reshape(4, 3)

In [4]: x
Out[4]:
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

In [5]: np.mean(x, axis=0)
Out[5]: array([4.5, 5.5, 6.5])

In [6]: x = x - np.mean(x, axis=0)

In [7]: x
Out[7]:
array([[-4.5, -4.5, -4.5],
       [-1.5, -1.5, -1.5],
       [ 1.5,  1.5,  1.5],
       [ 4.5,  4.5,  4.5]])
```

Charles R. Harris, K. Jarrod Millman, 2020, Array programming with NumPy, nature, https://www.nature.com/articles/s41586-020-2649-2

Copyright FAST CAMPUS Corp. All Rights Reserved

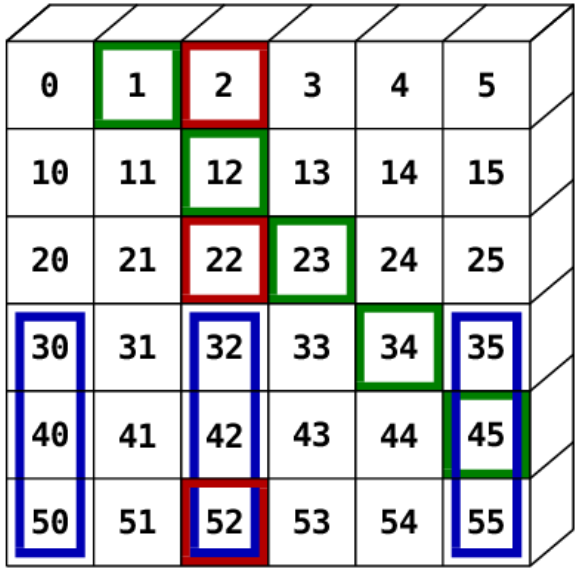
Python & NumPy

```
>>> a[0, 3:5]
array([3, 4])

>>> a[4:, 4:]
array([[44, 55],
       [54, 55]])

>>> a[:, 2]
a([2, 12, 22, 32, 42, 52])

>>> a[2::2, ::2]
array([[20, 22, 24],
       [40, 42, 44]])
```



ndarry[start : stop : step]
  start: start index (default = 0)
  stop: end index (default = last index)
  step: default =1

```
>>> a[(0,1,2,3,4), (1,2,3,4,5)]
array([1, 12, 23, 34, 45])

>>> a[3:, [0,2,5]]
array([[30, 32, 35],
       [40, 42, 45],
       [50, 52, 55]])

>>> mask = np.array([1,0,1,0,0,1], dtype=bool)
>>> a[mask, 2]
array([2, 22, 52])
```

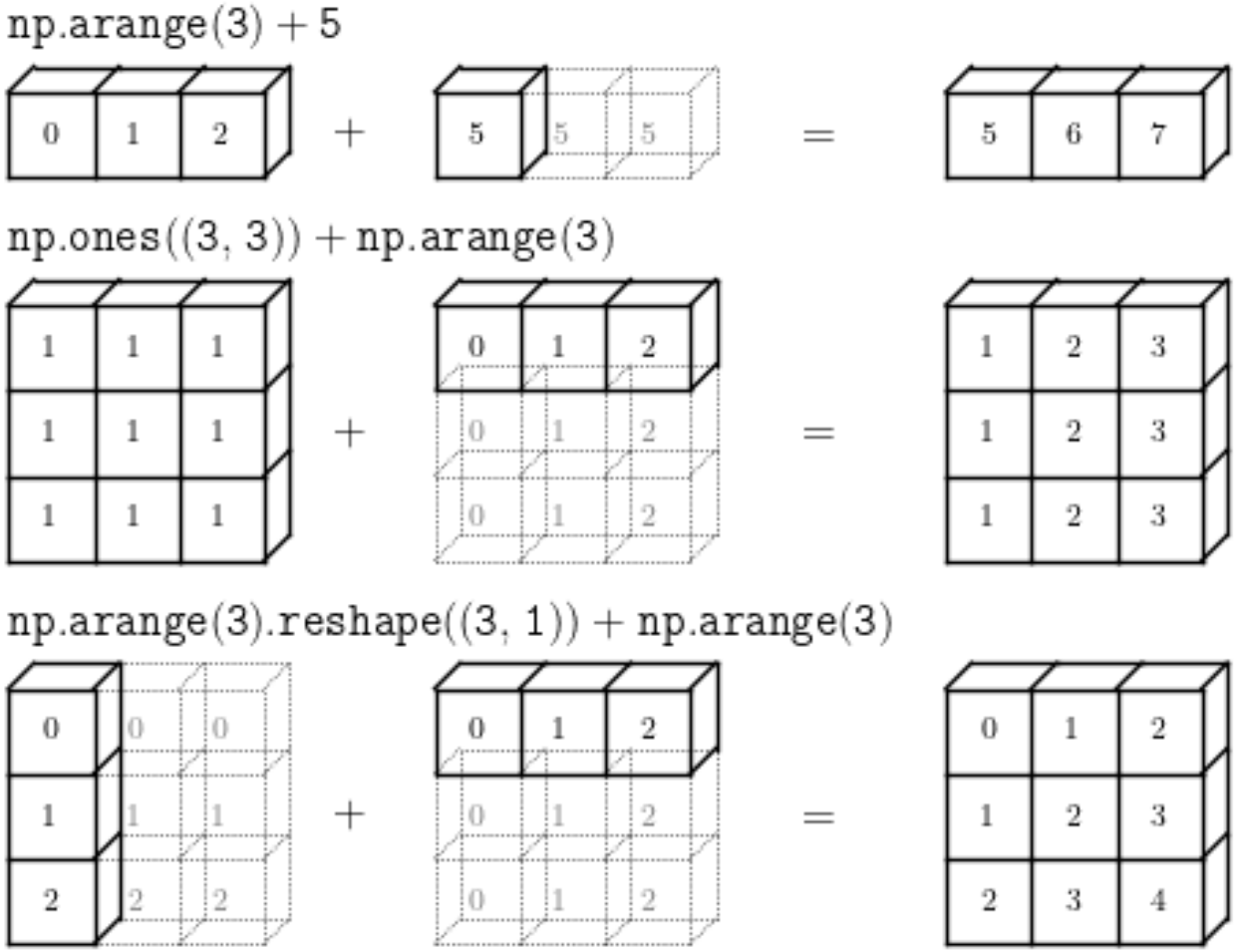


Scipy Lecture Notes, 2022, https://scipy-lectures.org/intro/numpy/array_object.html

Copyright FAST CAMPUS Corp. All Rights Reserved

Python & NumPy

stretch



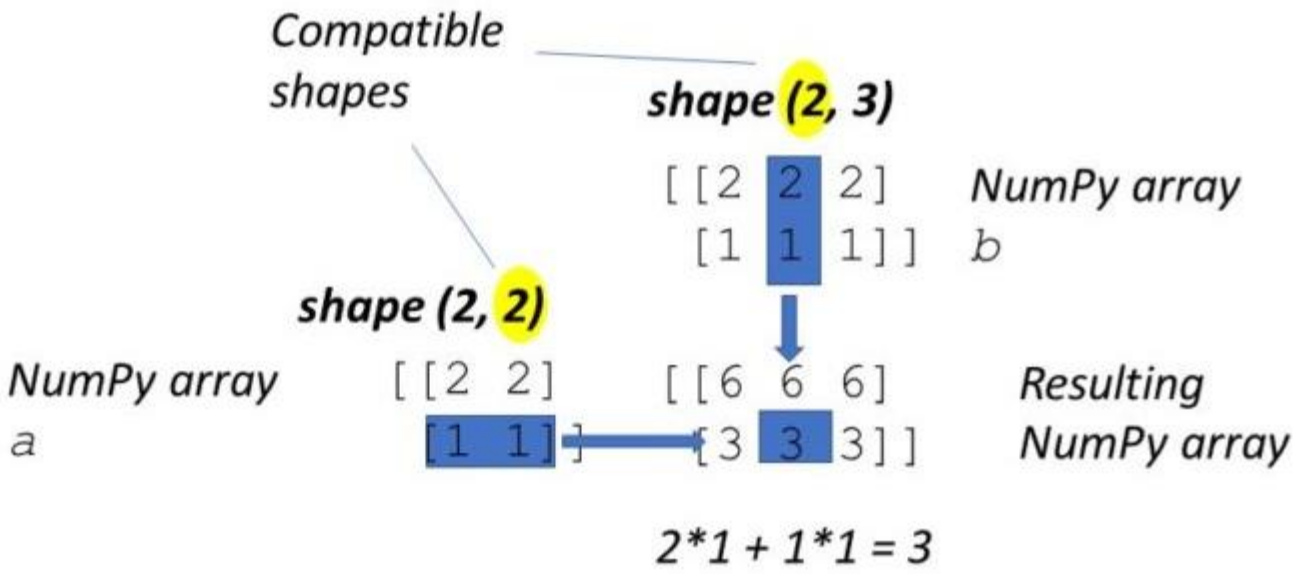
Broadcast Visualization, https://www.astroml.org/book_figures/appendix/fig_broadcast_visual.html

Copyright FASTCAMPUS Corp. All Rights Reserved

Python & NumPy

```
a@b
np.dot(a,b)
np.matmul(a,b)
```

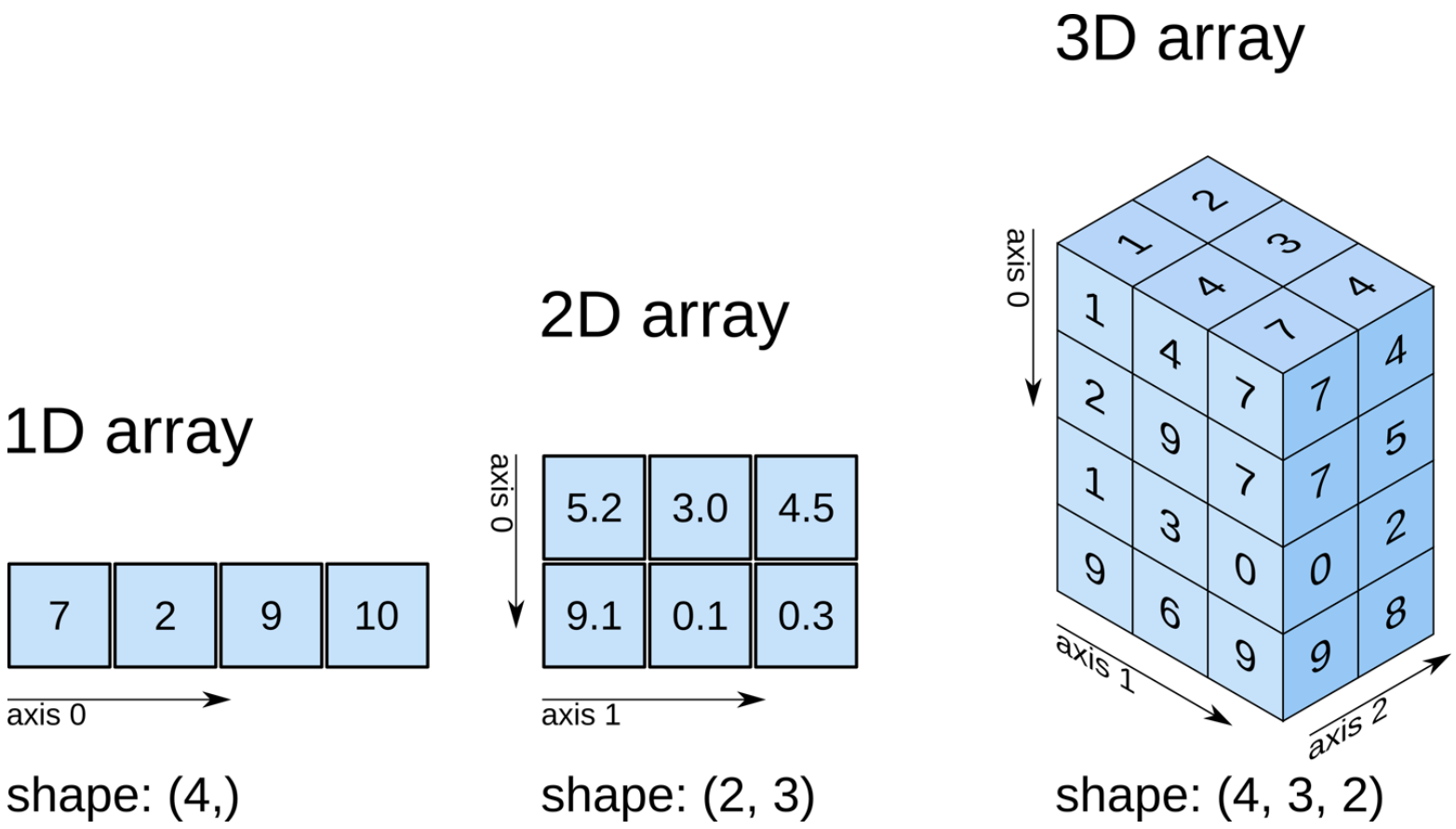
Equivalent Semantics:
Matrix Multiplication



Adam Murphy, NumPy Matrix Multiplication, https://blog.finxter.com/numpy-matmul-operator/

Copyright FAST CAMPUS Corp. All Rights Reserved

Data shape



https://fgnt.github.io/python_crashkurs_doc/include/numpy.html

Copyright FAST CAMPUS Corp. All Rights Reserved

Data shape

```
import numpy as np
a = np.arange(120).reshape(2, 3, 4, 5)
print(a.shape)
# (2, 3, 4, 5)
```

```
[[[[  0   1   2   3   4]
   [  5   6   7   8   9]
   [ 10  11  12  13  14]
   [ 15  16  17  18  19]]

  [[ 20  21  22  23  24]
   [ 25  26  27  28  29]
   [ 30  31  32  33  34]
   [ 35  36  37  38  39]]

  [[ 40  41  42  43  44]
   [ 45  46  47  48  49]
   [ 50  51  52  53  54]
   [ 55  56  57  58  59]]]

 [[[ 60  61  62  63  64]
   [ 65  66  67  68  69]
   [ 70  71  72  73  74]
   [ 75  76  77  78  79]]

  [[ 80  81  82  83  84]
   [ 85  86  87  88  89]
   [ 90  91  92  93  94]
   [ 95  96  97  98  99]]

  [[100 101 102 103 104]
   [105 106 107 108 109]
   [110 111 112 113 114]
   [115 116 117 118 119]]]]
```

```
print(a[:, :, :, 0])
# [[[  0    5   10   15]
#    [ 20   25   30   35]
#    [ 40   45   50   55]]
#
#  [[ 60   65   70   75]
#    [ 80   85   90   95]
#    [100  105  110  115]]]
```

```
print(a[..., 0])
# [[[  0    5   10   15]
#    [ 20   25   30   35]
#    [ 40   45   50   55]]
#
#  [[ 60   65   70   75]
#    [ 80   85   90   95]
#    [100  105  110  115]]]
```

```
print(a[Ellipsis, 0])
# [[[  0    5   10   15]
#    [ 20   25   30   35]
#    [ 40   45   50   55]]
#
#  [[ 60   65   70   75]
#    [ 80   85   90   95]
#    [100  105  110  115]]]
```

https://fgnt.github.io/python_crashkurs_doc/include/numpy.html

Copyright FAST CAMPUS Corp. All Rights Reserved

Args, kwargs

*args (Non-Keyword Arguments)
**kwargs (Keyword Arguments)

```
def myFun(*argv):
    for arg in argv:
        print(arg)

myFun('Hello', 'Welcome', 'to', 'GeeksforGeeks')
```

```
First argument : Hello
Next argument through *argv : Welcome
Next argument through *argv : to
Next argument through *argv : GeeksforGeeks
```

```
def myFun(**kwargs):
    for key, value in kwargs.items():
        print("%s == %s" % (key, value))


# Driver code
myFun(first='Geeks', mid='for', last='Geeks')
```

```
first == Geeks
mid == for
last == Geeks
```

```
tf.keras.layers.Add(
    **kwargs
)
```

https://www.geeksforgeeks.org/args-kwargs-python/

Copyright FAST CAMPUS Corp. All Rights Reserved

Data shape

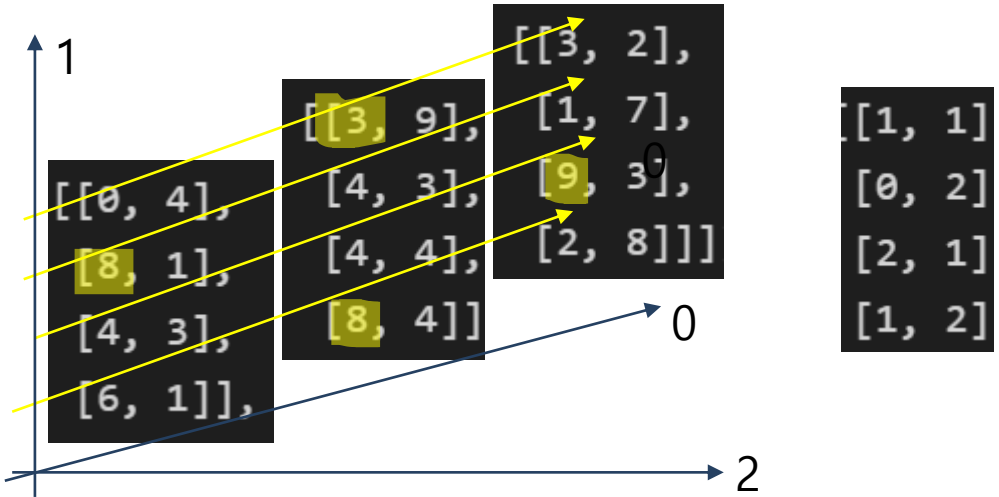
```
>>> input_shape = (2, 3, 4)
>>> x1 = tf.random.normal(input_shape)
>>> x2 = tf.random.normal(input_shape)
>>> y = tf.keras.layers.Add()([x1, x2])
>>> print(y.shape)
(2, 3, 4)
```

```
model = tf.keras.Sequential()
model.add(tf.keras.layers.Reshape((3, 4), input_shape=(12,)))
model.output_shape  # (None, 3, 4)
model.add(tf.keras.layers.Reshape((-1, 2, 2)))   # (None, 3, 2, 2)
```

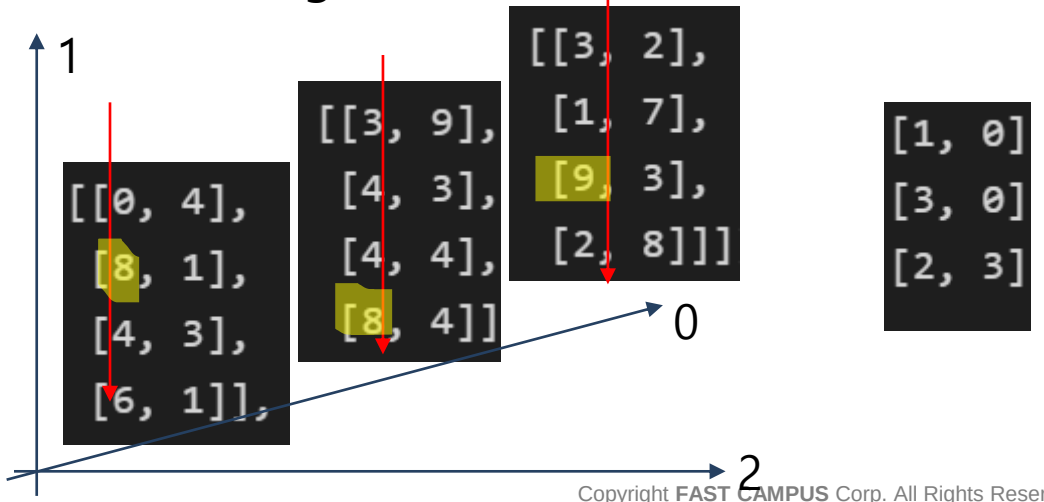
```
x = torch.rand(2, 3, 4)  # [2, 3, 4] = 24
y = x.view(2, -1)        # [2, 12]
z = y.unsqueeze(0)       # [1, 2, 12]
z = z.squeeze(0)         # [2, 12]  = 24
```

```
x = torch.arange(0, 5).reshape(1, 5)
y = x.transpose(0, 1)  # [5, 1]
z = x.permute(1, 0)  # [5, 1]
```

```
x = torch.randint(10, (3, 4, 2))
y = torch.argmax(x, dim=0)  # return max index, dim=collapse.
```

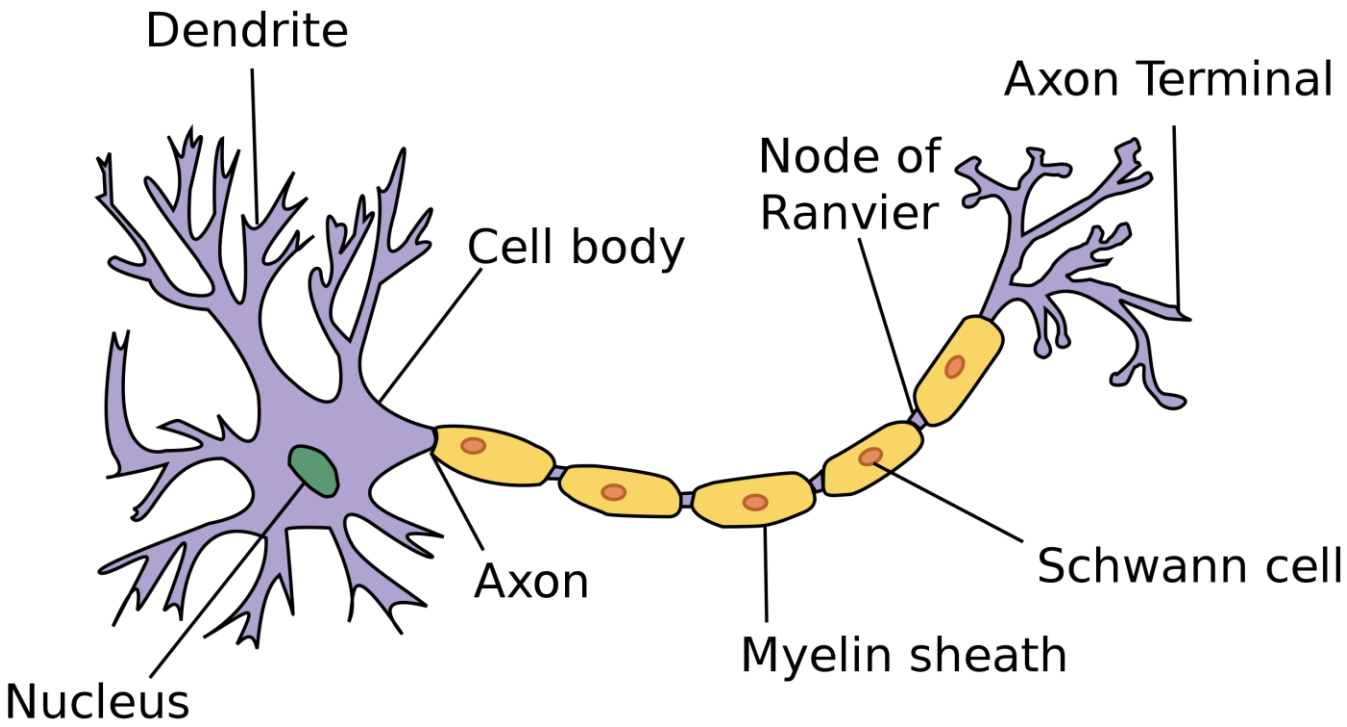



```
z = torch.argmax(x, dim=1)
```



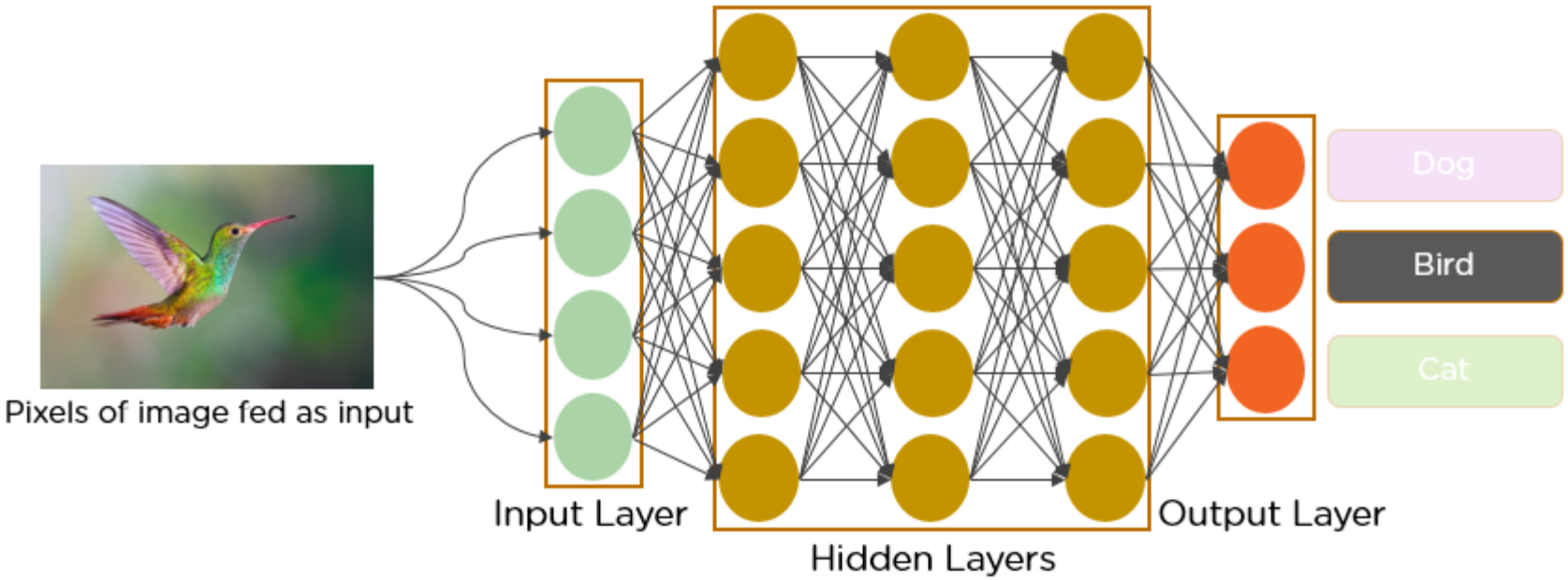
Copyright FAST CAMPUS Corp. All Rights Reserved

NN concept definition - Glossary of terms



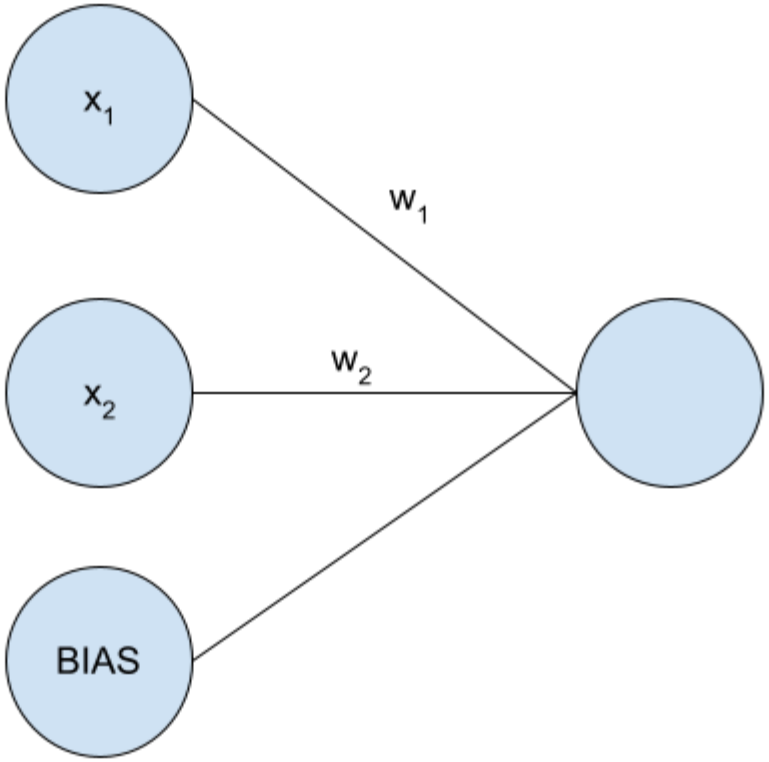
Copyright FAST CAMPUS Corp. All Rights Reserved

NN concept definition - Glossary of terms



Copyright FAST CAMPUS Corp. All Rights Reserved

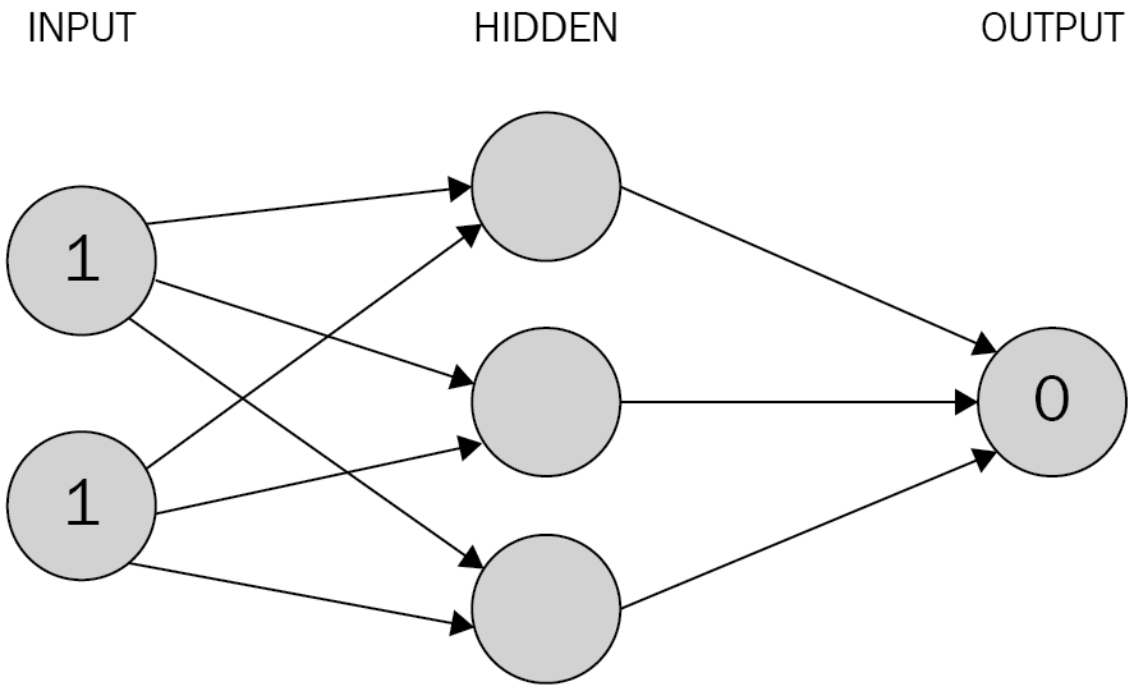
Neural Network



$$Y = XW + B$$

Copyright **FAST CAMPUS** Corp. All Rights Reserved

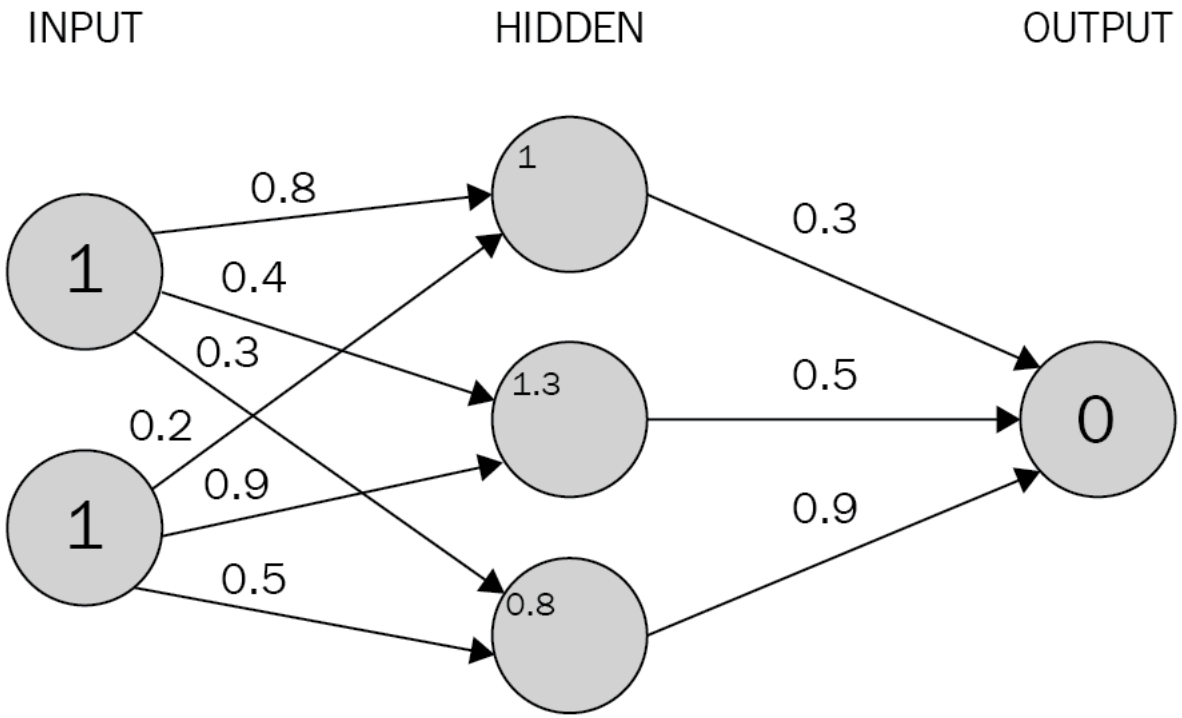
Neural Network – Forward propagation



Packtpub, Feed-forward propagation from scratch in Python

Copyright FAST CAMPUS Corp. All Rights Reserved

Neural Network – Forward propagation

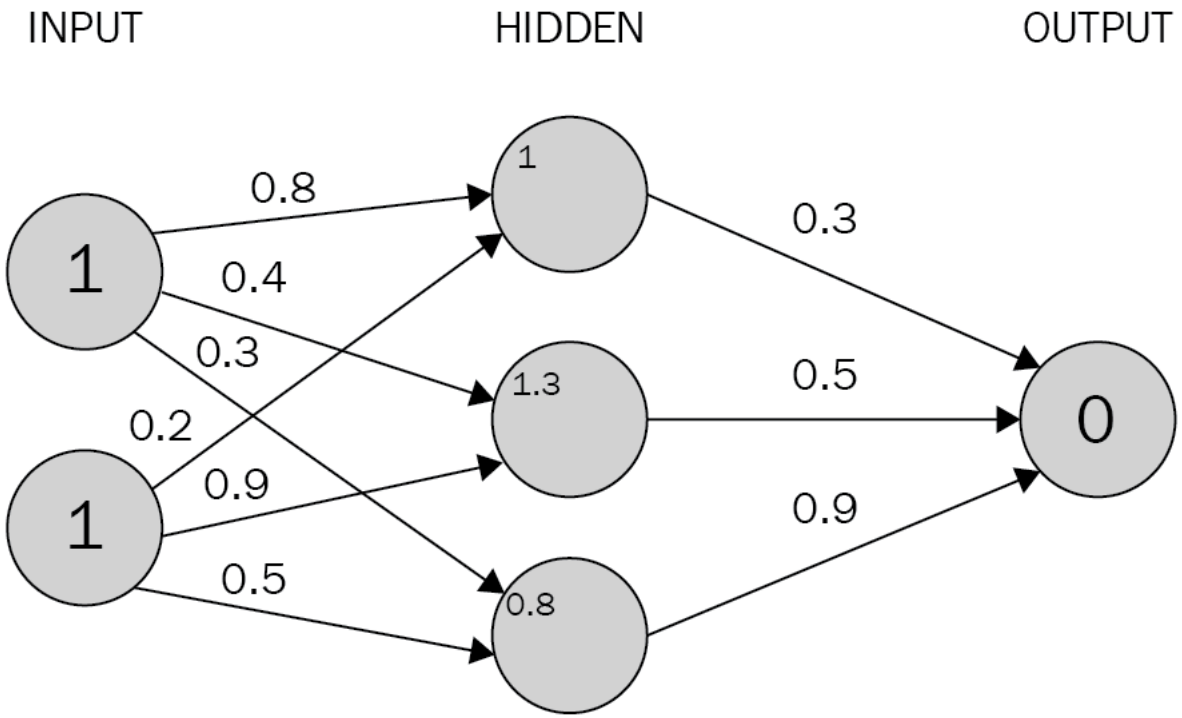


H1=1*0.8 + 1*0.2=1
H2=1*0.4 + 1*0.9=1.3
H3=1*0.3 + 1*0.5=0.8

Packtpub, Feed-forward propagation from scratch in Python

Copyright FAST CAMPUS Corp. All Rights Reserved

Neural Network – Forward propagation

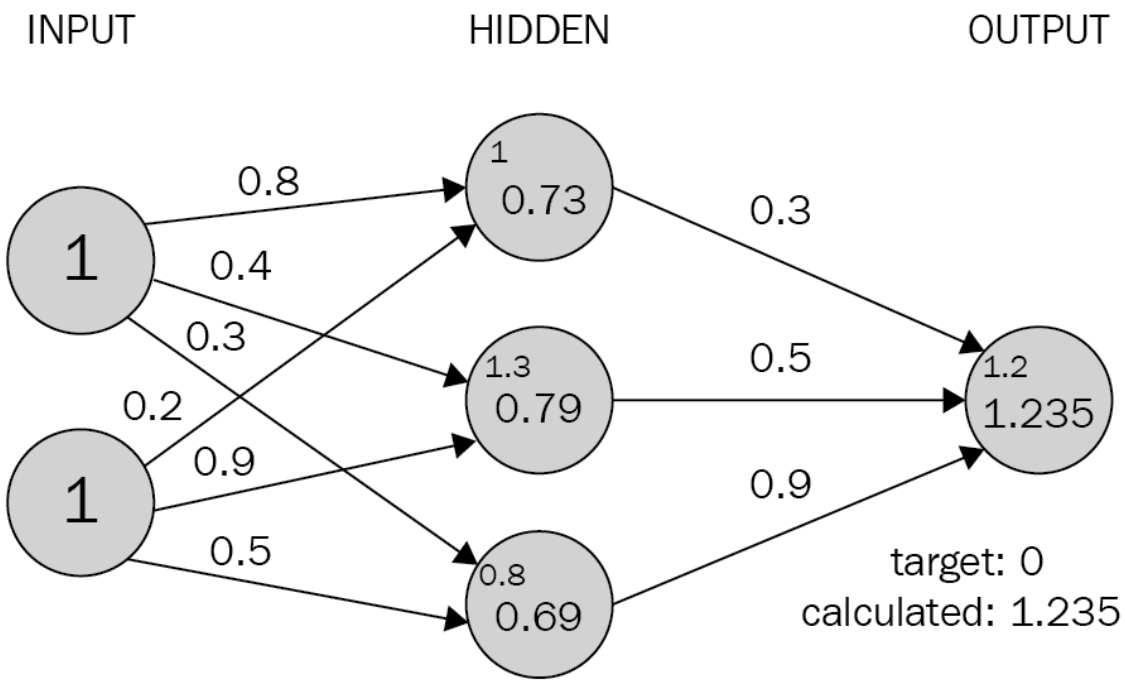


H1=1*0.8 + 1*0.2=1
H2=1*0.4 + 1*0.9=1.3
H3=1*0.3 + 1*0.5=0.8

Packtpub, Feed-forward propagation from scratch in Python

Copyright FAST CAMPUS Corp. All Rights Reserved

Neural Network – Forward propagation

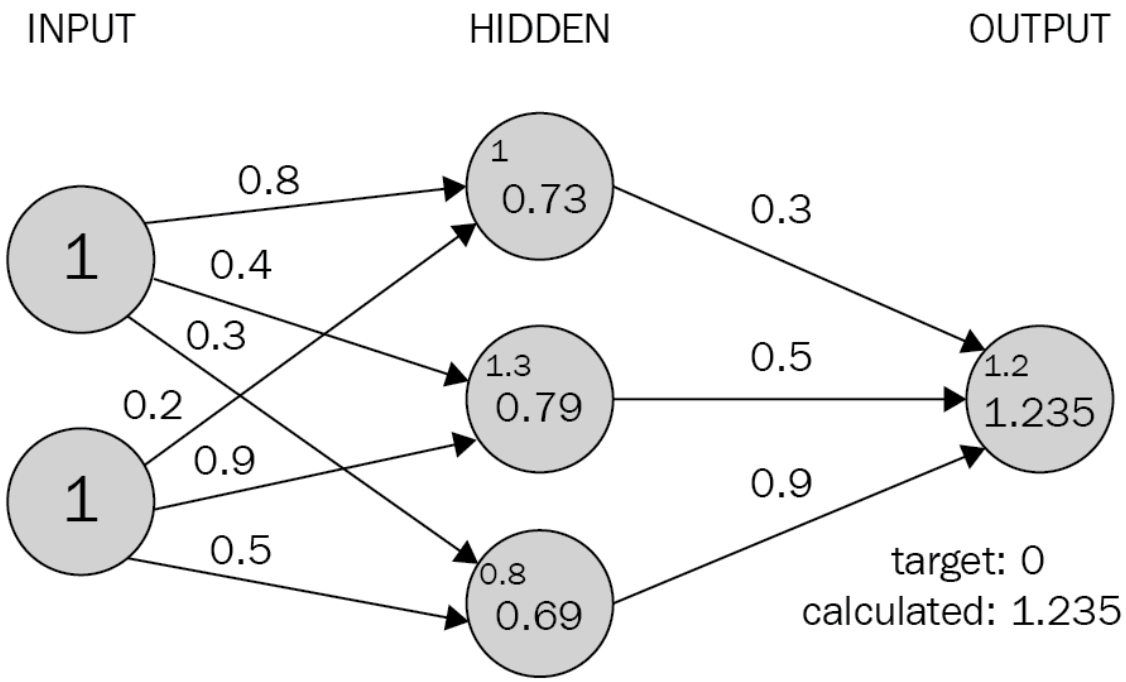


$$0.73 * 0.3 + 0.79 * 0.5 + 0.69 * 0.9 = 1.235$$

Packtpub, Feed-forward propagation from scratch in Python

Copyright FAST CAMPUS Corp. All Rights Reserved

Neural Network – Forward propagation



0.73 * 0.3 + 0.79 * 0.5 + 0.69 * 0.9 = 1.235

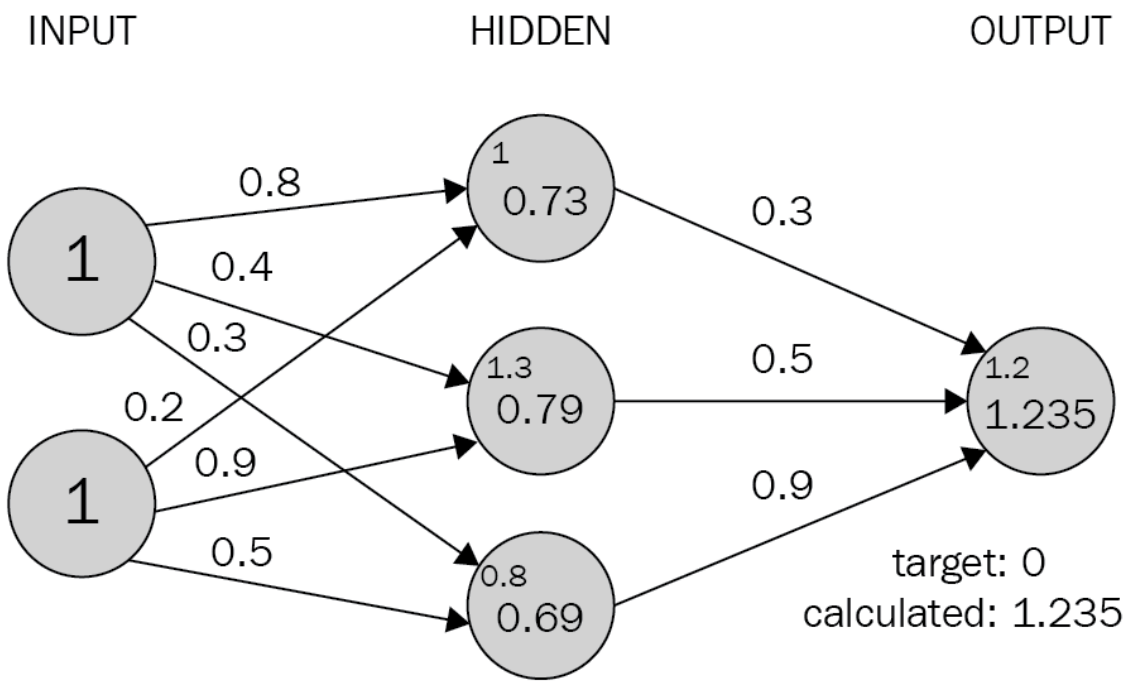
$$y_j = b_j + \sum_i x_i w_{ij} \qquad X = \begin{bmatrix} x_1 & \dots & x_i \end{bmatrix} \quad W = \begin{bmatrix} w_{11} \dots w_{1j} \\ \vdots \quad \ddots \quad \vdots \\ w_{i1} \dots w_{ij} \end{bmatrix} \qquad B = \begin{bmatrix} b_1 & \dots & b_j \end{bmatrix}$$

$$Y = XW + B$$

Packtpub, Feed-forward propagation from scratch in Python

Copyright FAST CAMPUS Corp. All Rights Reserved

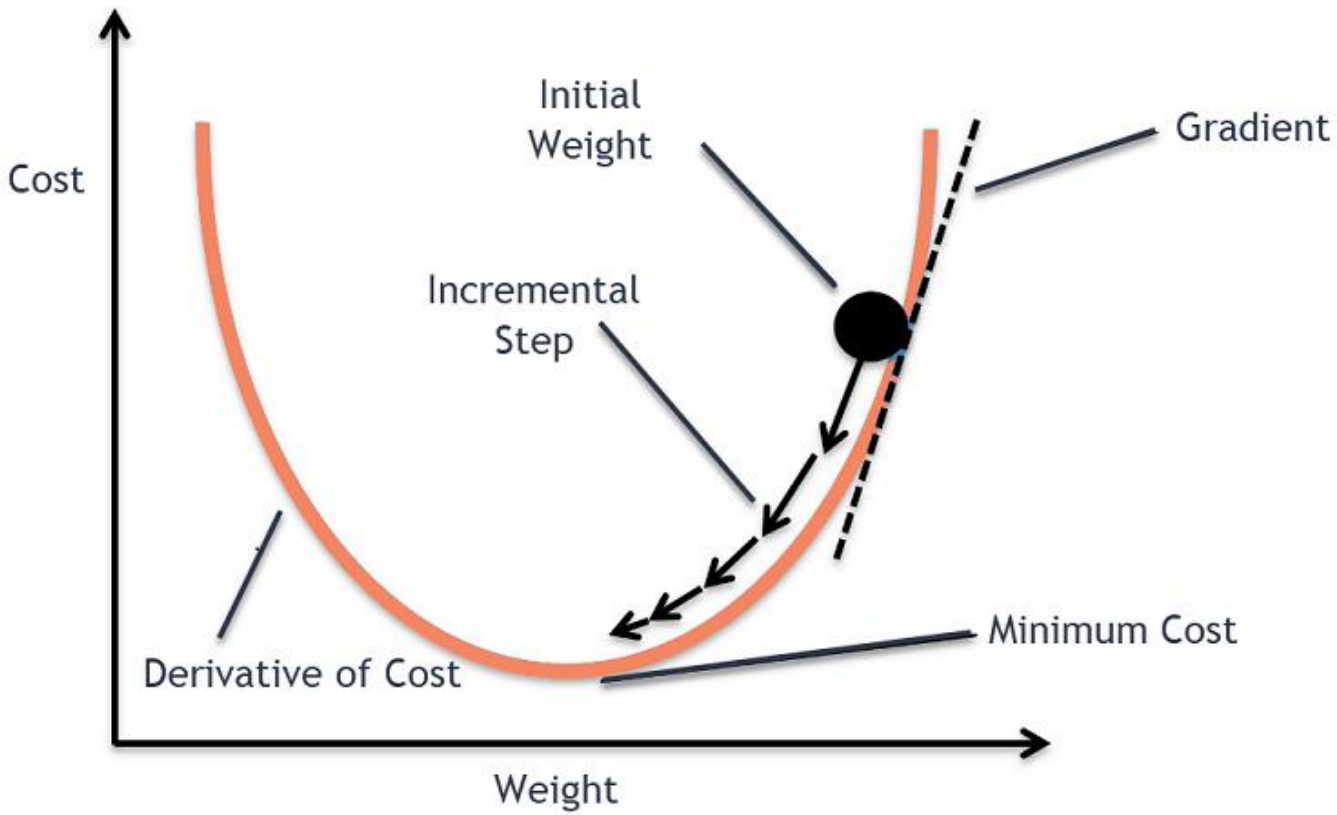
Neural Network – Back propagation & Gradient Descent



$$L(W,b) = \frac{1}{N}\sum_{i}^{N}(\hat{y} - y)^2$$

$$\Delta w_i = -\alpha \frac{dL}{dw_i}$$

$$w_{next} = w + \Delta w$$



Sushant Patrikar, 2019, Batch, Mini Batch & Stochastic Gradient Descent
https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1

Copyright FAST CAMPUS Corp. All Rights Reserved

Internal Neural Network – mapping code from equation

Dive

X				wh			bh			hidden_layer_input			hidden_layer_activations			wout	bout	output	y	E
1	0	1	0																1	
1	0	1	1																1	
0	1	0	1																0	

X				wh			bh			hidden_layer_input			hidden_layer_activations			wout	bout	output	y	E
1	0	1	0	0.42	0.88	0.55	0.46	0.72	0.08							0.30	0.69		1	
1	0	1	1	0.10	0.73	0.68										0.25			1	
0	1	0	1	0.60	0.18	0.47										0.23			0	
				0.92	0.11	0.52														

X				wh			bh			hidden_layer_input			hidden_layer_activations			wout	bout	output	y	E
1	0	1	0	0.42	0.88	0.55	0.46	0.72	0.08	1.48	1.78	1.10				0.30	0.69		1	
1	0	1	1	0.10	0.73	0.68				2.40	1.89	1.61				0.25			1	
0	1	0	1	0.60	0.18	0.47				1.48	1.56	1.27				0.23			0	
				0.92	0.11	0.52														

X				wh			bh			hidden_layer_input			hidden_layer_activations			wout	bout	output	y	E
1	0	1	0	0.42	0.88	0.55	0.46	0.72	0.08	1.48	1.78	1.10	0.81	0.86	0.75	0.30	0.69		1	
1	0	1	1	0.10	0.73	0.68				2.40	1.89	1.61	0.92	0.87	0.83	0.25			1	
0	1	0	1	0.60	0.18	0.47				1.48	1.56	1.27	0.81	0.83	0.78	0.23			0	
				0.92	0.11	0.52														

X				wh			bh			hidden_layer_input			hidden_layer_activations			wout	bout	output	y	E
1	0	1	0	0.42	0.88	0.55	0.46	0.72	0.08	1.48	1.78	1.10	0.81	0.86	0.75	0.30	0.69	0.79	1	
1	0	1	1	0.10	0.73	0.68				2.40	1.89	1.61	0.92	0.87	0.83	0.25		0.80	1	
0	1	0	1	0.60	0.18	0.47				1.48	1.56	1.27	0.81	0.83	0.78	0.23		0.79	0	
				0.92	0.11	0.52														

Sunil Ray, 2017, Understanding and coding Neural Networks From Scratch in Python and R, https://medium.com/analytics-vidhya/understanding-and-coding-neural-networks-from-scratch-in-python-and-r-b8c760f0ad1c

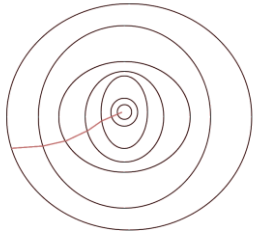
Copyright FAST CAMPUS Corp. All Rights Reserved

Epoch & Batch

Fast campus

Epoch

Updating weight factor using gradient descent (GD).
Train dataset / Batch



Batch gradient descent (BGD)

$$L(W,b) = \frac{1}{N}\sum_{i}^{N}(\hat{y}-y)^2 \quad \Delta w_i = -\alpha\frac{dL}{dw_i}$$

$$w_{next_step} = w_{cur_step} - \alpha\frac{dL}{dw_i}$$

for index in range(epoch):
  $\hat{y}$=Tensor(All dataset) * NN in GPU
  Loss = Sum $(\hat{y}-y)$ / N
  Back propagation (Loss)

Batch size

1 Epoch

Dataset number
= Sample = 10

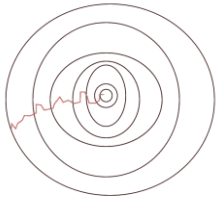
Mini-batch gradient descent (MB-GD. batch size or subdivion)

for index in range(epoch):
  for mini_batch in (Batch / subidivision):
    $\hat{y}$=Tensor(mini_batch.dataset) * NN in GPU
    Loss = Sum $(\hat{y}-y)$ / N
    Back propagation (Loss) to update NN weights using Graident Descent (GD)

Dataset = 10, Batch size = 2 (GD), Steps of 1 Epoch = 5

Mini-batch size = 2

Stochastic gradient descent (SDG) 확률적 경사하강



Batch Gradient Descent: Batch Size = Size of Training Set
Stochastic Gradient Descent: Batch Size = 1
Mini-Batch Gradient Descent: 1 < Batch Size < Size of Training Set.

Copyright FAST CAMPUS Corp. All Rights Reserved

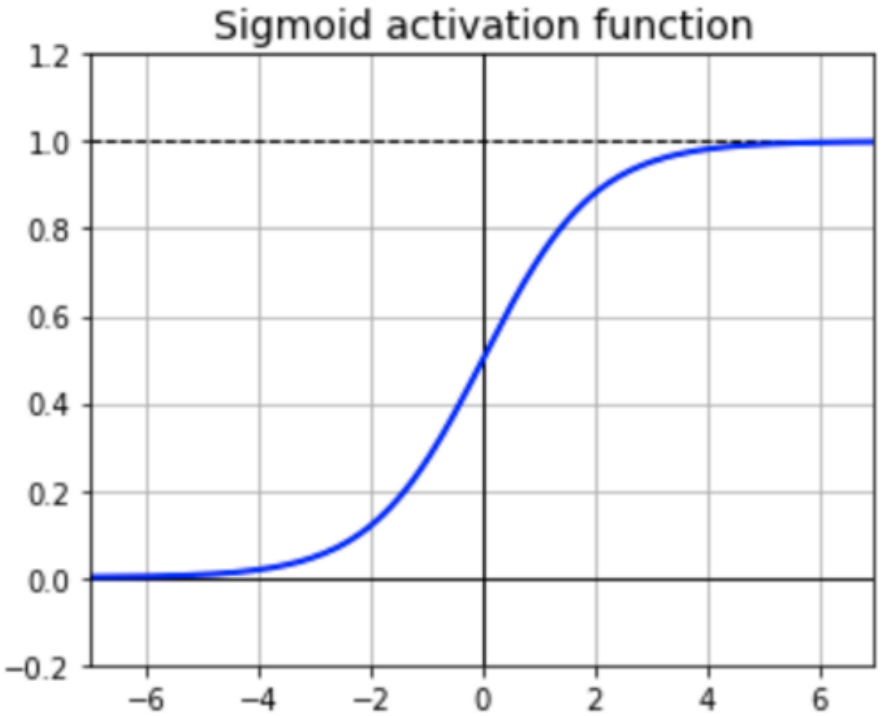
Activation Function

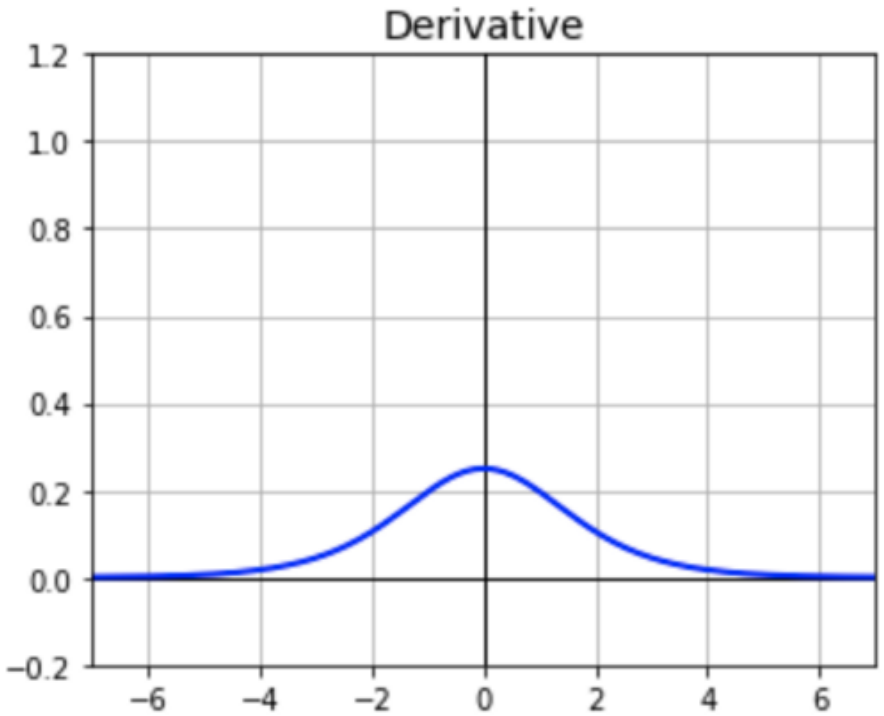
Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Outlier removal
Binary classification

Vanishing gradient
Computationally expensive
The output is not zero centered





B. Chen, 2021, 7 popular activation functions you should know in Deep Learning and how to use them with Keras and TensorFlow 2, https://towardsdatascience.com/7-popular-activation-functions-you-should-know-in-deep-learning-and-how-to-use-them-with-keras-and-27b4d838dfe6

Copyright FAST CAMPUS Corp. All Rights Reserved

Activation Function

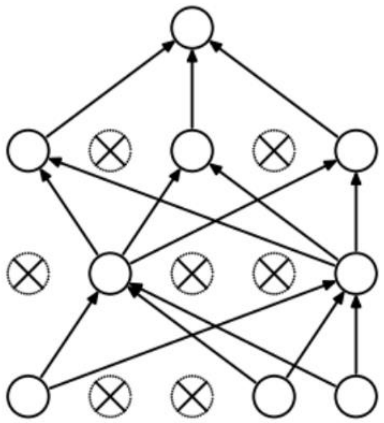
ReLU (Rectified Linear Unit)

$$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$$

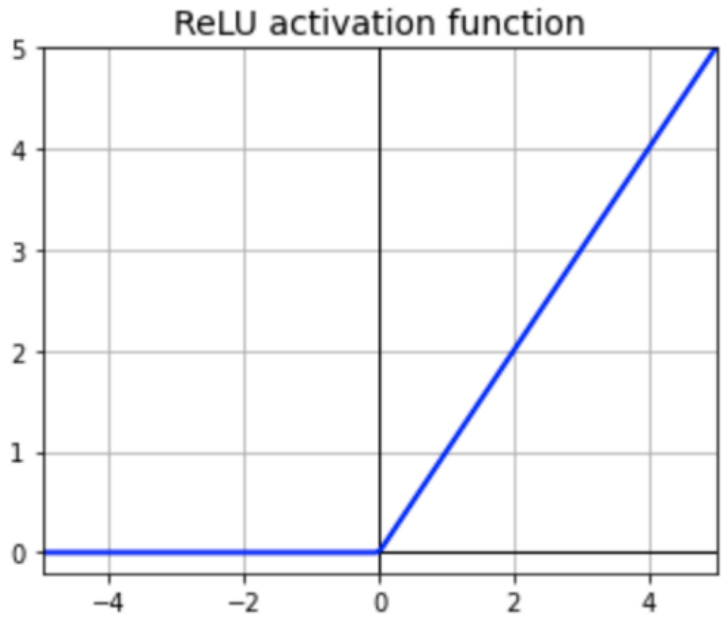
$$f(x) = \max(0, x)$$

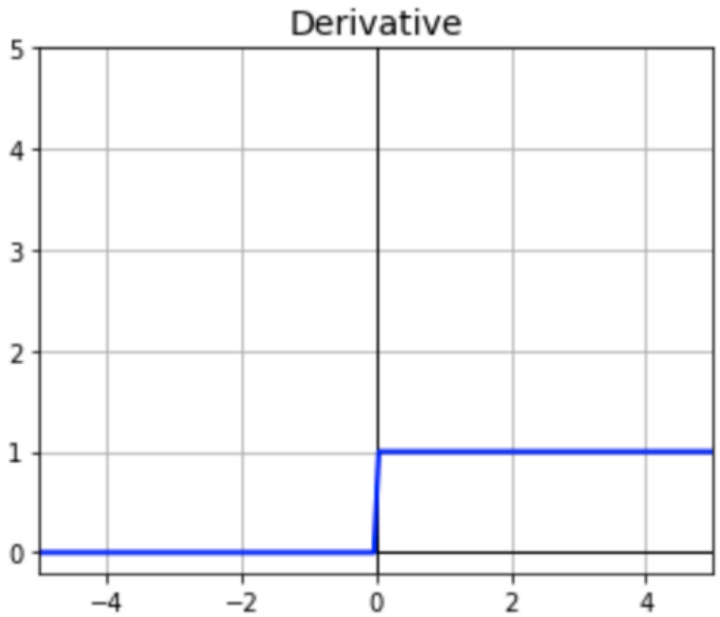
Vanishing gradient solution
Simple, fast.

Dying ReLU



```
def relu(x):
  if x < 0:
    return 0
  else:
return x
```





B. Chen, 2021, 7 popular activation functions you should know in Deep Learning and how to use them with Keras and TensorFlow 2, https://towardsdatascience.com/7-popular-activation-functions-you-should-know-in-deep-learning-and-how-to-use-them-with-keras-and-27b4d838dfe6

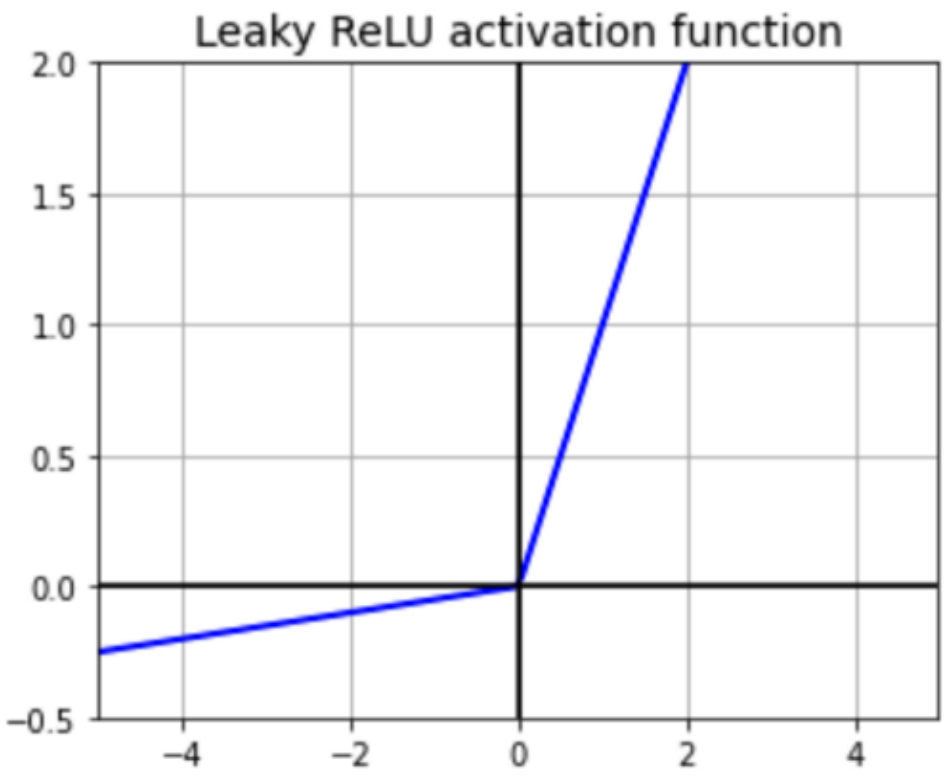
Copyright FAST CAMPUS Corp. All Rights Reserved

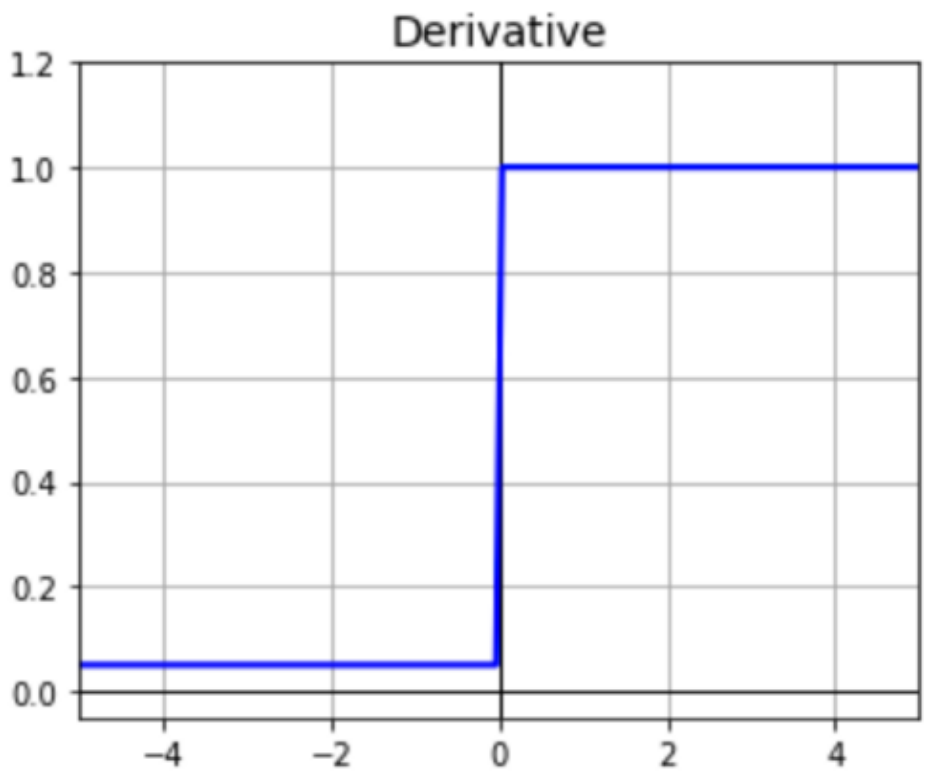
Activation Function

Leaky ReLU

$$f(x) = \max(ax, x)$$

Dying ReLU solution





B. Chen, 2021, 7 popular activation functions you should know in Deep Learning and how to use them with Keras and TensorFlow 2, https://towardsdatascience.com/7-popular-activation-functions-you-should-know-in-deep-learning-and-how-to-use-them-with-keras-and-27b4d838dfe6

Copyright FAST CAMPUS Corp. All Rights Reserved

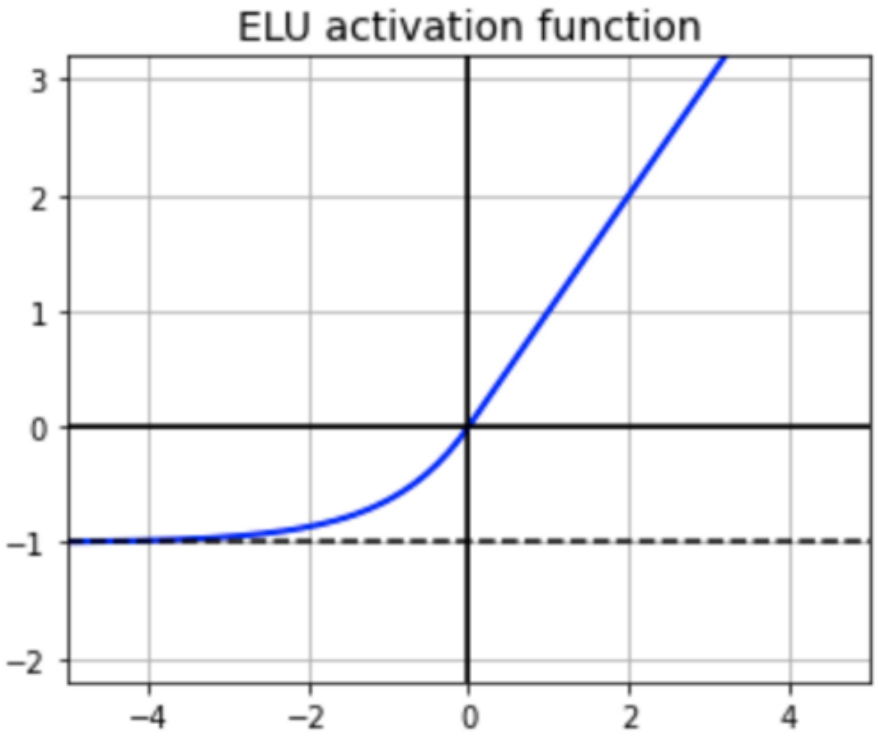
Activation Function

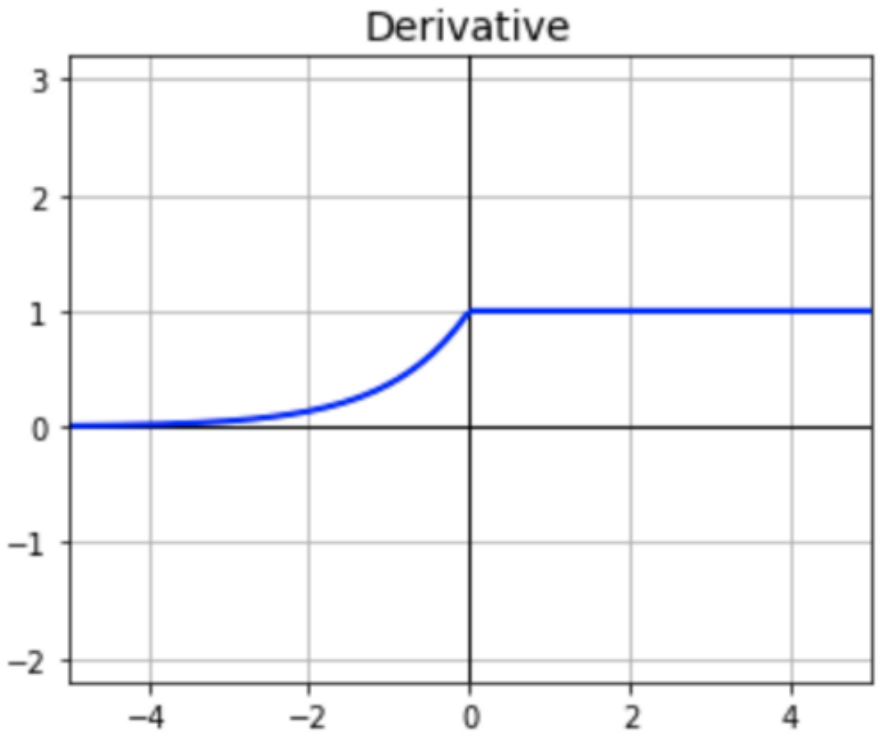
Exponential Linear Unit

$$R(z) = \begin{cases} z & z > 0 \\ \alpha.(e^z - 1) & z <= 0 \end{cases}$$

Perform better than ReLU

```
def elu(z,alpha):
        return z if z >= 0 else alpha*(e^z -1)
```





B. Chen, 2021, 7 popular activation functions you should know in Deep Learning and how to use them with Keras and TensorFlow 2, https://towardsdatascience.com/7-popular-activation-functions-you-should-know-in-deep-learning-and-how-to-use-them-with-keras-and-27b4d838dfe6

Copyright FAST CAMPUS Corp. All Rights Reserved

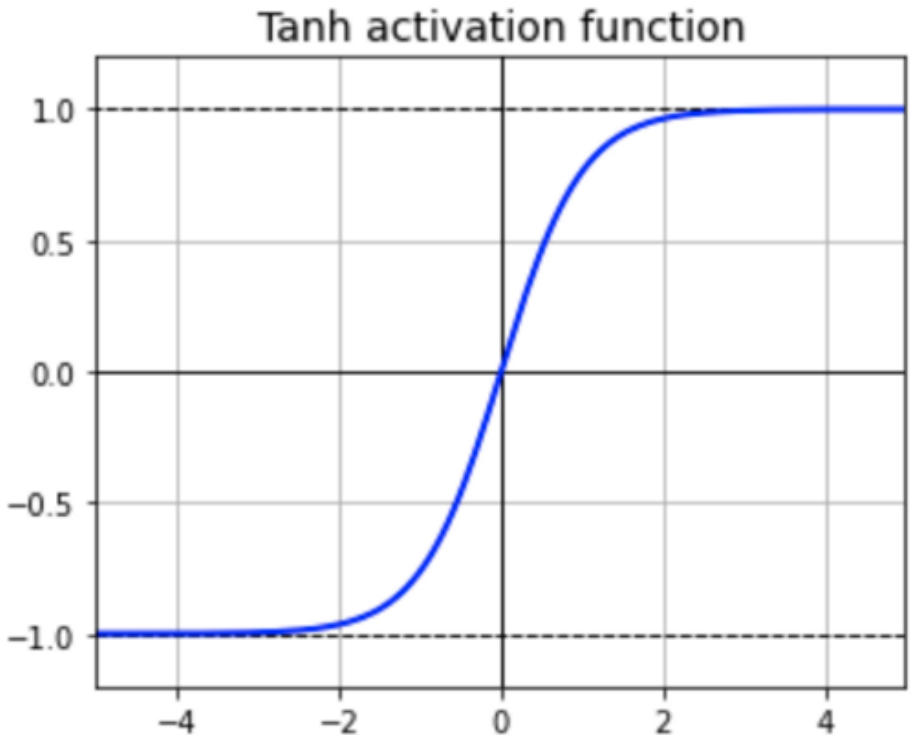
Activation Function

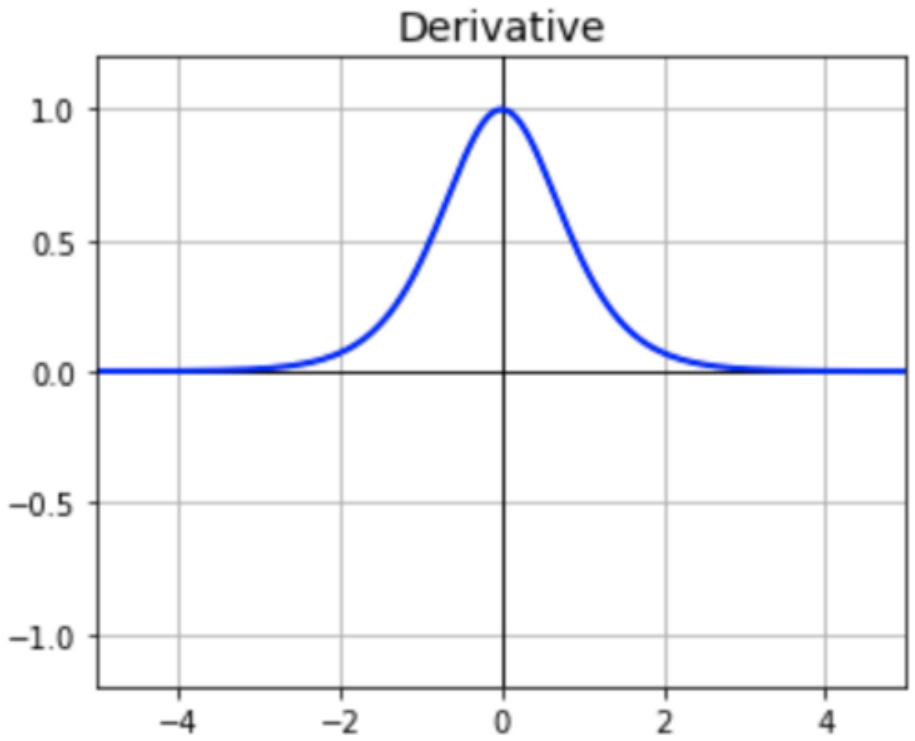
tanh

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Higher gradient value
Big learning steps
Approaching zero average of data

Vanishing (kill) gradient
Computationally expensive
Slowly learning





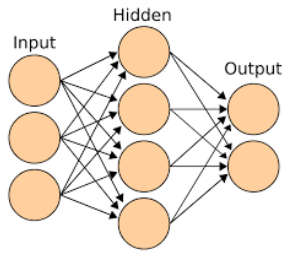
B. Chen, 2021, 7 popular activation functions you should know in Deep Learning and how to use them with Keras and TensorFlow 2, https://towardsdatascience.com/7-popular-activation-functions-you-should-know-in-deep-learning-and-how-to-use-them-with-keras-and-27b4d838dfe6

Copyright FAST CAMPUS Corp. All Rights Reserved

Activation Function

Softmax

Tabby
Dog
Bear



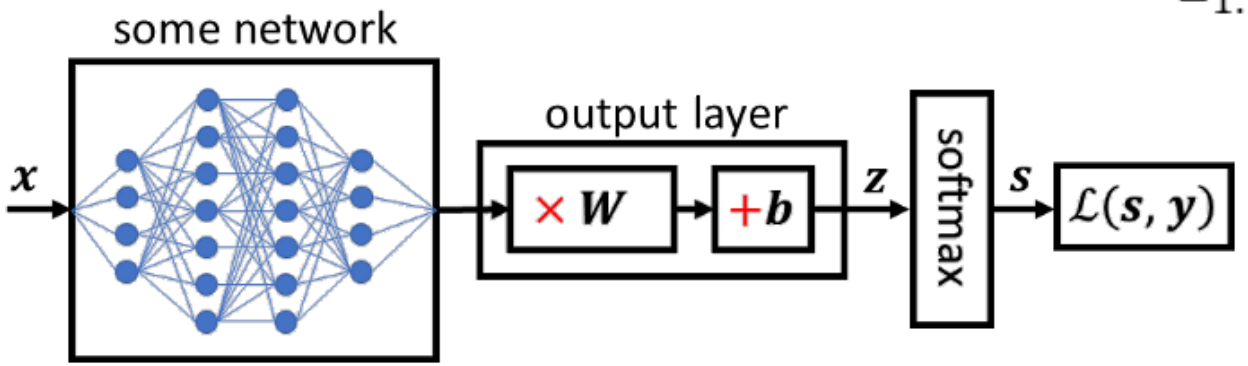
[1, 0, 0]
[0, 1, 0]
[0, 0, 1]

$$s_i = \frac{e^{z_i}}{\sum_{l=1}^{n} e^{z_l}}$$

$$\boldsymbol{z} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} \left\{ \begin{array}{l} 1.1 \\ 2.2 \\ 0.2 \\ -1.7 \end{array} \right. \xrightarrow{s_i = \frac{e^{z_i}}{\sum_l e^{z_l}}} \left. \begin{array}{l} 0.224 \\ 0.672 \\ 0.091 \\ 0.013 \end{array} \right\} \boldsymbol{s} = \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \end{pmatrix}$$

Zi	e^Zi	Si
1.1	3.00417	0.2236
2.2	9.02501	0.6718
0.2	1.22140	0.0909
-1.7	0.18268	0.0136
sum(e^Zi)	13.43	1.000

$$-1.7 < 0.2 < 1.1 < 2.2 \quad \rightarrow \quad 0.013 < 0.091 < 0.224 < 0.672$$



Thomas Kurbiel, 2021, Derivative of the Softmax Function and the Categorical Cross-Entropy Loss, https://towardsdatascience.com/derivative-of-the-softmax-function-and-the-categorical-cross-entropy-loss-ffceefc081d1

Copyright FAST CAMPUS Corp. All Rights Reserved

Loss Function

Mean Square Error (L2-norm) Loss

$$L = (y - f(x))^2$$

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

Mean Absolute Error (L1-norm) Loss

$$MAE = \frac{\sum_{i=1}^{n} \mid y_i - \hat{y}_i \mid}{n}$$

Root Mean Square Error Loss (L2 Loss)

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

```
import numpy as np
def root_mean_squared_error(act, pred):

    diff = pred - act
    differences_squared = diff ** 2
    mean_diff = differences_squared.mean()
    rmse_val = np.sqrt(mean_diff)
    return rmse_val

act = np.array([1.1,2,1.7])
pred = np.array([1,1.7,1.5])

print(root_mean_squared_error(act, pred))
```

0.21602468994692867

Jayant Verma, 2022, Loss Functions in Python - Easy Implementation, https://www.digitalocean.com/community/tutorials/loss-functions-in-python

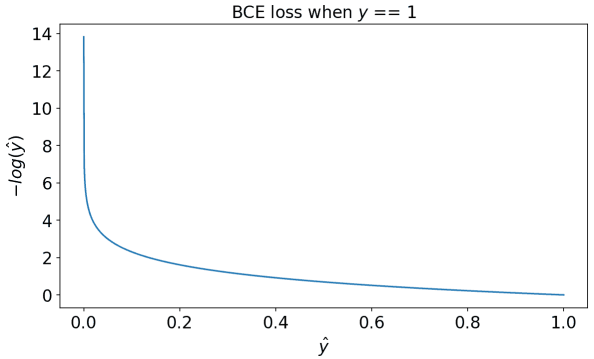
Copyright FAST CAMPUS Corp. All Rights Reserved

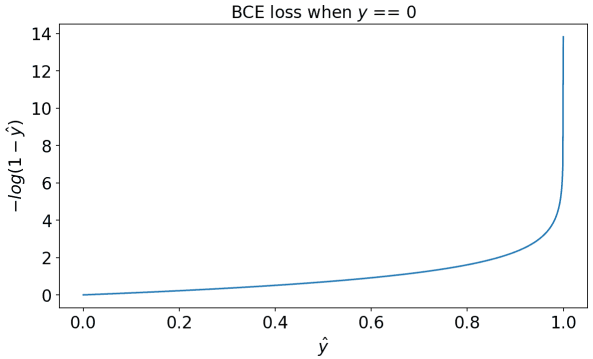
Loss Function

Binary Cross Entropy Loss (BCE)

Binary classification loss function
Easy loss penalty calculation based on distance

$$L = -\frac{1}{n}\sum_{i}^{n} y_i \cdot log\hat{y}_i + (1 - y_i) \cdot log(1 - \hat{y}_i)$$





y (label)	ŷ (pred)	term1	term2	Log L	N
1	0.9	-0.10536	0.00000	-0.105361	
0	0.1	0.00000	-0.10536	-0.105361	
				0.105361	2

y (label)	ŷ (pred)	term1	term2	Log L	N
1	0.1	-2.30259	0.00000	-2.302585	
0	0.9	0.00000	-2.30259	-2.302585	
				2.302585	2

```
def safe_log(x):
    return 0 if x == 0 else np.log(x)

def binary_cross_entropy(y, y_hat):
    total = 0
    for curr_y, curr_y_hat in zip(y, y_hat):
        total += (curr_y * safe_log(curr_y_hat)
                  + (1 - curr_y) * safe_log(1 - curr_y_hat))
    return - total / len(y)

print(binary_cross_entropy(y=[1, 0], y_hat=[0.9, 0.1]))
print(binary_cross_entropy(y=[1, 0], y_hat=[0.1, 0.9]))
```

Master Machine Learning: Logistic Regression From Scratch With Python, 2021.3,
https://betterdatascience.com/mml-logistic-regression/
Better data science, Binary Cross Entropy Explained

Copyright FAST CAMPUS Corp. All Rights Reserved

Loss Function

Categorical Cross entropy

$$L = \frac{1}{N}\sum_{i=0}^{N}\sum_{j=0}^{C}(y_{ij}\log(\hat{y}_{ij}))$$

C = classes
y = actual value

One hot encoding

Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

no	y			y^			y * log (y^)			Loss
1	0	0	1	0	0	1	0.0000	0.0000	0.0001	0.0001
2	0	0	1	0	0	0.5	0.0000	0.0000	-0.6929	-0.6929
3	0	0	1	0	0	0	0.0000	0.0000	-9.2103	-9.2103
4	0	1	1	0	0	1	0.0000	-9.2103	0.0001	-9.2102
5	0	1	1	0	0	0.5	0.0000	-9.2103	-0.6929	-9.9033
6	0	1	1	0	0	0	0.0000	-9.2103	-9.2103	-18.4207
7	1	1	1	0	0	1	-9.2103	-9.2103	0.0001	-18.4206
8	1	1	1	0	0	0.5	-9.2103	-9.2103	-0.6929	-19.1136
9	1	1	1	0	0	0	-9.2103	-9.2103	-9.2103	-27.6310

Copyright FAST CAMPUS Corp. All Rights Reserved

Backpropagation

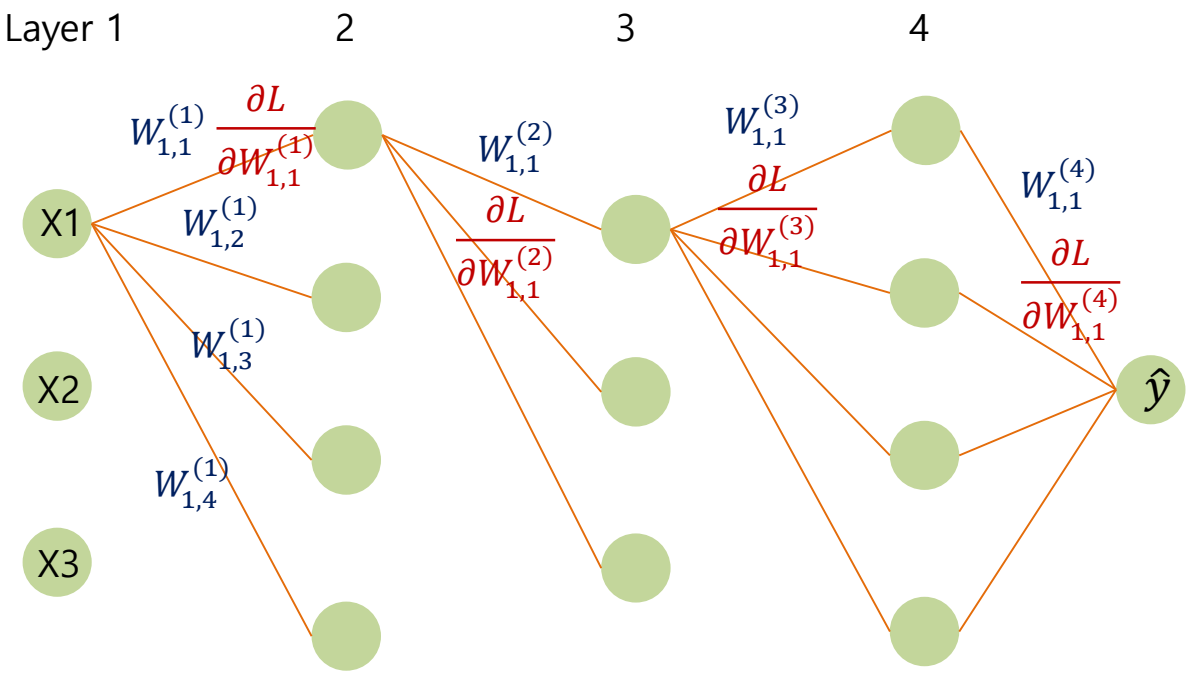
Forward & Backward propagation

$$\hat{y} = \sigma \cdot W^{(3)} \cdot \sigma \cdot W^{(2)} \cdot \sigma \cdot W^{(1)} \cdot (x)$$

$$\hat{y} = L(W, b) = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i) \qquad N = batch\ size$$

$$\Delta w_i = -\alpha \frac{dL}{dw_i}$$

$$w_{next} = w_{current} - \Delta w \qquad w = w - \alpha \frac{dL}{dw_i}$$



Differentiation chain rule

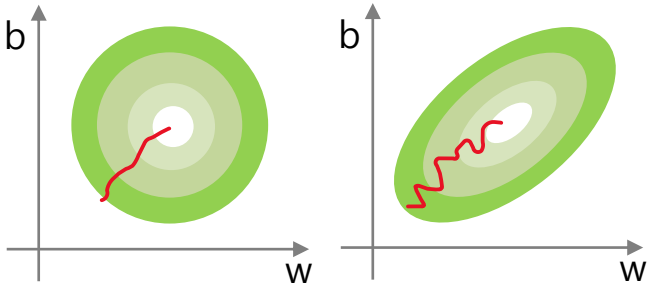
$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx}$$

$$\frac{dL}{dw_{1,1}^{(1)}} = \frac{dL}{dw_{1,1}^{(4)}} \cdot \frac{dw_{1,1}^{(4)}}{dw_{1,1}^{(3)}} \cdot \frac{dw_{1,1}^{(3)}}{dw_{1,1}^{(2)}} \cdot \frac{dw_{1,1}^{(2)}}{dw_{1,1}^{(1)}}$$

Copyright FAST CAMPUS Corp. All Rights Reserved

Backpropagation

Optimization algorithm



Stochastic Gradient Descent

$$W(t+1) = W(t) - \alpha \frac{\partial}{\partial w} Cost(w)$$

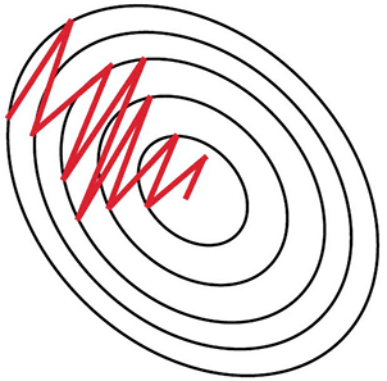
Gradient of divergence

Gradient Descent with Momentum

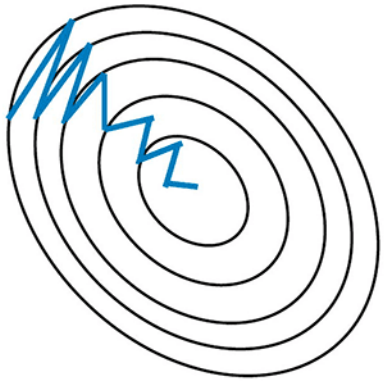
$$V(t) = m * V(t-1) - \alpha \frac{\partial}{\partial w} Cost(w)$$

speed

$$W(t+1) = W(t) + V(t)$$



Stochastic Gradient Descent **without** Momentum



Stochastic Gradient Descent **with** Momentum

ADAM(Adaptive moment estimation) Fast campus

$$M(t) = \beta_1 M(t-1) + (1-\beta_1) \frac{\partial}{\partial w(t)} Cost(w(t))$$

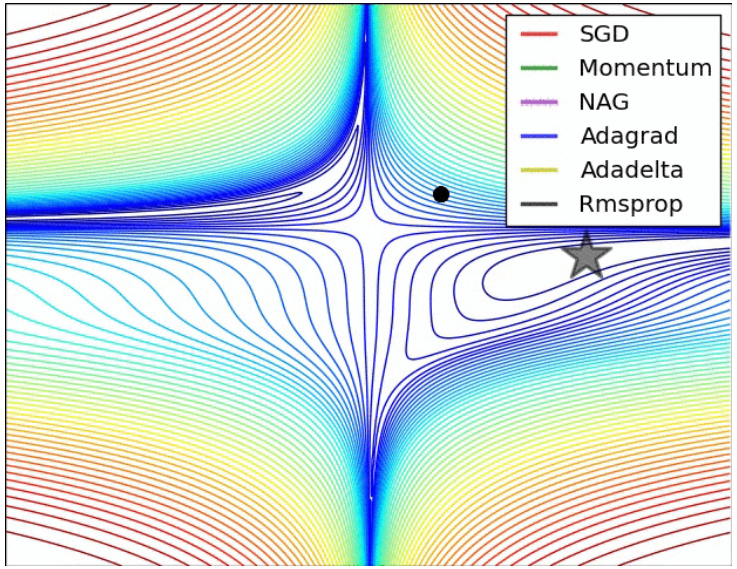
$$V(t) = \beta_2 V(t-1) + (1-\beta_2) \left(\frac{\partial}{\partial w(i)} Cost(w(i)) \right)^2$$

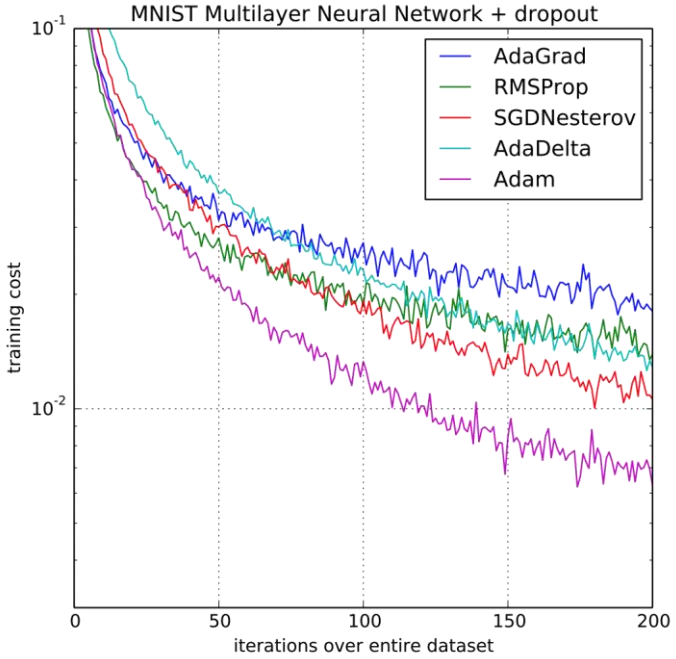
$$\hat{M}(t) = \frac{M(t)}{1-\beta_1{}^t} \qquad \hat{V}(t) = \frac{V(t)}{1-\beta_2{}^t}$$

$$W(t+1) = W(t) - \alpha * \frac{\hat{M}(t)}{\sqrt{\hat{V}(t) + \epsilon}}$$

momentum

Friction





SEBASTIAN RUDER, 2016, An overview of gradient descent optimization algorithms, https://ruder.io/optimizing-gradient-descent/index.html#visualizationofalgorithms, Kingma, Adam: A Method for Stochastic Optimization, https://paperswithcode.com/method/adam

Copyright FAST CAMPUS Corp. All Rights Reserved

Backpropagation

Learning rate and decay

Time-based decay

lr *= (1. / (1. + decay * iterations))

```
learning_rate = 0.1
decay_rate = learning_rate / epochs
momentum = 0.8
sgd = SGD(lr=learning_rate, momentum=momentum, decay=decay_rate,
nesterov=False)
```

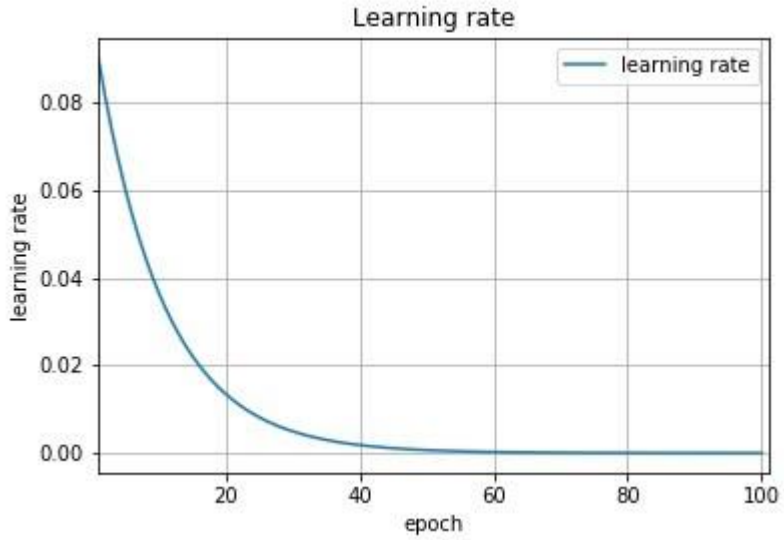
Step decay

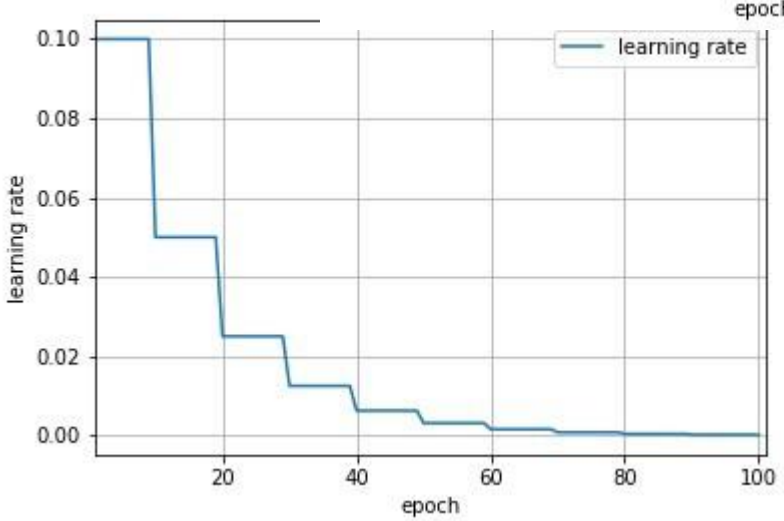
lr = lr0 * drop ^ floor(epoch / epochs_drop)

```
def step_decay(epoch):
   initial_lrate = 0.1
   drop = 0.5
   epochs_drop = 10.0
   lrate = initial_lrate * math.pow(drop,
           math.floor((1+epoch)/epochs_drop))
   return lrate
lrate = LearningRateScheduler(step_decay)
```

Exp decay

```
def exp_decay(epoch):
   initial_lrate = 0.1
   k = 0.1
   lrate = initial_lrate * exp(-k*t) # t=iteration number
   return lrate
```





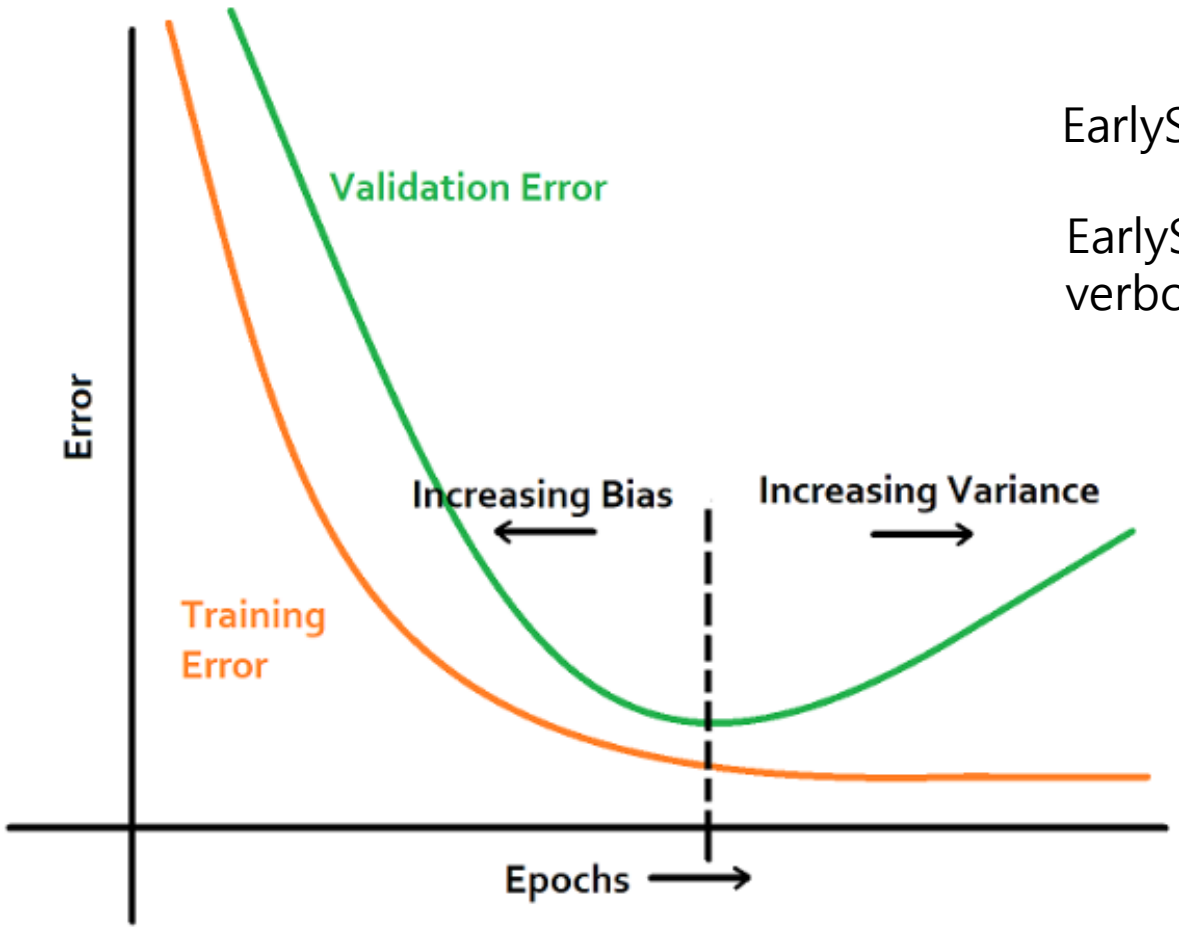
Suki Lau, 2017, Learning Rate Schedules and Adaptive Learning Rate Methods for Deep Learning, https://towardsdatascience.com/learning-rate-schedules-and-adaptive-learning-rate-methods-for-deep-learning-2c8f433990d1

Copyright FAST CAMPUS Corp. All Rights Reserved

Backpropagation

Fast campus

Early stopping



EarlyStopping(monitor='val_loss', mode='min)

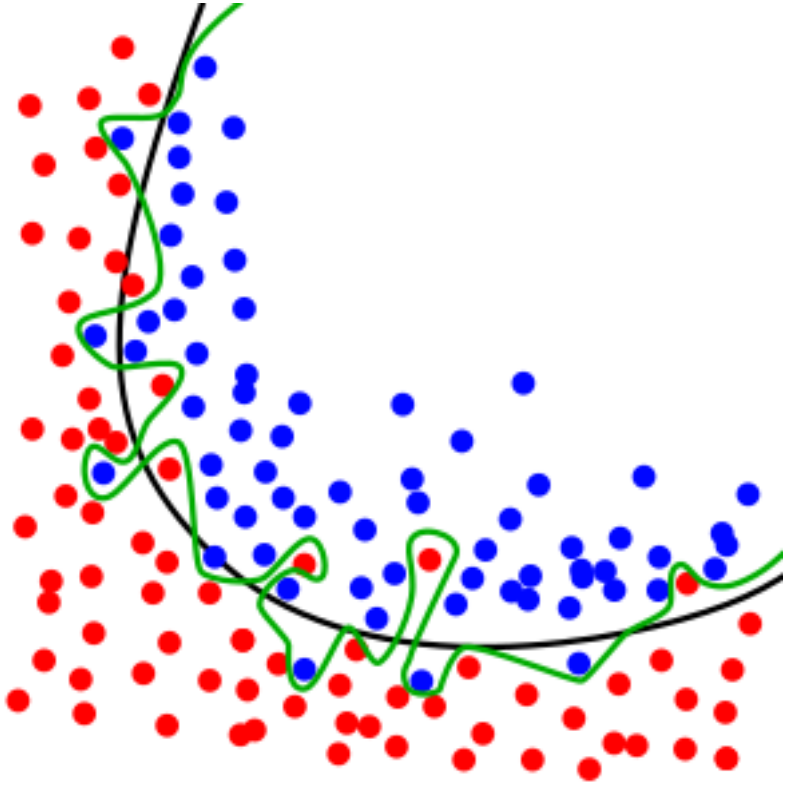
EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=50)

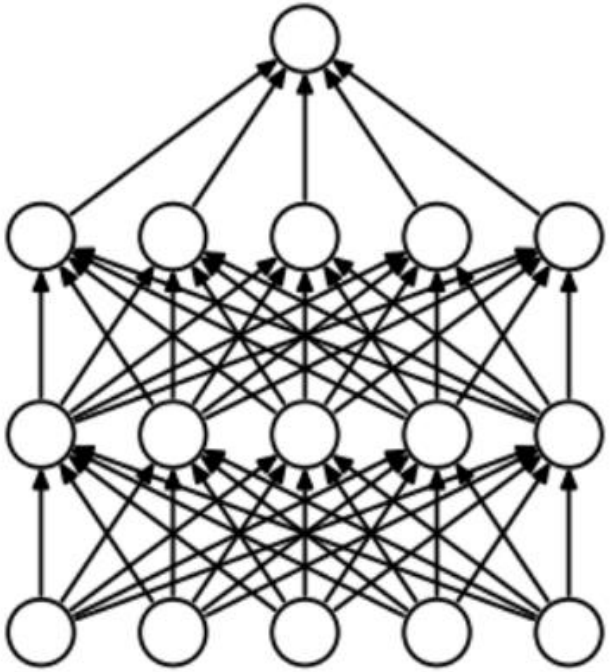
Rahul Jain, 2020, Why "early-stopping" works as Regularization?, https://medium.com/@rahuljain13101999/why-early-stopping-works-as-regularization-b9f0a6c2772

Copyright FAST CAMPUS Corp. All Rights Reserved

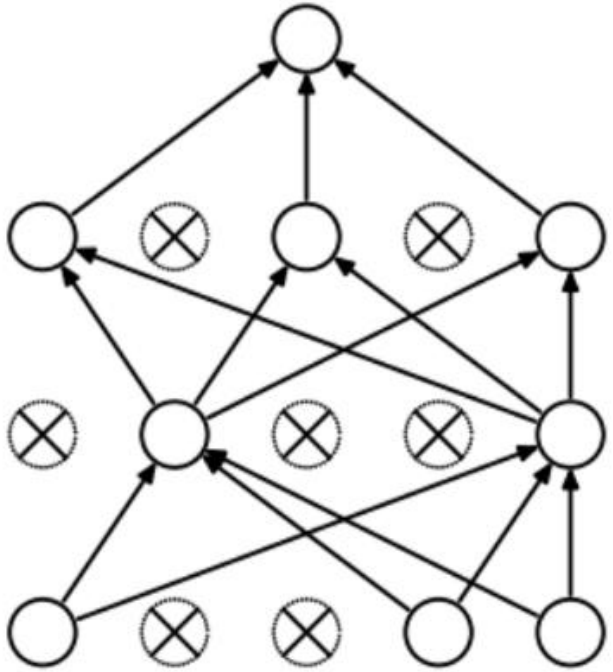
Overfitting solution

Dropout





(a) Standard Neural Net



(b) After applying dropout.

tf.model.add(tf.keras.layers.Dropout(drop_rate))

wikipeidia

Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting", JMLR 2014

Copyright FAST CAMPUS Corp. All Rights Reserved

Overfitting solution

L2 regularization

L1 Norm = sum of abs (vector_2 – vector_1)

$$L = \sum_{i=1}^{n} |y_i - f(x_i)|$$

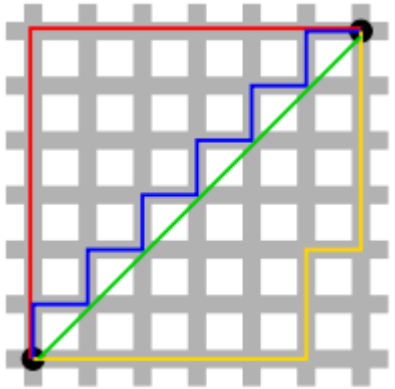
$$Cost = \frac{1}{n}\sum_{i=1}^{n}\{L(y_i, \hat{y}_i) + \frac{\lambda}{2}|w|\}$$

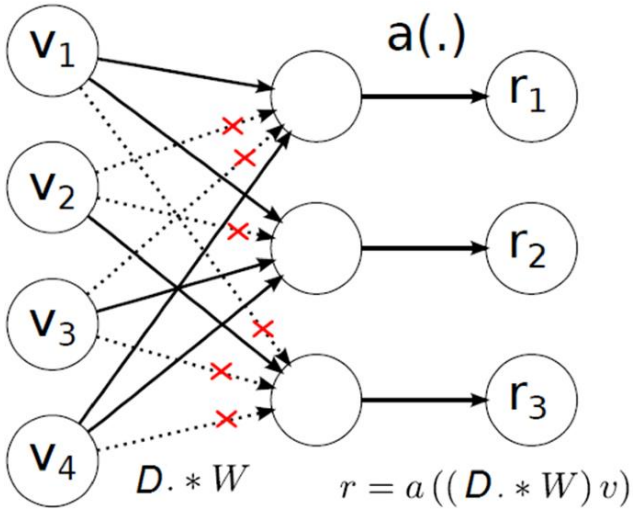
L2 Norm = sum of (vector_2 – vector_1) ^ 2 > Euclidean distance

$$L = \sum_{i=1}^{n}(y_i - f(x_i))^2$$

$$Cost = \frac{1}{n}\sum_{i=1}^{n}\{L(y_i, \hat{y}_i) + \frac{\lambda}{2}|w|^2\}$$

L1 Cost < L2 Cost





Weight decay

$$\Delta w_i = -\alpha \frac{dC}{dw_i}$$

Mode.add(Desnse(units=12,
kernel_regularizer=regularizers.l12(lambda),
activation='relu')

Copyright FAST CAMPUS Corp. All Rights Reserved

Overfitting solution

Dive

Fast campus

Batch normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β

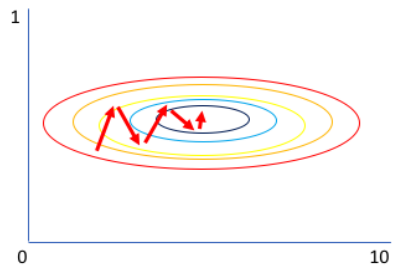
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

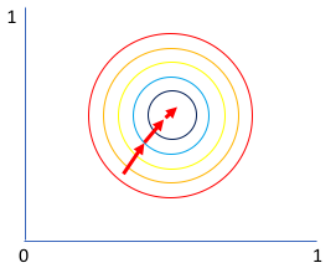
$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$





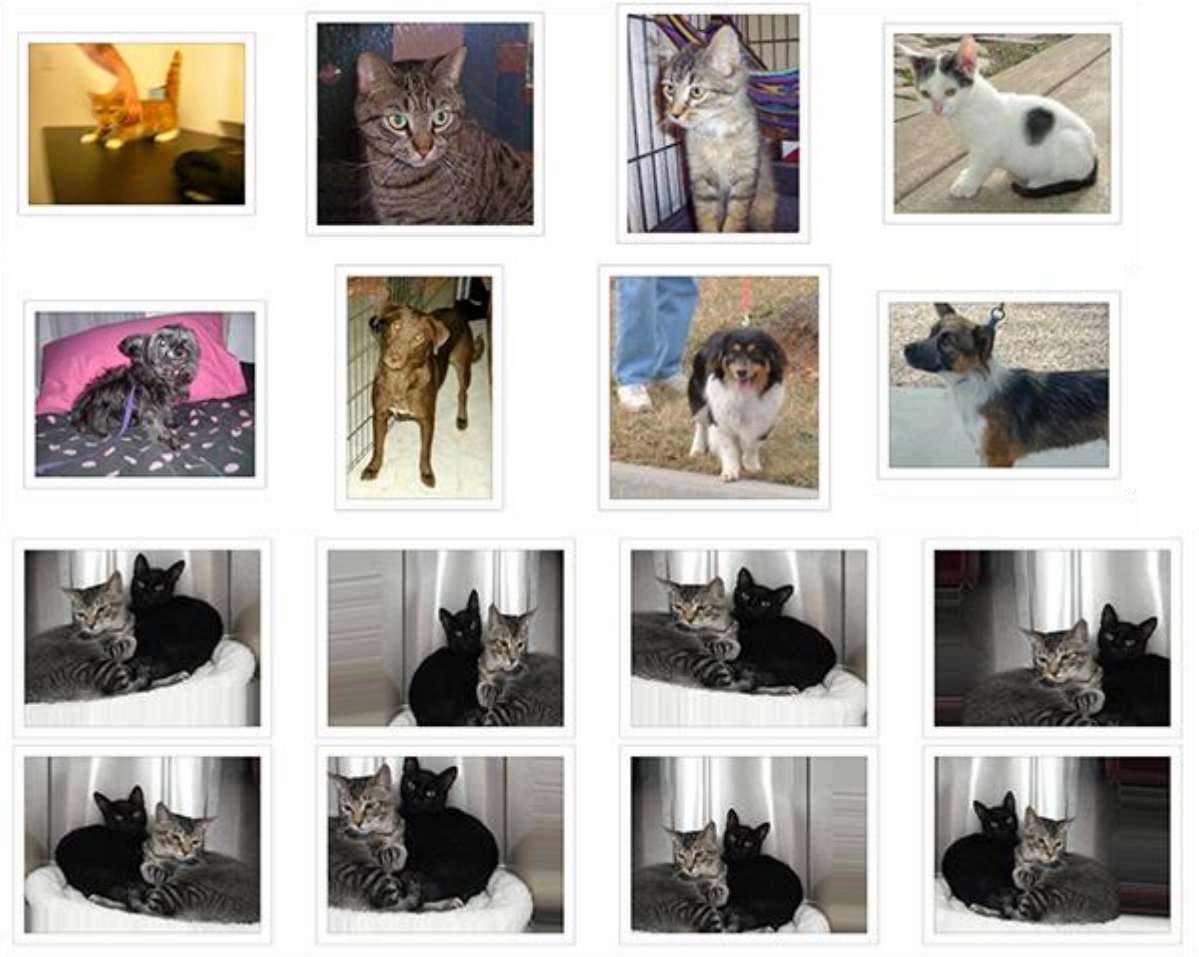
JEREMY JORDAN, 2018, Normalizing your data, https://www.jeremyjordan.me/batch-normalization/
Aakash bindal, Normalization Techniques in Deep Neural Networks, https://medium.com/techspace-usict/normalization-techniques-in-deep-neural-networks-9121bf100d8

```
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.layers import BatchNormalization
from keras.utils import to_categorical
from sklearn.datasets import make_blobs

X, y = make_blobs(n_samples=1000, centers=4, n_features=2,
cluster_std=2, random_state=2)

y = to_categorical(y)
n_train = 800
trainX, testX = X[:n_train, :], X[n_train:, :]
trainy, testy = y[:n_train], y[n_train:]
print(trainX.shape, testX.shape)

model = Sequential()
model.add(Dense(25, input_dim=2, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(8, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.2))
model.add(Dense(4, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

model.summary()
```

Copyright FAST CAMPUS Corp. All Rights Reserved

Overfitting solution

Data Augmentation



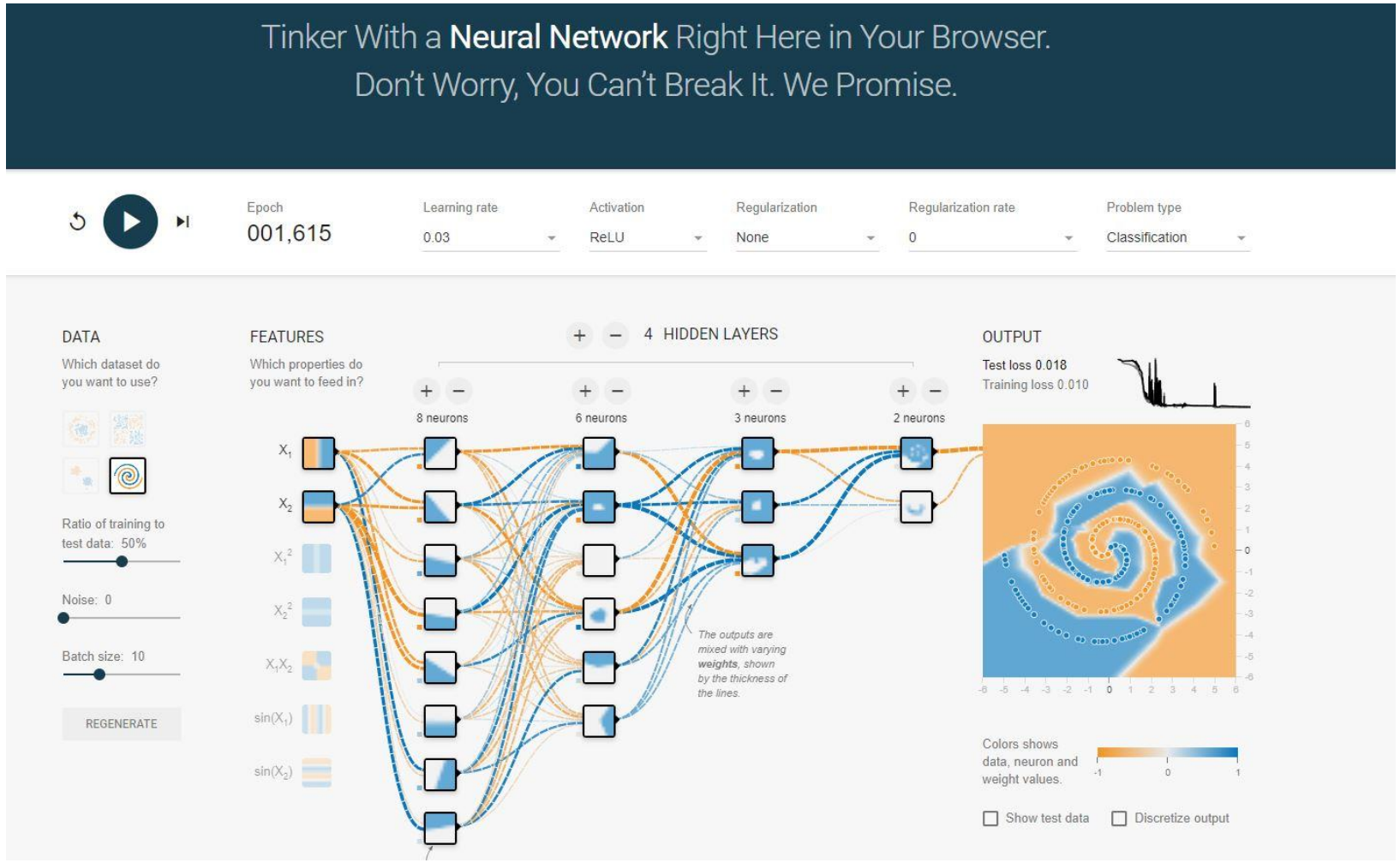
```
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import array_to_img, img_to_array, load_img

datagen = ImageDataGenerator(
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')

img = load_img('./cat0.jpg')  # this is a PIL image
x = img_to_array(img)  # this is a Numpy array with shape (3, 150, 150)
x = x.reshape((1,) + x.shape)  # this is a Numpy array with shape (1, 3, 150, 150)

# the .flow() command below generates batches of randomly transformed images
# and saves the results to the `preview/` directory
i = 0
for batch in datagen.flow(x, batch_size=1,
                          save_to_dir='preview', save_prefix='cat', save_format='jpeg'):
    i += 1
    if i > 20:
        break  # otherwise the generator would loop indefinitely
```

Keras Blog, Building powerful image classification models using very little data,
https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Copyright FAST CAMPUS Corp. All Rights Reserved

Training strategy

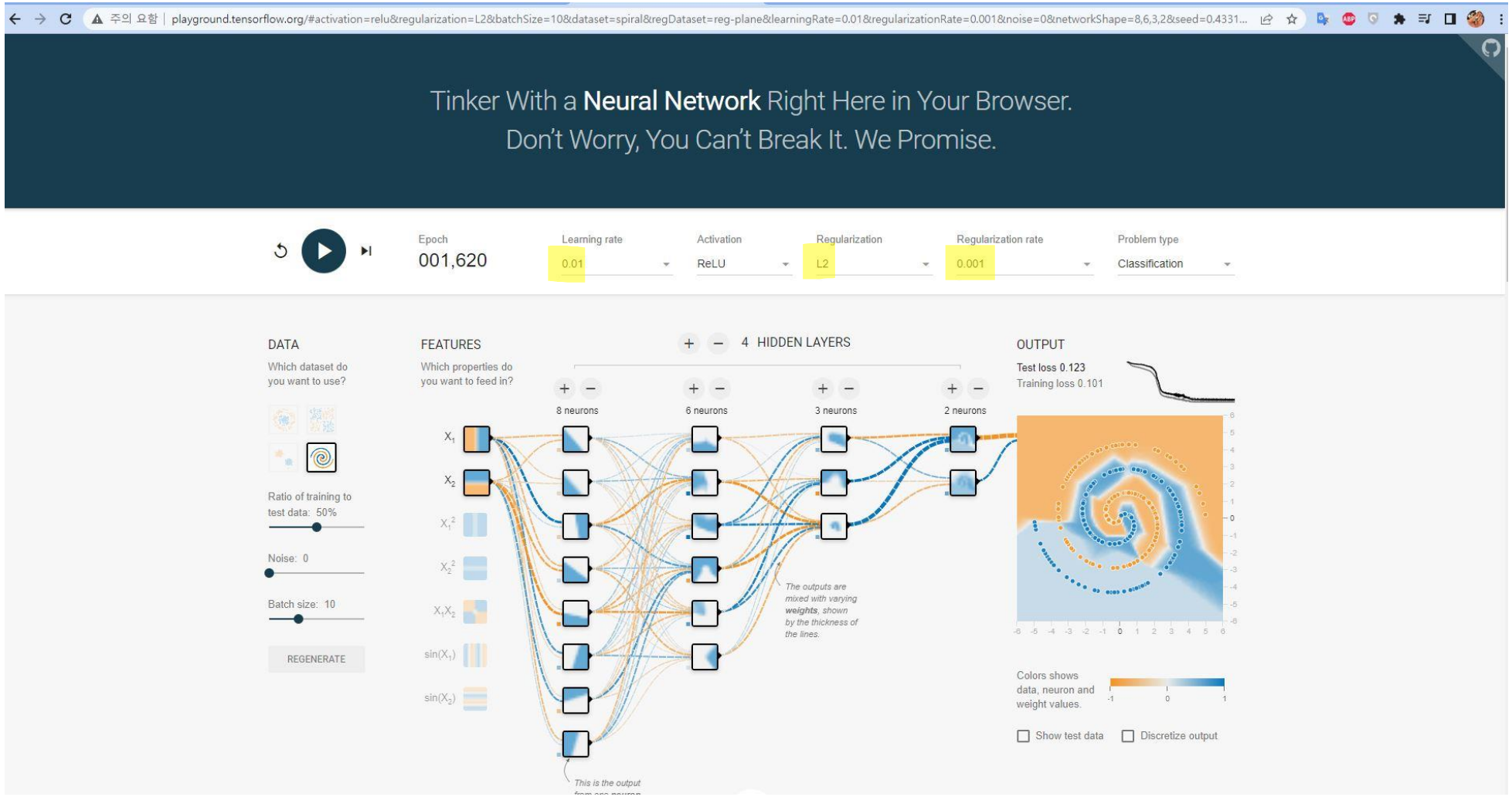
[Neural Network in your browser link]



Tinker With a Neural Network
http://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=10&dataset=spiral&regDataset=reg-plane&learningRate=0.01&regularizationRate=0.001&noise=0&networkShape=8,6,3,2&seed=0.88358&showTestData=false&discretize=false&percTrainData=80&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

Copyright FAST CAMPUS Corp. All Rights Reserved

Traning strategy

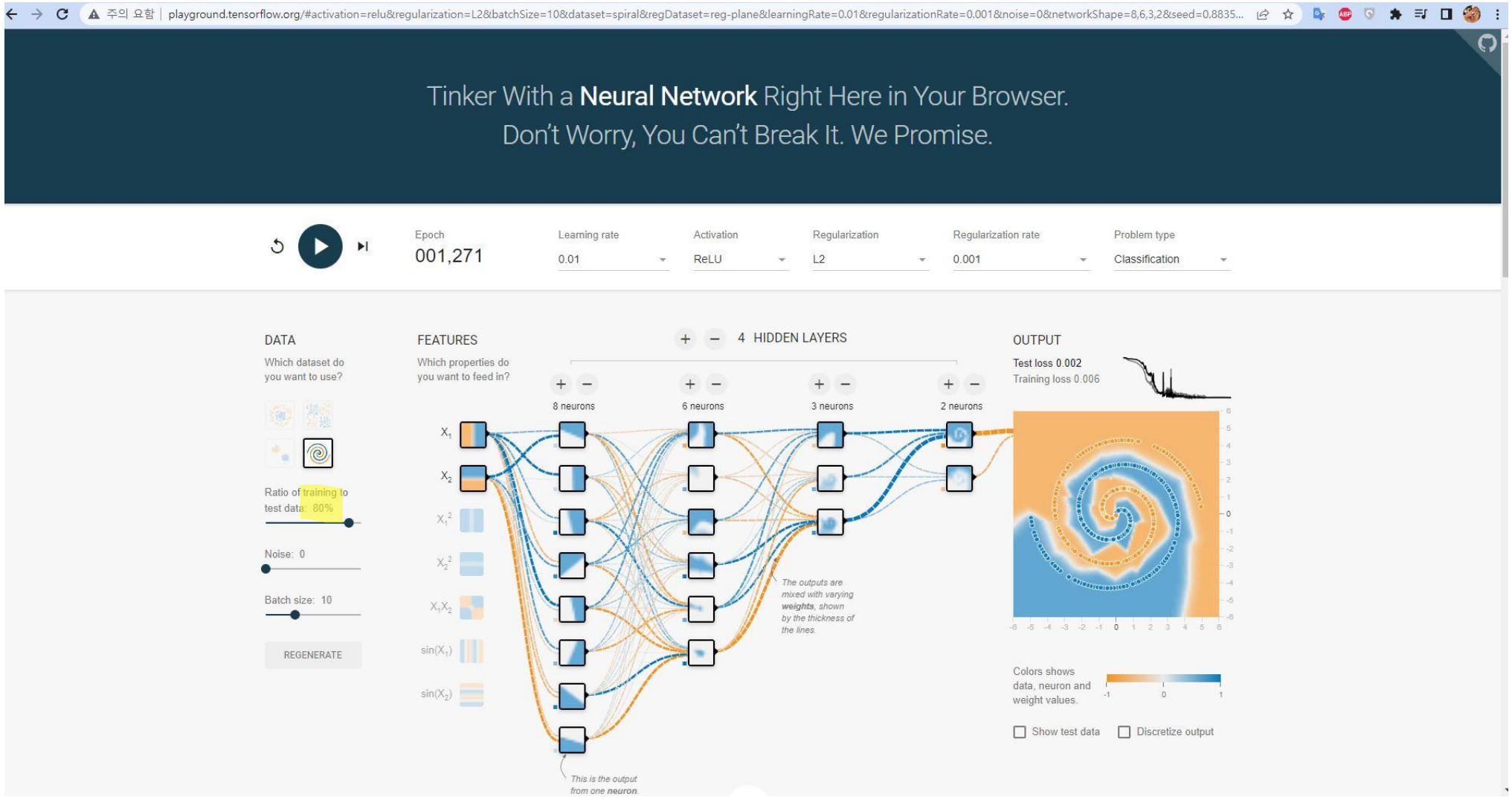


Tinker With a Neural Network
http://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=10&dataset=spiral&regDataset=reg-plane&learningRate=0.01&regularizationRate=0.001&noise=0&networkShape=8,6,3,2&seed=0.88358&showTestData=false&discretize=false&percTrainData=80&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

Copyright FAST CAMPUS Corp. All Rights Reserved

Traning strategy

Fast campus

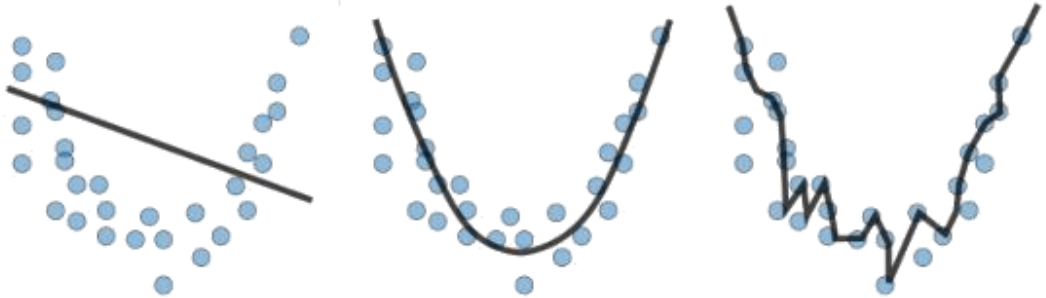
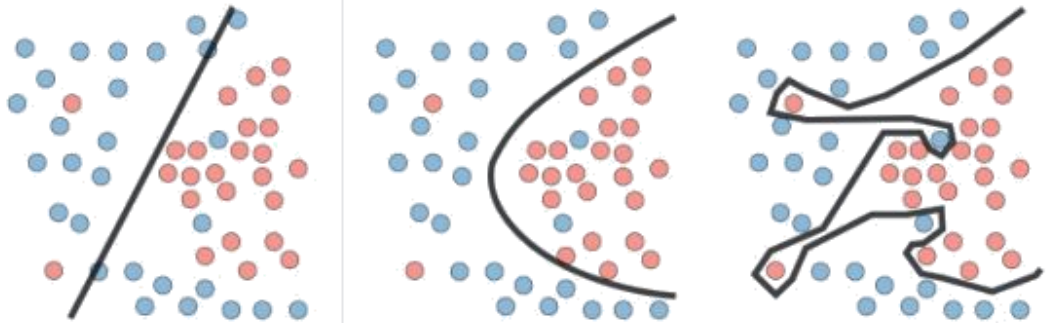


Link

Tinker With a Neural Network
http://playground.tensorflow.org/#activation=relu&regularization=L2&batchSize=10&dataset=spiral&regDataset=reg-plane&learningRate=0.01&regularizationRate=0.001&noise=0&networkShape=8,6,3,2&seed=0.88358&showTestData=false&discretize=false&percTrainData=80&x=true&y=true&xTimesY=false&xSquared=false&ySquared=false&cosX=false&sinX=false&cosY=false&sinY=false&collectStats=false&problem=classification&initZero=false&hideText=false

Copyright FAST CAMPUS Corp. All Rights Reserved

Traning strategy

	Underfitting	Just right	Overfitting
Symptoms	• High training error • Training error close to test error • High bias	• Training error slightly lower than test error	• Very low training error • Training error much lower than test error • High variance
Regression illustration			
Classification illustration			
Deep learning illustration			
Possible remedies	• Complexify model • Add more features • Train longer		• Perform regularization • Get more data

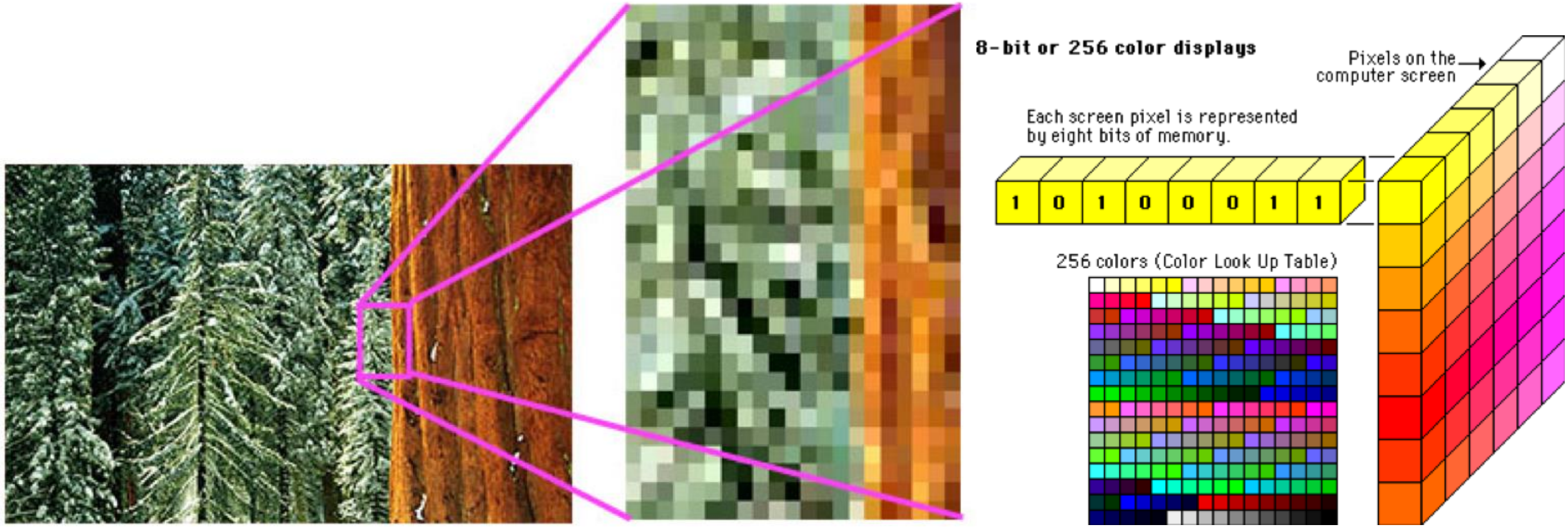
Abhishek Shrivastava, 2020, Underfitting Vs Just right Vs Overfitting in Machine learning, Kaggle(www.kaggle.com/getting-started/166897)

Copyright FAST CAMPUS Corp. All Rights Reserved

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

Vision

Vision – image, pixel

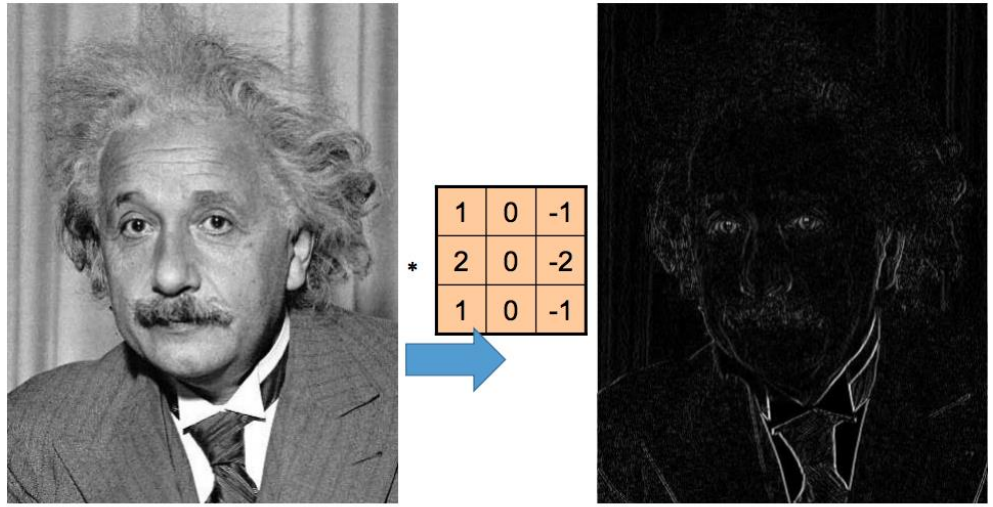


Michelle Venables, 2019, An Overview of Computer Vision, towardsdatascience.com/an-overview-of-computer-vision-1f75c2ab1b66

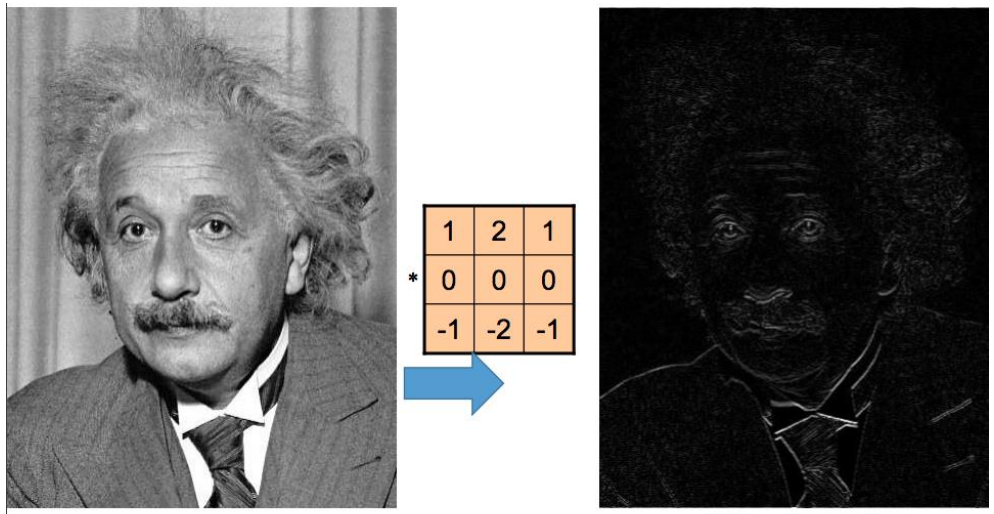
Copyright FAST CAMPUS Corp. All Rights Reserved

Vision

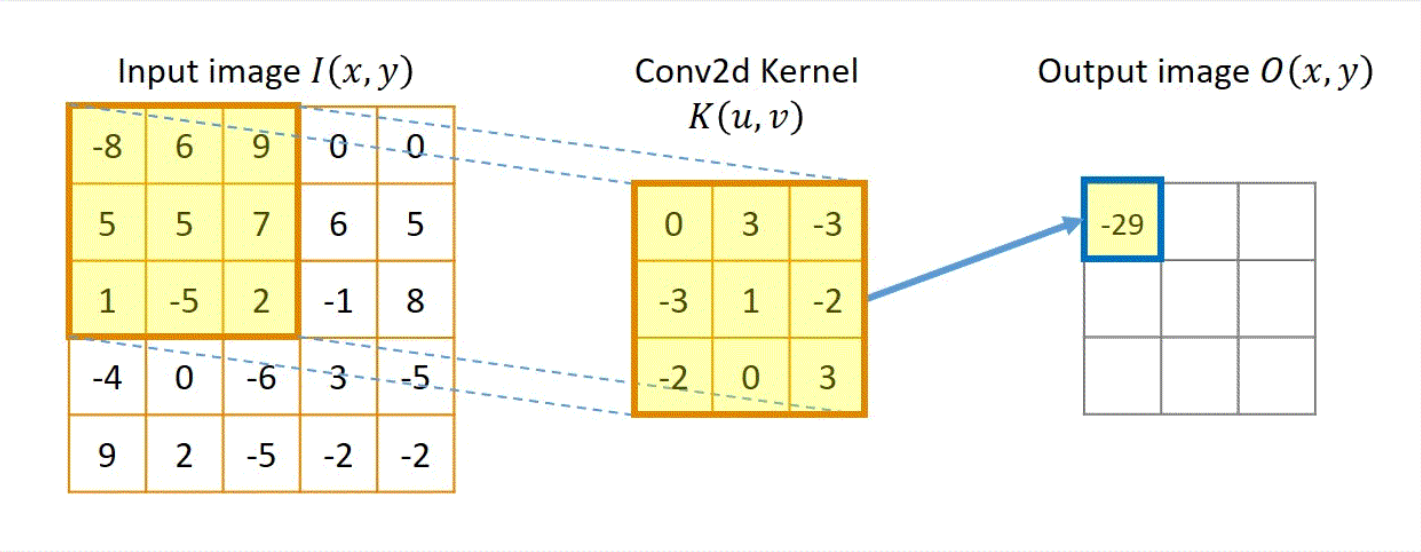
Filter



Vertical Edge
(absolute value)



Horizontal Edge
(absolute value)



$$G_{out}(x, y) = \omega_{out} * F(x, y) = \left(\sum_{\delta x=-k_i}^{k_i} \sum_{\delta y=-k_j}^{k_j} \omega_{out}(\delta x, \delta y) \cdot F(x + \delta x, y + \delta y) \right) + \omega_{out_{bias}}$$

Toronto university, Convolutional Neural Networks, https://www.cs.toronto.edu/~lczhang/360/lec/w04/convnet.html
Xiyun Song, 2020, Understanding Artificial Neural Networks with Hands-on Experience - Part 2. Convolution, Its Gradients and Custom Implementations.
https://coolgpu.github.io/coolgpu_blog/github/pages/2020/10/04/convolution.html

Copyright FAST CAMPUS Corp. All Rights Reserved

Vision

Swdevperestroika, 2014, Image processing with Python, Numpy & Scipy – image convolution,
https://softwaredevelopmentperestroika.wordpress.com/2014/02/11/image-processing-with-python-numpy-scipy-image-convolution/

Copyright FAST CAMPUS Corp. All Rights Reserved

Vision

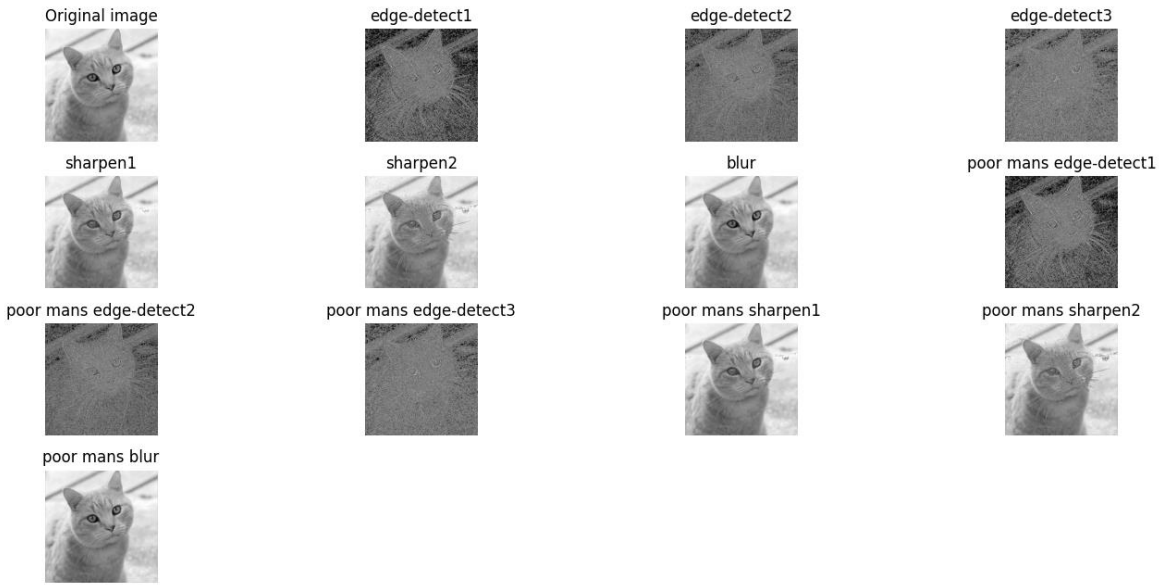
```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.image as mpimg
import scipy.ndimage.filters as filter

def normalize(matrix):
    sum = np.sum(matrix)
    if sum > 0.:
        return matrix / sum
    else:
        return matrix

def neighbors(r,c,supermatrix):
    m = supermatrix[r:r+3,c:c+3]
    return m

def convolve(n,kernel):
    sum = 0
    for (rr,cc),value in np.ndenumerate(n):
        sum += n[rr,cc] * kernel[rr,cc]

    return sum % 255

def poor_mans_convolve(matrix,super,kernel,shape):
    result = np.ndarray(shape,dtype=np.float)

    for (r,c),value in np.ndenumerate(matrix):
        n = neighbors(r,c,super)
        result[r,c] = convolve(n,kernel)

    return result
```

```
fig = plt.figure(figsize=(14, 6.5), dpi=100)

kernel_edge_detect1 = np.array([[1.,0.,-1.],
                                [0.,0.,0.],
                                [-1.,0.,1.]])

kernel_edge_detect2 = np.array([[0.,1.,0.],
                                [1.,-4.,1.],
                                [0.,1.,0.]])

kernel_edge_detect3 = np.array([[-1.,-1.,-1.],
                                [-1.,8.,-1.],
                                [-1.,-1.,-1.]])

kernel_sharpen = np.array([[0.,-1.,0.],
                           [-1.,5.,-1.],
                           [0.,-1.,0.]])

kernel_sharpen2 = np.array([[-1.,-1.,-1.],
                            [-1.,9.,-1.],
                            [-1.,-1.,-1.]])

kernel_blur = np.array([[1.,1.,1.],
                        [1.,1.,1.],
                        [1.,1.,1.]])

kernel_list =
[kernel_edge_detect1,kernel_edge_detect2,kernel_edge_detect3,kernel_sharpen,kernel
_sharpen2,kernel_blur]
title_list = ['edge-detect1','edge-detect2','edge-detect3','sharpen1','sharpen2','blur']
```

Swdevperestroika, 2014, Image processing with Python, Numpy & Scipy – image convolution,
https://softwaredevelopmentperestroika.wordpress.com/2014/02/11/image-processing-with-python-numpy-scipy-image-convolution/

Copyright FAST CAMPUS Corp. All Rights Reserved

Vision




```
image=mpimg.imread('portrait.jpg')[:,:,0]
shape = image.shape
newimage = np.ndarray(shape,dtype=np.float)
poorimage = np.ndarray(shape,dtype=np.float)
supershape = (shape[0] + 2,shape[1] + 2)
supermatrix = np.zeros(supershape,dtype=np.float)
supermatrix[1:-1,1:-1] = image

imagelist_std_convolve = []
imagelist_poor_convolve = []

fig.add_subplot(4,4,1)
plt.title('Original image')
plt.imshow(image,cmap=plt.cm.gray)
plt.axis('off')
```

```
for i in range(len(kernel_list)):
    kernel_list[i] = normalize(kernel_list[i])
    newimage = filter.convolve(image,kernel_list[i],mode='constant', cval=0)
    imagelist_std_convolve.append(newimage)
    print 'kernel'
    print kernel_list[i]

    print
    fig.add_subplot(4,4,i+2)
    plt.title(title_list[i])
    plt.imshow(newimage,cmap=plt.cm.gray)
    plt.axis('off')

for i in range(len(kernel_list)):
    flipped_kernel = kernel_list[i].copy()
    flipped_kernel = np.fliplr(flipped_kernel)
    flipped_kernel = np.flipud(flipped_kernel)
    flipped_kernel = normalize(flipped_kernel)
    poorimage=poor_mans_convolve(image,supermatrix,flipped_kernel,shape)
    imagelist_poor_convolve.append(poorimage)
    print 'flipped_kernel'
    print flipped_kernel
    print
    fig.add_subplot(4,4,i+2+6)
    plt.title('poor mans ' + title_list[i])
    plt.imshow(poorimage,cmap=plt.cm.gray)
    plt.axis('off')

plt.tight_layout()
plt.show()
```

Swdevperestroika, 2014, Image processing with Python, Numpy & Scipy – image convolution,
https://softwaredevelopmentperestroika.wordpress.com/2014/02/11/image-processing-with-python-numpy-scipy-image-convolution/

Copyright FAST CAMPUS Corp. All Rights Reserved

Vision – Dataset format

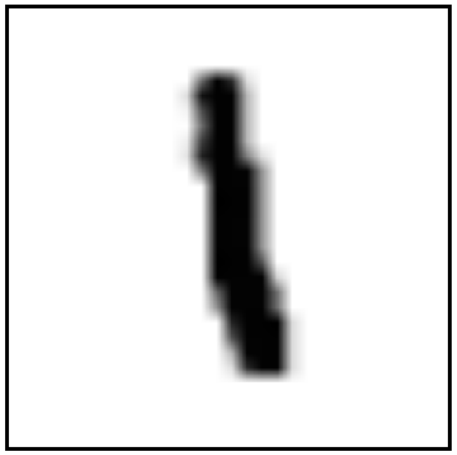
Fast campus

MNIST

Yann LeCun

train-images-idx3-ubyte.gz: training set images (9912422 bytes)
train-labels-idx1-ubyte.gz: training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz: test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz: test set labels (4542 bytes)

28x28 gray color. 60,000 trainset. 10,000 testset



≈

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

TRAINING SET LABEL FILE (train-labels-idx1-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

TRAINING SET IMAGE FILE (train-images-idx3-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel

https://daddynkidsmakers.blogspot.com/2017/06/hello-world-mnist.html

Vision – Dataset format

Fast campus

CIFAR Alex Krizhevsky, Vinod Nair, Geoffrey Hinton http://www.cs.toronto.edu/~kriz/cifar.html

CIFAR-10 60000 32x32 RGB images. 10 classes. 50000 train images. 10000 test images.

```
|-- annotation.txt
|-- 0001.jpg
|-- 0002.jpg
|...
|-- n.jpg
```

→

```
0001.jpg <label ID>
0002.jpg <label ID>
...
n.jpg <label ID>
```

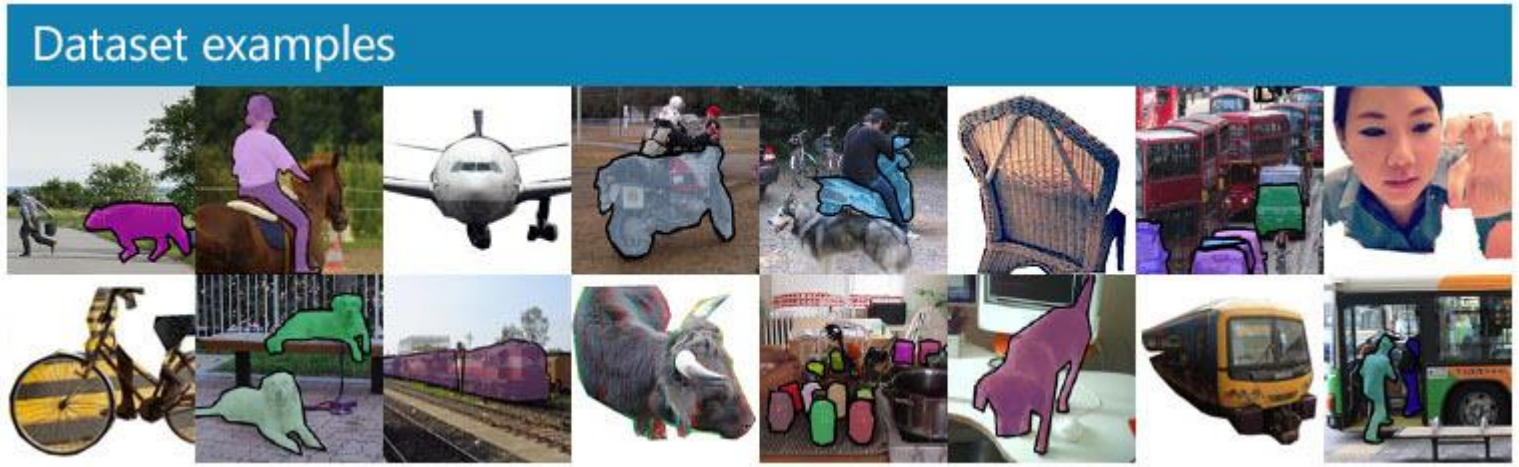
CIFAR-10 python version

Copyright FAST CAMPUS Corp. All Rights Reserved

Vision – Dataset format

COCO(Common Objects in Context)

Microsoft. 330,000 images. 200,000 labeled objects information.
80 classes



instances_val2014_fakebbox100_results.json (TOSHIBA EXT /media/ktw/TOSHIBA EXT/05.Dat

Open Save


```
[{"image_id":42,"category_id":18,"bbox":[258.15,41.29,348.26,243.78],"score":0.236},
{"image_id":73,"category_id":11,"bbox":[61,22.75,504,609.67],"score":0.318},
{"image_id":73,"category_id":4,"bbox":[12.66,3.32,268.6,271.91],"score":0.726},
{"image_id":74,"category_id":18,"bbox":[87.87,276.25,296.42,103.18],"score":0.546},
{"image_id":74,"category_id":2,"bbox":[0,3.66,142.15,312.4],"score":0.3},
{"image_id":74,"category_id":1,"bbox":[296.55,93.96,18.42,58.83],"score":0.407},
```


JSON Tab Width: 8 Ln 1, Col 347 INS

5 annotation type = {object detection, key point, stuff segmentation, panoptic segmentation, image captioning}

Copyright FAST CAMPUS Corp. All Rights Reserved

Vision – Dataset format

COCO(Common Objects in Context)

Annotation = {
segmentation: segmentation masks vertex x, y list
area: bounding box area (pixel)
iscrowd: if single object then 0 else 1. RLE(run length encoding) is used with 1
imageid: image ID
bbox: bounding box. top left x top left y, width, height
category: category definition
id: annotation ID
}

```
annotation{
"id": int,
"image_id": int,
"category_id": int,
"segmentation": RLE or [polygon],
"area": float,
"bbox": [x,y,width,height],
"iscrowd": 0 or 1,
}
```

```
"annotations": [
    {
        "segmentation":
[[510.66,423.01,511.72,420.03,...,510.45,423.01]],
        "area": 702.10,
        "iscrowd": 0,
        "image_id":  397133,
        "bbox": [433.07,355.93,138.65,228.67],
        "category_id": 18,
        "id": 1768
    },
    {
        "segmentation":
      {
        "counts":[12,56,198,10]
        "size":[120, 240]
      }
        "area": 500.2,
        "iscrowd": 1,
        "image_id": 397122,
        "bbox": [473.07,395.93,38.65,28.67],
        "category_id": 18,
        "id": 1768
    }]
```

```
|-- val
      |-- 0001.jpg
      |-- 0002.jpg
      ...
      |-- n.jpg
|-- annotations
      |-- instances_val.json
```

PUS Corp. All Rights Reserved

Vision – Dataset format

Pascal(Pattern Analysis, Statistical Modeling and Computational Learning) VOC(Visual Object Classes). VOC2012 = 20 classes, Train/val 11,530, 6,929 segmentations

http://host.robots.ox.ac.uk/pascal/VOC/

http://host.robots.ox.ac.uk/pascal/VOC/voc2012/#testdata

ImageNet

Kaggle

```
<annotation>
        <folder>Kangaroo</folder>
        <filename>00001.jpg</filename>
        <path>./Kangaroo/stock-12.jpg</path>
        <source>
                <database>Kangaroo</database>
        </source>
        <size>
                <width>450</width>
                <height>319</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>kangaroo</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>233</xmin>
                        <ymin>89</ymin>
                        <xmax>386</xmax>
                        <ymax>262</ymax>
                </bndbox>
        </object>
</annotation>
```

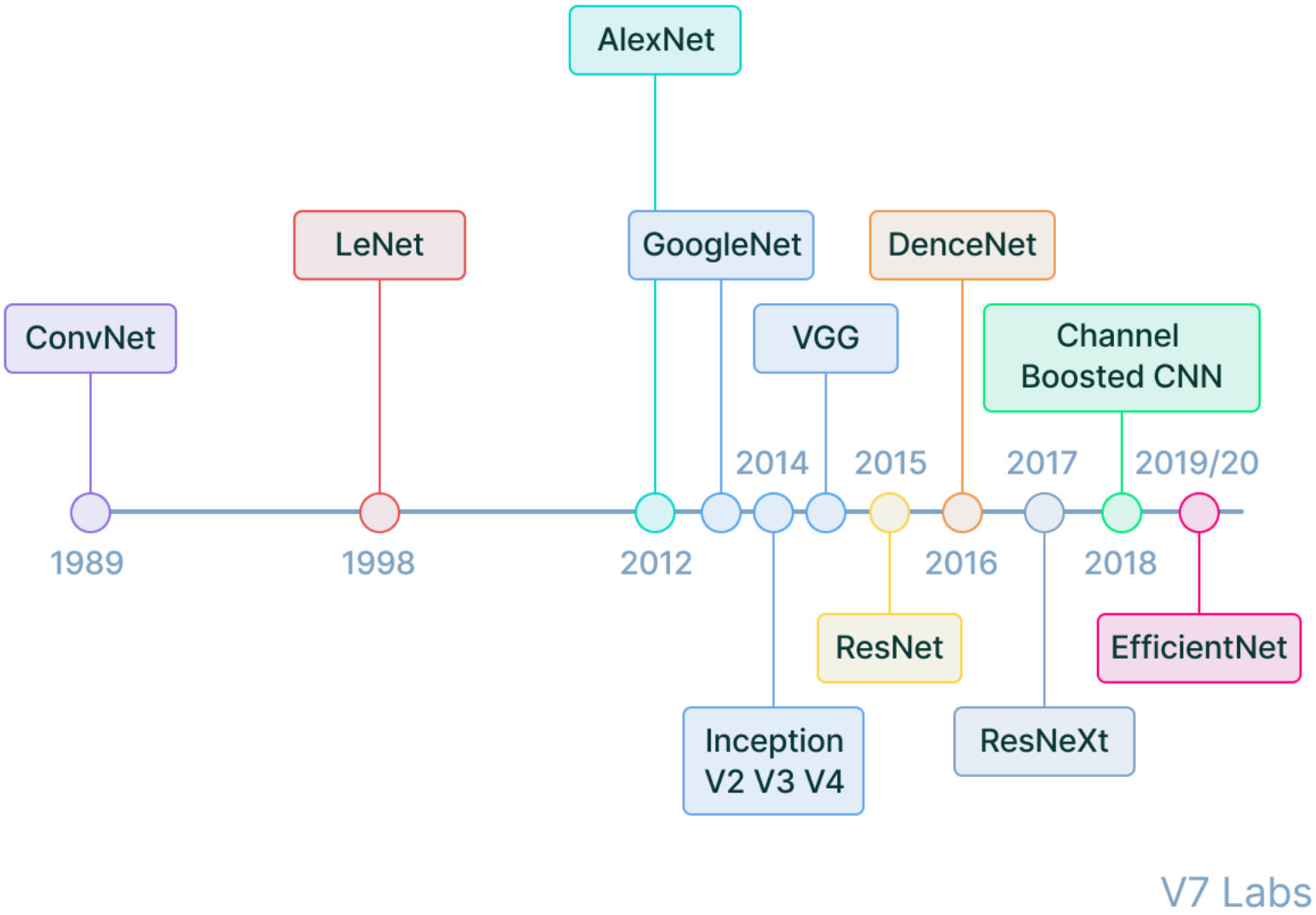
```
|-- VOCdevkit
    |-- VOC
        |-- Annotations
            |-- 0001.xml
            |-- 0002.xml
            ...
            |-- n.xml
        |-- ImageSets
            |-- Layout
                |-- test.txt
            |-- Main
                |-- 0001_test.txt
                |-- 0002_test.txt
                ...
                |-- n_test.txt
            |-- Segmentation
                |-- test.txt
        |-- Images
            |-- 0001.jpg
            |-- 0002.jpg
            ...
            |-- n.jpg
        |-- SegmentationClass
            |-- 0001.png
            |-- 0002.png
            ...
            |-- n.png
        |-- SegmentationObject
            |-- 0001.png
            |-- 0002.png
            ...
            |-- n.png
```

Copyright FAST CAMPUS Corp. All Rights Reserved

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

Object Recognition

Object Recognition



Pragati Baheti, 2022, A Comprehensive Guide to Convolutional Neural Networks, v7labs, https://www.v7labs.com/blog/convolutional-neural-networks-guide

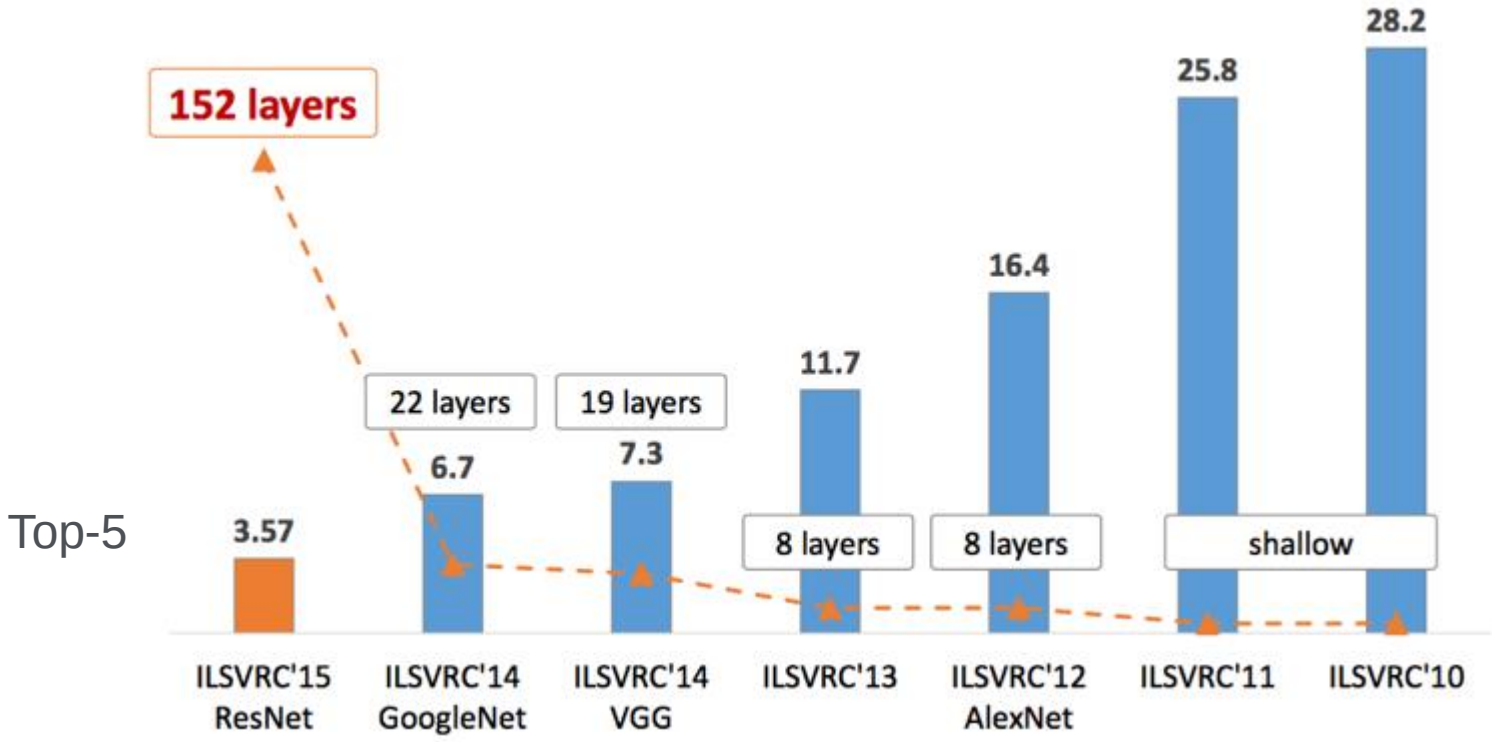
Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition

Fast campus

classification

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)



Siddharth Das, 2017, CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more, medium, medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition

Year	CNN	Developed by	Place	Top-5 error rate	No. of parameters
1998	LeNet(8)	Yann LeCun et al			60 thousand
2012	AlexNet(7)	Alex Krizhevsky, Geoffrey Hinton, Ilya Sutskever	1st	15.3%	60 million
2013	ZFNet()	Matthew Zeiler and Rob Fergus	1st	14.8%	
2014	GoogLeNet(19)	Google	1st	6.67%	4 million
2014	VGG Net(16)	Simonyan, Zisserman	2nd	7.3%	138 million
2015	ResNet(152)	Kaiming He	1st	3.6%	

MNIST
CNN

Max pooling (down sampling)
Drop out
ReLU
L2 Regularization Softmax, lr decay
GPU

→ Deep Learning.
Over 50, 100, 200 layers

Inception module
Conv1x1 Reduce layer (decrease operation cost)
Bottleneck layer

Conv3x3,Pooling2x2 Simplification
Training speed improvement

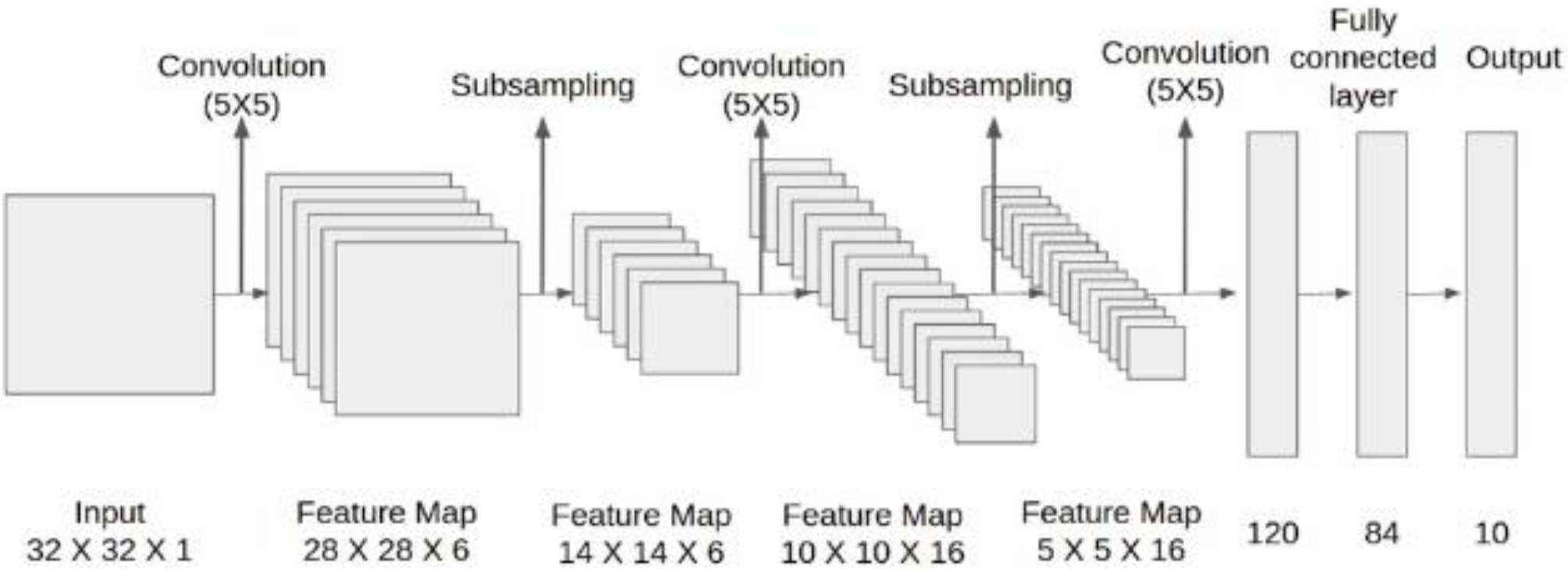
Skip connection (shortcut)
Vanishing gradient solution

Siddharth Das, 2017, CNN Architectures: LeNet, AlexNet, VGG, GoogLeNet, ResNet and more, medium, medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition - LeNet

MNIST
CNN



Shipra Saxena, 2021, The Architecture of Lenet-5, analyticsvidhya (www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/)

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition - Convolution

Fast campus

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

$$308 + -498 + 164 + 1 = -25$$

Bias = 1

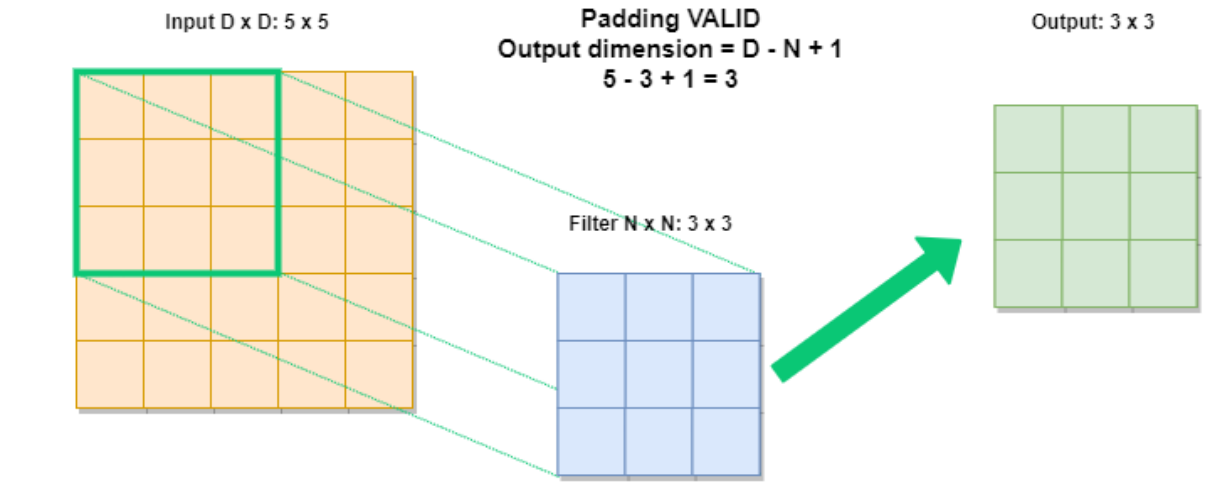
Output

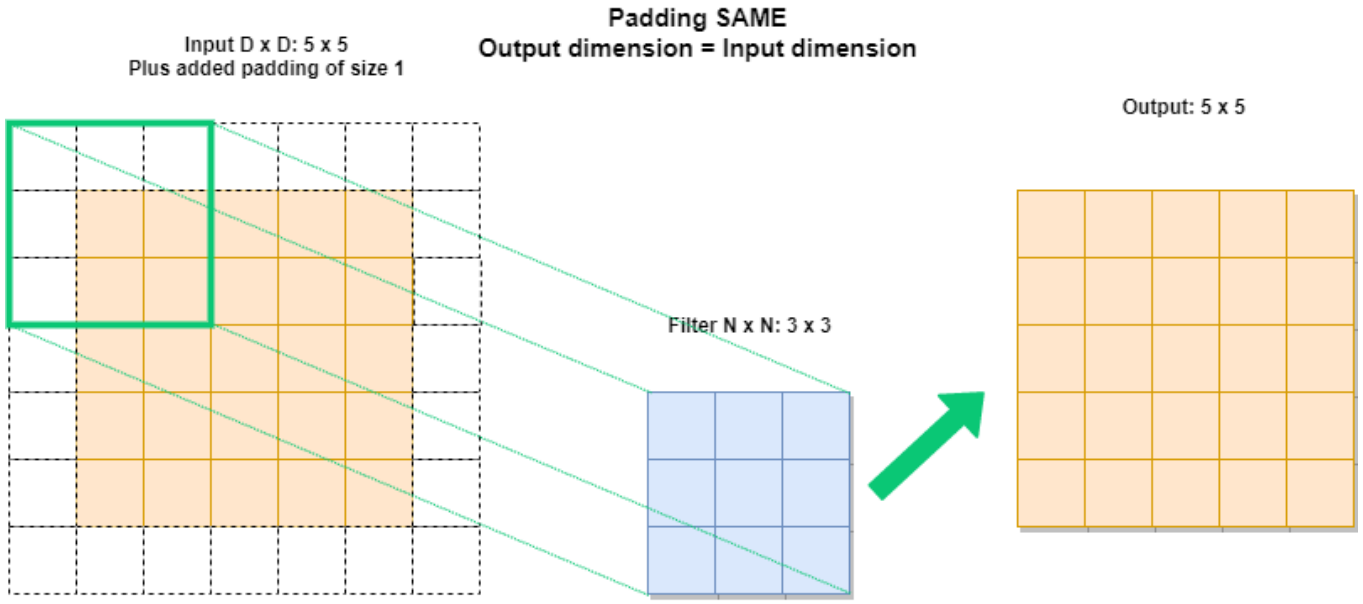
-25				...
				...
				...
				...
...	...	...	...	...

Mustafa Murat ARAT, 2019, Implementing 'SAME' and 'VALID' padding of Tensorflow in Python, https://mmuratarat.github.io/2019-01-17/implementing-padding-schemes-of-tensorflow-in-python

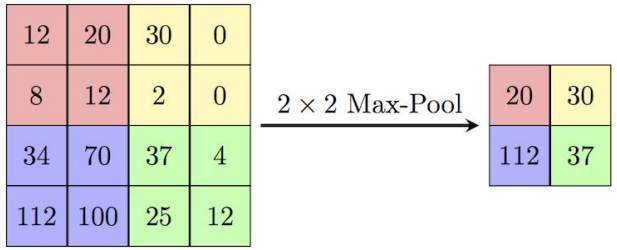
Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition – Convolution & Pooling





AIGeekProgrammer.com © 2019



Network Excel Calculator

Convolution Output dimension = [(I - K +2 * P) / S] +1, D

$$O = \frac{I - K + 2P}{S} + 1$$

Pooling Output dimension = [(I - K) / S] +1

$$O = \frac{I - K}{S} + 1$$

parmeters = filter size x kernel size x previous filter size + filter size

No	Image x (or input)	Image y	Image ch	Net	Filter size (channel) or output size	Kernel x size	Kernel y size
1	28	**28**	**1**	Conv	32	3	
2	28	28	32	Max pooling	32		
3	14	14	32	Conv	64	3	
4	14	14	64	Max pooling	64		
5	7	7	64	Flatten			
6	3136			Dense	64		
7	64			Dense	10		
Sum							

AI Geek Programmer, 2019, Convolutional neural network 2: architecture, https://aigeekprogrammer.com/convolutional-neural-network-image-recognition-part-2/

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition - LeNet

Dive

Fast campus

Layer	# filters / neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv 1	6	5 * 5	1	28 X 28 X 6	tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv 2	16	5 * 5	1	10 X 10 X 16	tanh
Avg. pooling 2		2 * 2	2	5 X 5 X 16	
Conv 3	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax

```
model = Sequential()

model.add(Conv2D(6, kernel_size=(5, 5), strides=(1, 1), activation='tanh', input_shape=input_shape, padding='same'))

model.add(AveragePooling2D(pool_size=(2, 2), strides=2, padding='valid'))

model.add(Conv2D(16, kernel_size=(5, 5), strides=(1, 1), activation='tanh', padding='valid'))

model.add(AveragePooling2D(pool_size=(2, 2), strides=2, padding='valid'))

model.add(Conv2D(120, kernel_size=(5, 5), strides=(1, 1), activation='tanh', padding='valid'))

model.add(Flatten())
model.add(Dense(84, activation='tanh'))
model.add(Dense(10, activation='softmax'))

model.summary()
```

Shipra Saxena, 2021, The Architecture of Lenet-5, analyticsvidhya (www.analyticsvidhya.com/blog/2021/03/the-architecture-of-lenet-5/)

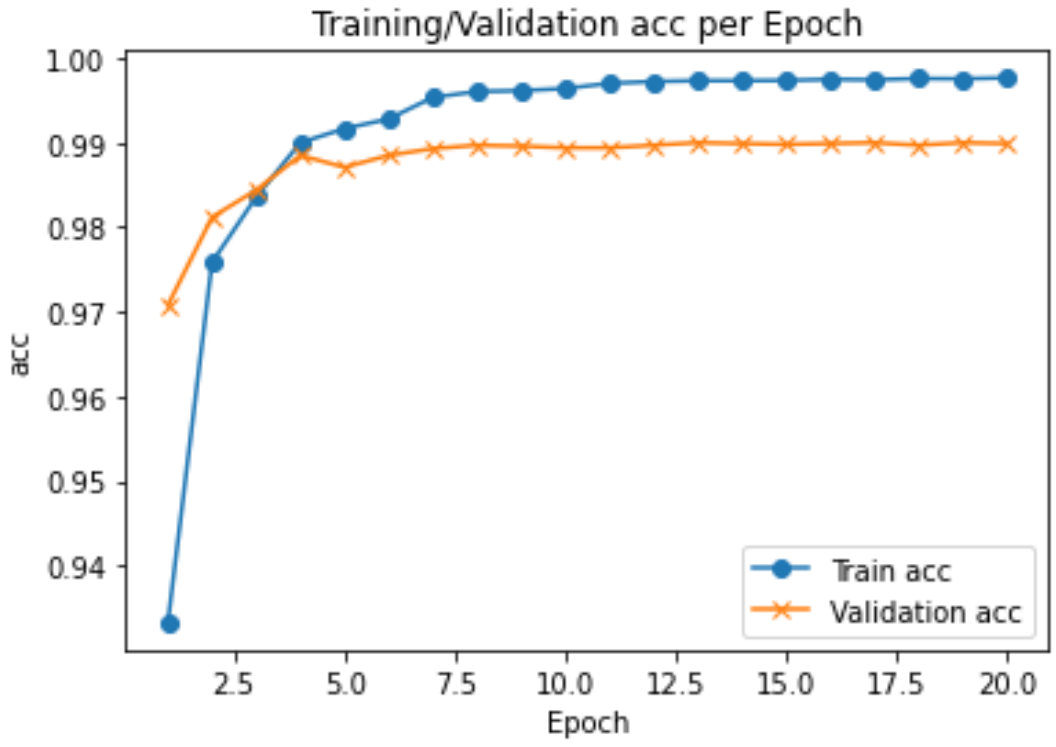
Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition - LeNet

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 6)         156

 average_pooling2d (AverageP  (None, 14, 14, 6)        0
 ooling2D)

 conv2d_1 (Conv2D)           (None, 10, 10, 16)        2416

 average_pooling2d_1 (Averag  (None, 5, 5, 16)         0
 ePooling2D)

 conv2d_2 (Conv2D)           (None, 1, 1, 120)         48120

 flatten (Flatten)           (None, 120)               0

 dense (Dense)               (None, 84)                10164

 dense_1 (Dense)             (None, 10)                850

=================================================================
Total params: 61,706
Trainable params: 61,706
Non-trainable params: 0
_________________________________________________________________
```

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition - LeNet




```
Epoch 19: val_loss did not improve from 0.03015
1875/1875 - 37s - loss: 0.0114 - accuracy: 0.9976 - val_loss: 0.0303 - val_accuracy: 0.9900 - lr: 1.0000e-05 - 37s/epoch - 20ms/step
Epoch 20/20

Epoch 20: val_loss did not improve from 0.03015
1875/1875 - 39s - loss: 0.0113 - accuracy: 0.9977 - val_loss: 0.0302 - val_accuracy: 0.9899 - lr: 1.0000e-05 - 39s/epoch - 21ms/step
```

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition – CNN & Pooling

Dive

Fast campus

Keras

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv2D, MaxPool2D, Flatten
from keras.layers import BatchNormalization

model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(MaxPool2D())
model.add(Conv2D(16, (3, 3), activation='relu'))
model.add(MaxPool2D())
model.add(Flatten())
model.add(Dense(10, activation='softmax'))

model.summary()
```

PyTorch

```
import torch

nn = torch.nn
F = torch.nn.functional

class Net(nn.Module):
  def __init__(self):
    super(Net, self).__init__()
    self.conv1 = nn.Conv2d(3, 32, 3)
    self.conv2 = nn.Conv2d(32, 16, 3)
    self.fc1 = nn.Linear(16 * 6 * 6, 10)
    self.pool = nn.MaxPool2d(2, 2)

def forward(self, x):
    x = self.pool(F.relu(self.conv1(x)))
    x = self.pool(F.relu(self.conv2(x)))
    x = x.view(-1, 16 * 6 * 6)
    x = F.log_softmax(self.fc1(x), dim=-1)
    return x

model = Net()
print(model)
```

Piotr Migdal and Rafał Jakubanis, 2018.6, Keras or PyTorch as your first deep learning framework, deepsense.ai, https://deepsense.ai/keras-or-pytorch/

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition - AlexNet

Max pooling (down sampling)
Drop out
ReLU
L2 Regularization Softmax, lr decay
GPU

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

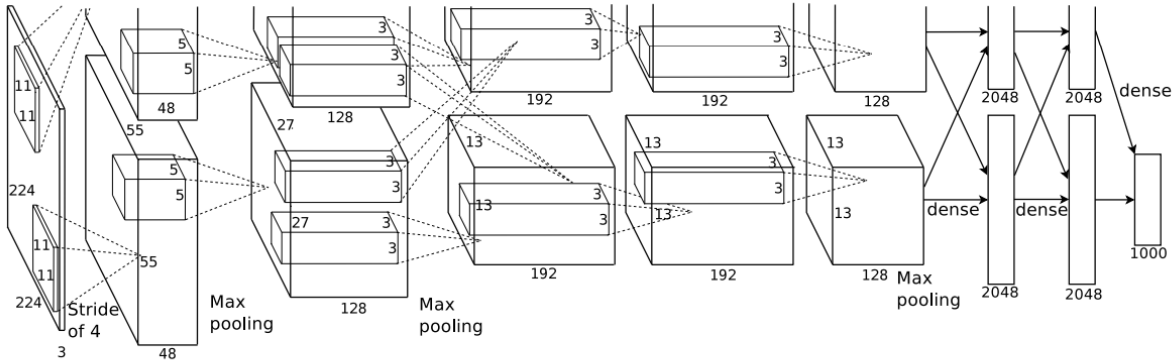
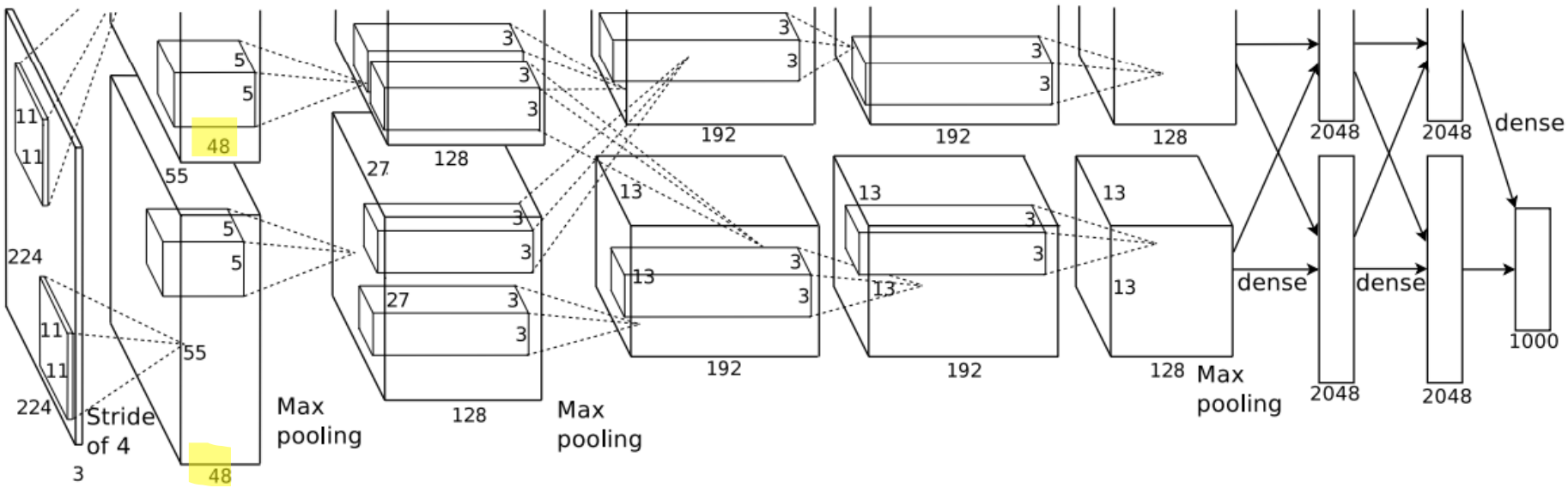
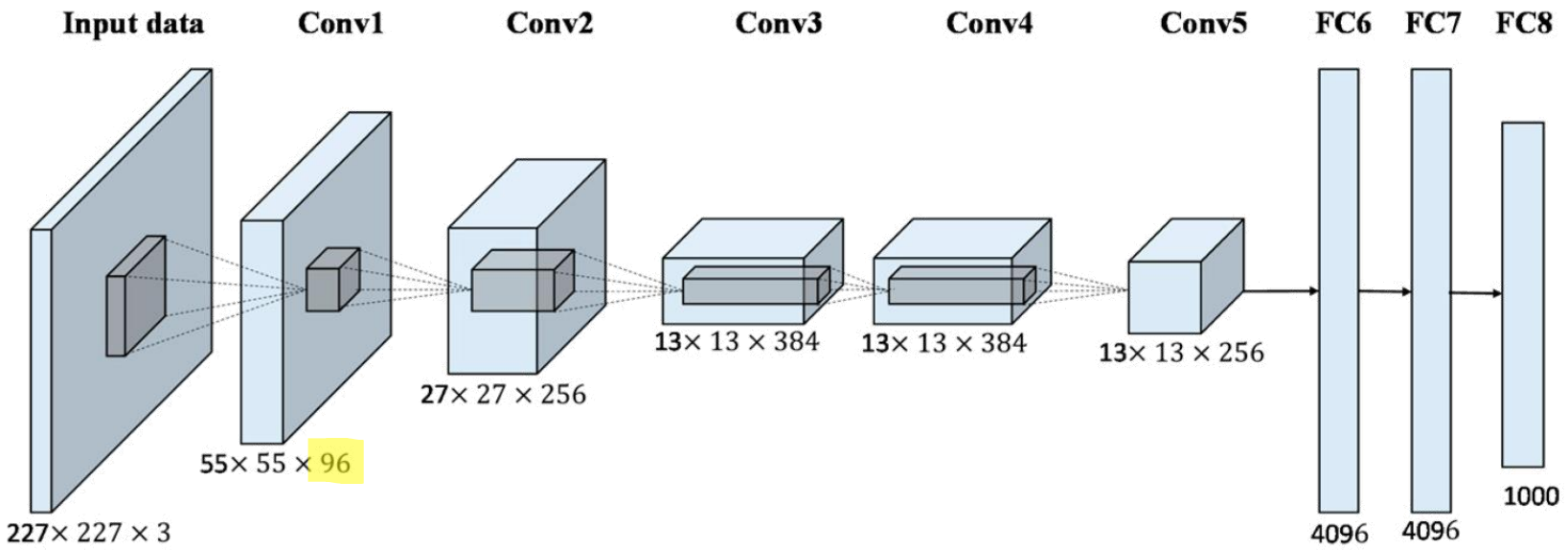


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition - AlexNet





Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition - AlexNet

Dive

Fast campus

```
import tensorflow
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv2D, MaxPool2D, Flatten, AveragePooling2D, Activation
from keras.layers import BatchNormalization
from keras.regularizers import l2

model = Sequential(name="Alexnet")
# Conv1 (conv + pool + batchnorm)
model.add(Conv2D(filters= 96, kernel_size=(11,11), strides=(4,4), padding='valid', kernel_regularizer=l2(0.0005), input_shape =
(227,227,3)))
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size=(3,3), strides= (2,2), padding='valid'))   # pool 5x5
model.add(BatchNormalization())

# Conv2 (conv + pool + batchnorm)
model.add(Conv2D(filters=256, kernel_size=(5,5), strides=(1,1), padding='same', kernel_regularizer=l2(0.0005)))
model.add(Activation('relu'))
model.add(MaxPool2D(pool_size=(3,3), strides=(2,2), padding='valid'))
model.add(BatchNormalization())

# Conv3 (conv + batchnorm)
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='same', kernel_regularizer=l2(0.0005)))
model.add(Activation('relu'))
model.add(BatchNormalization())

# Conv4 (conv + batchnorm)
model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1), padding='same', kernel_regularizer=l2(0.0005)))
model.add(Activation('relu'))
model.add(BatchNormalization())

# Conv5 (conv + batchnorm)
model.add(Conv2D(filters=256, kernel_size=(3,3),
strides=(1,1), padding='same',
kernel_regularizer=l2(0.0005)))
model.add(Activation('relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(3,3), strides=(2,2),
padding='valid'))
model.add(Flatten())

# FC6 (Dense layer + dropout)
model.add(Dense(units = 4096, activation = 'relu'))
model.add(Dropout(0.5))

# FC7 (Dense layers)
model.add(Dense(units = 4096, activation = 'relu'))
model.add(Dropout(0.5))

# FC8 (softmax output layer)
model.add(Dense(units = 1000, activation = 'softmax'))

model.summary()
```

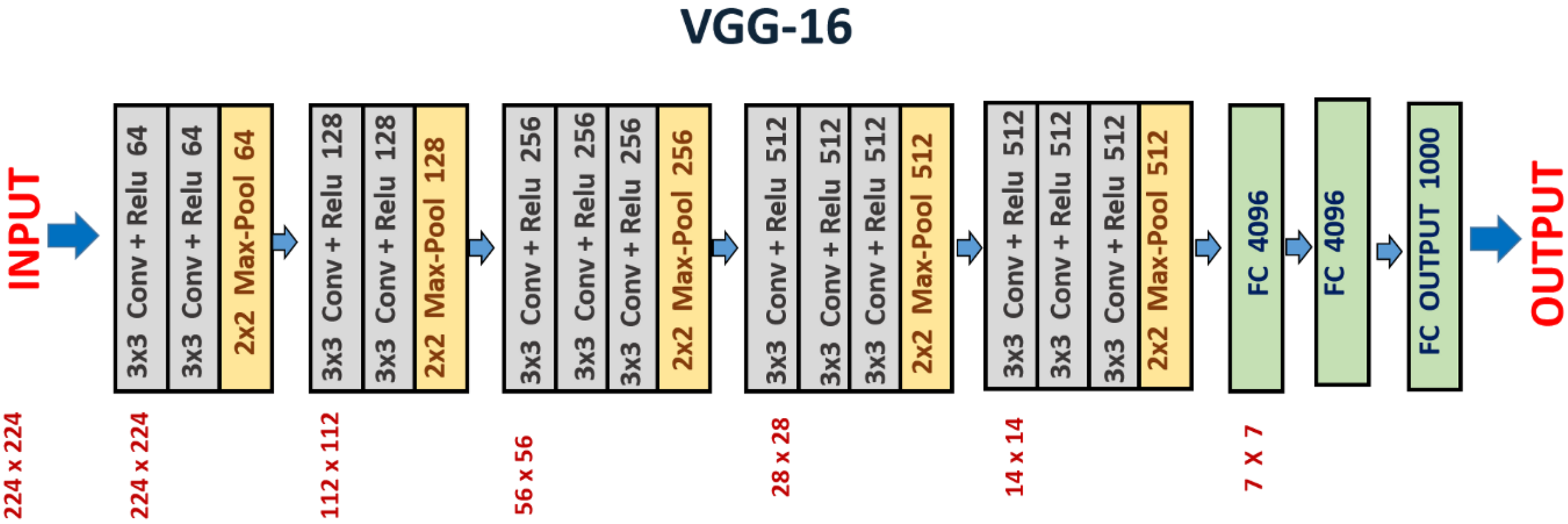
Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition – VGG16

Conv3x3,Pooling2x2 Simplification
Training speed improvement

Fast campus

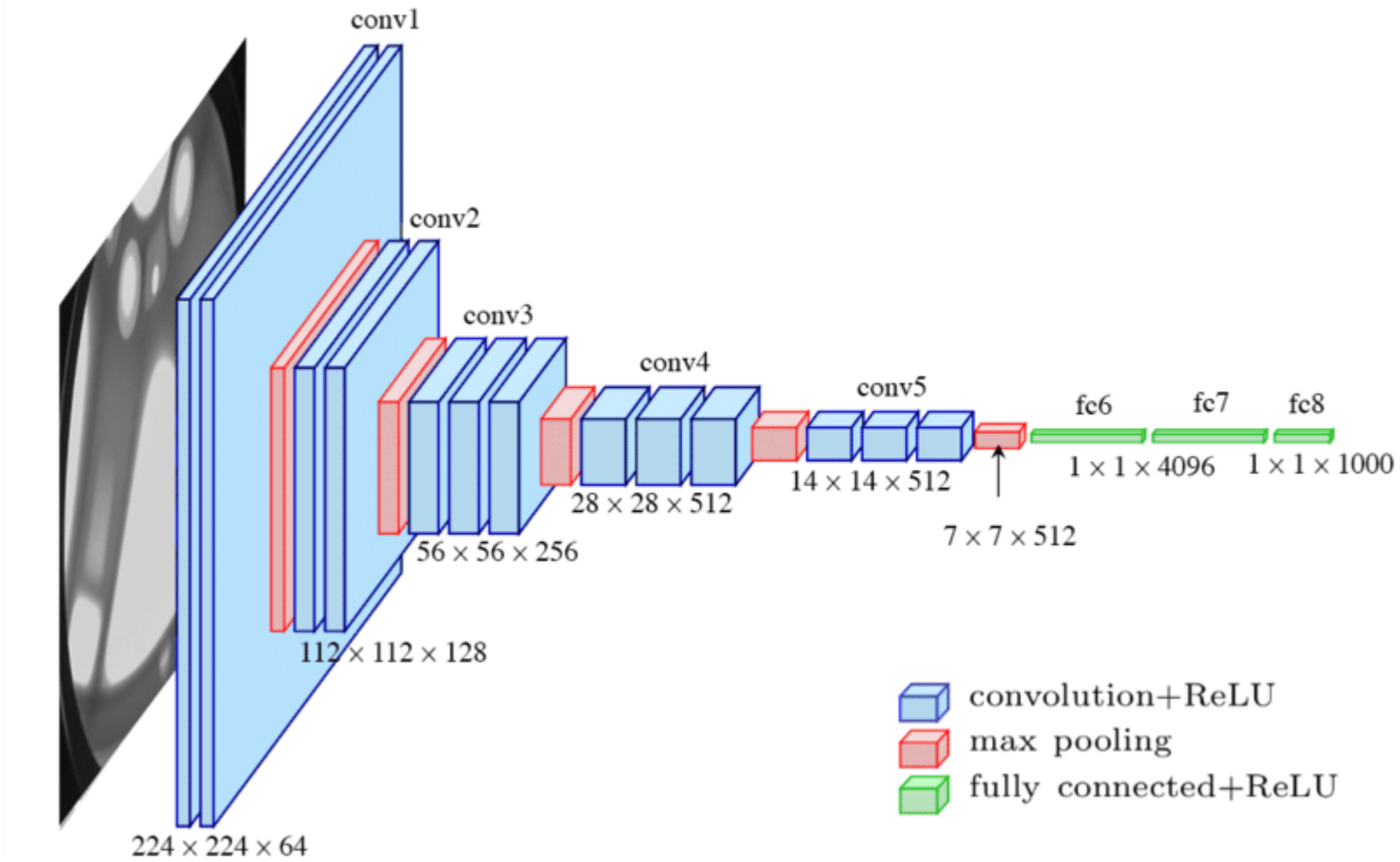
by Visual Geometry Group (Oxford)



ESIN MESELI, 2020, Fruit Detection with VGG-16,
www.kaggle.com/code/thepinokyo/fruit-detection-with-vgg-16

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition – VGG16



ESIN MESELI, 2020, Fruit Detection with VGG-16,
www.kaggle.com/code/thepinokyo/fruit-detection-with-vgg-16

Copyright **FAST CAMPUS** Corp. All Rights Reserved

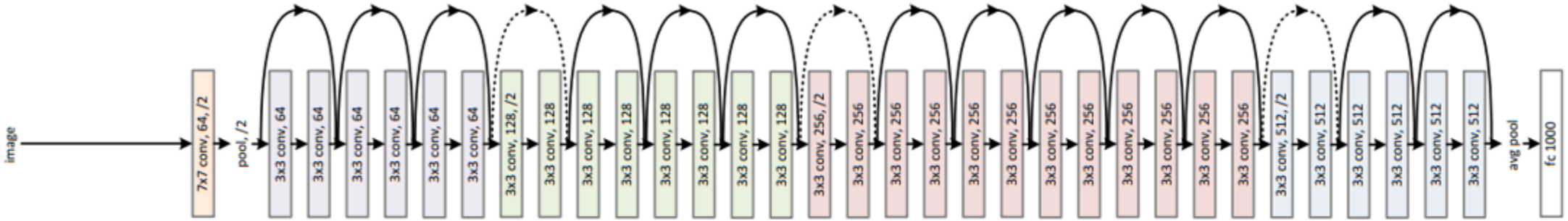
Object Recognition – ResNet

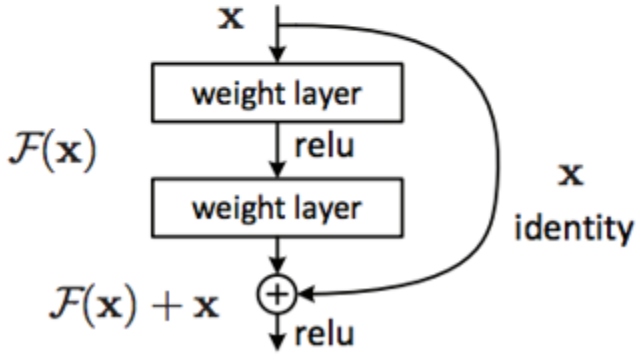
Skip connection (shortcut)
Vanishing gradient solution

Fast campus

Vanishing Gradient problem

Residual Networks (ResNet50)





relu(f(x) + x)

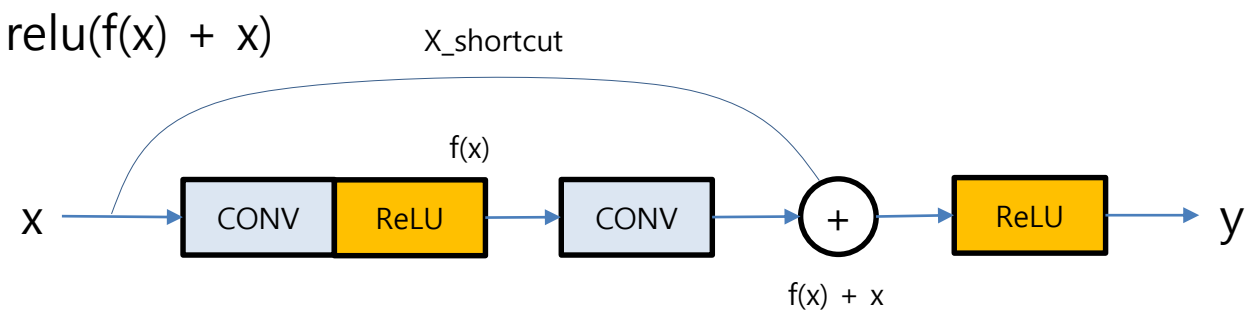
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, 2015, Deep Residual Learning for Image Recognition, jananisbabu.github.io/ResNet50_From_Scratch_Tensorflow

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition – ResNet

Fast campus

Vanishing Gradient problem



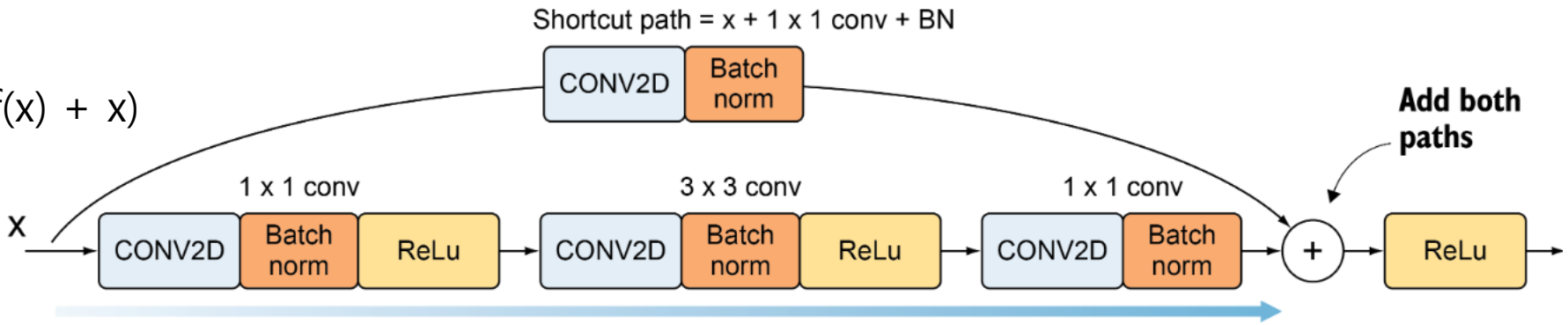

```
X_shortcut = X

X = Conv2D(filters = 64, kernel_size = (3, 3), strides = (1,1))(X)
X = Activation('relu')(X)
X = Conv2D(filters = 64, kernel_size = (3, 3), strides = (1,1))(X)

X = Add()([X, X_shortcut])

X = Activation('relu')(X)
```


Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition – ResNet

Bottlenet residual block relu(f(x) + x)

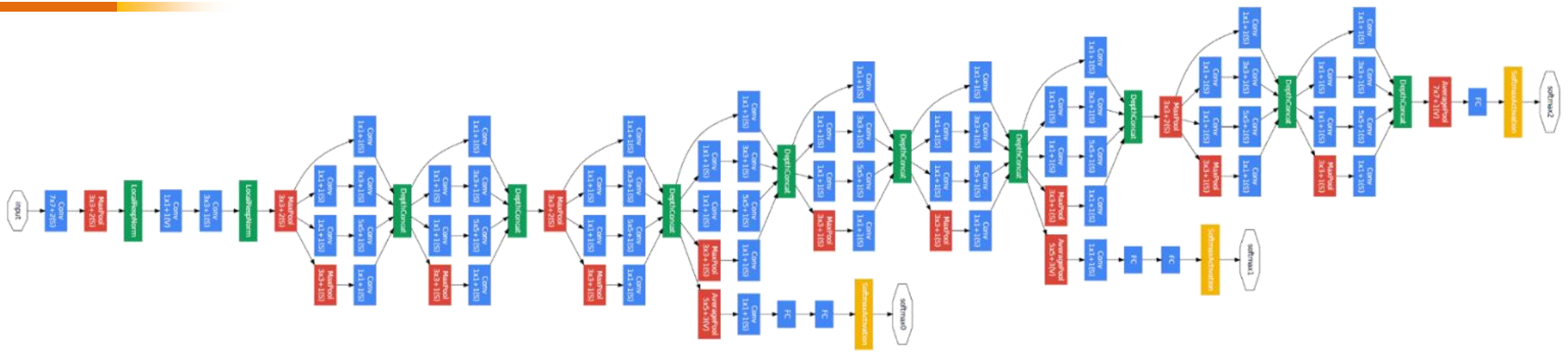
Sample N x image H x image W x C = **X**

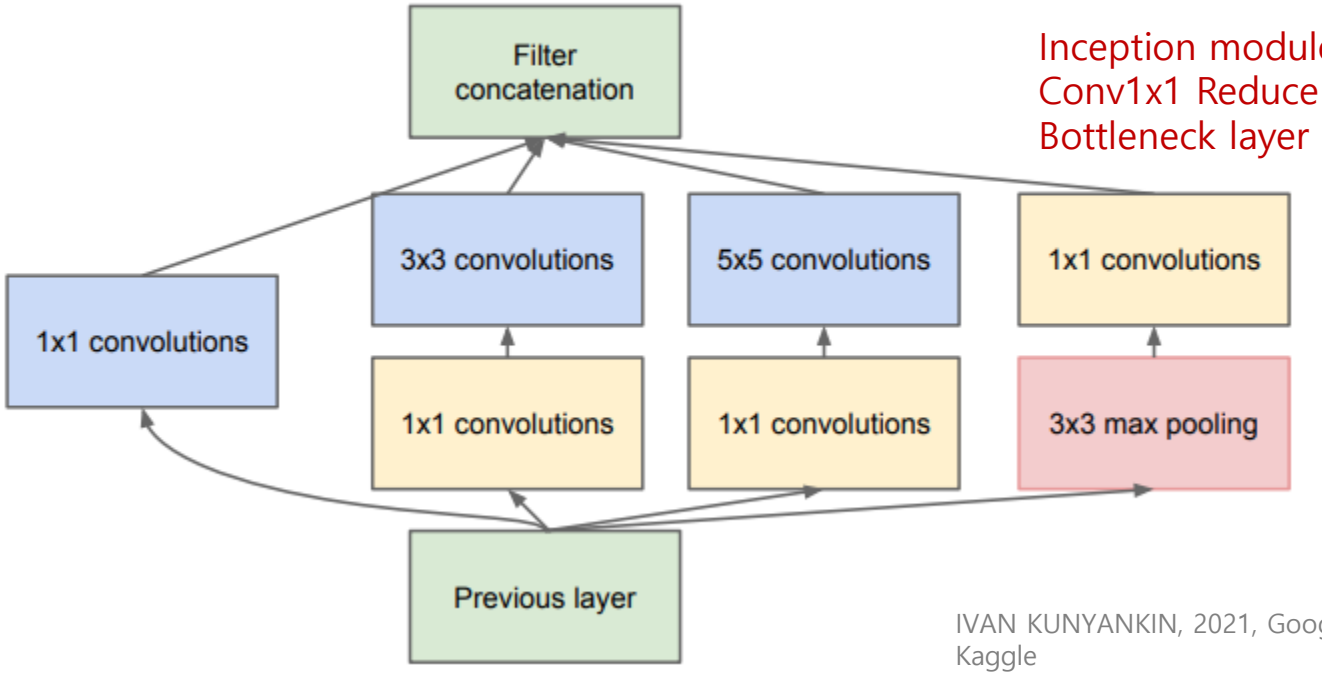



```
def bottleneck_residual_block(X, f, filters, stage, block, reduce=False, s=2):
    F1, F2, F3 = filters

    X_shortcut = X

    if reduce:
        X = Conv2D(filters = F1, kernel_size = (1, 1), strides = (s,s), padding = 'valid', name = conv_name_base + '2a')(X)
        X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)
        X = Activation('relu')(X)

        X_shortcut = Conv2D(filters = F3, kernel_size = (1, 1), strides = (s,s), padding = 'valid', name = conv_name_base + '1',
                    kernel_initializer = glorot_uniform(seed=0))(X_shortcut)
        X_shortcut = BatchNormalization(axis = 3, name = bn_name_base + '1')(X_shortcut)
    else:
        X = Conv2D(filters = F1, kernel_size = (1, 1), strides = (1,1), padding = 'valid', name = conv_name_base + '2a')(X)
        X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)
        X = Activation('relu')(X)

    X = Conv2D(filters = F2, kernel_size = (f, f), strides = (1,1), padding = 'same', name = conv_name_base + '2b')(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2b')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters = F3, kernel_size = (1, 1), strides = (1,1), padding = 'valid', name = conv_name_base + '2c')(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2c')(X)

    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)

    return X
```


Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition – GoogleLeNet (inception)





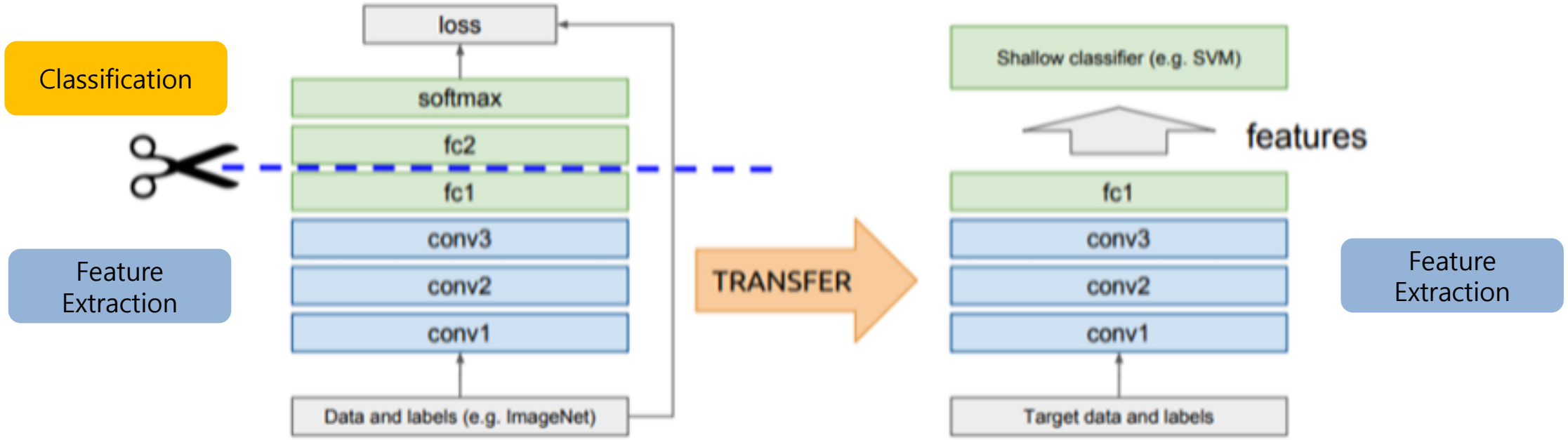
Inception module
Conv1x1 Reduce layer (decrease operation cost)
Bottleneck layer

IVAN KUNYANKIN, 2021, GoogLeNet (Inception) from scratch using Pytorch, Kaggle

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition – Transfer learning

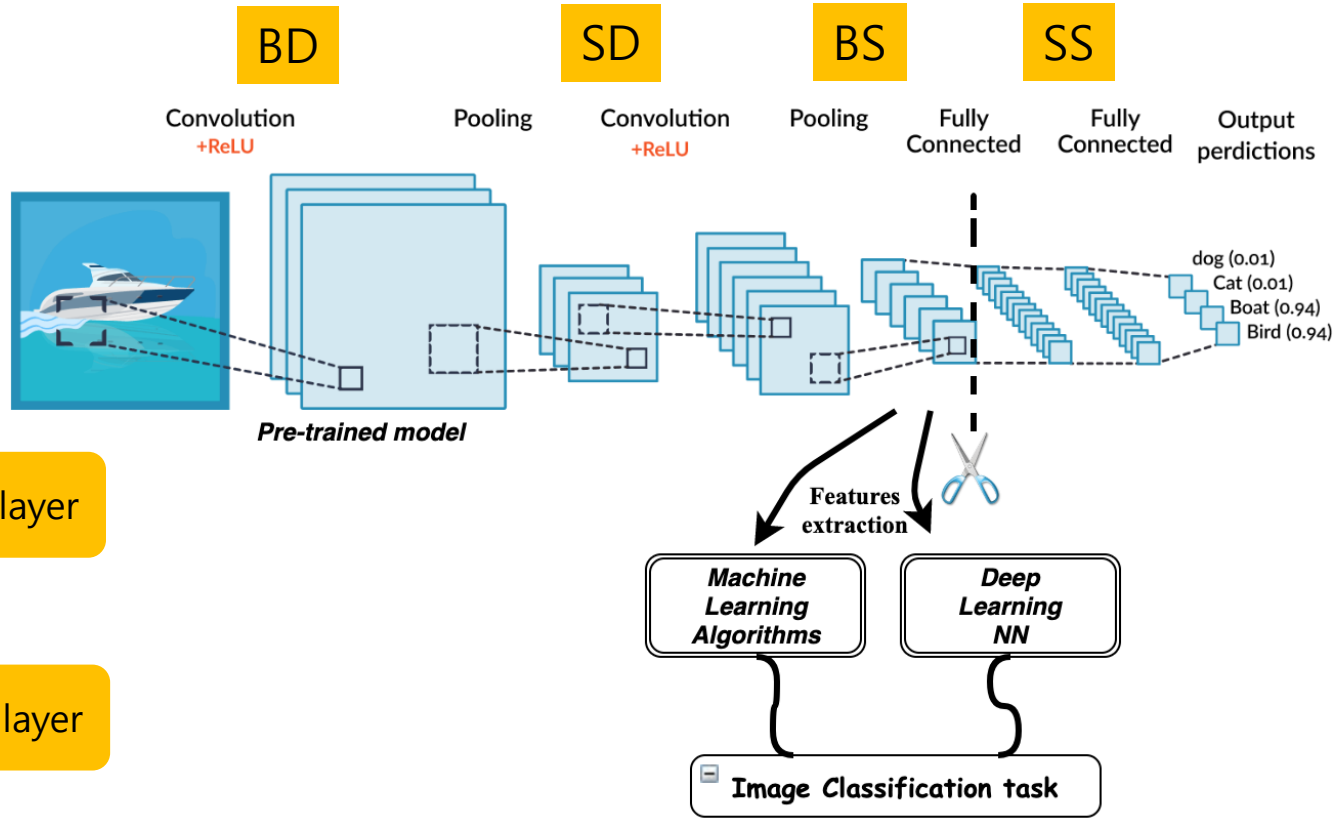
Solution? Lack of data
Excessive computational demands



Dipanjan Sarkar, 2018, A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning, towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition – Transfer learning



SS. Small target dataset size, Similar domains

Feature Extraction (fixed 90 ~ 100%) → Your Classification layer

BS. Big target dataset size, Similar domains

Feature Extraction (fixed 60 ~ 80%) → Your Classification layer

SD. Small target dataset size, Different domains

Feature Extraction (fixed 30 ~ 50%) → Your Classification layer

BD. Big target dataset size, Different domains

Feature Extraction → Your Classification layer

Lorenzo Famiglini, 2019.11, Transfer Learning with Deep Learning & Machine Learning techniques

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition – Transfer learning

Dive

Fast campus

```
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.applications import imagenet_utils
from keras.applications import vgg16
from keras.applications import mobilenet
from tensorflow.keras.optimizers import Adam, SGD
from keras.metrics import categorical_crossentropy

from sklearn.metrics import confusion_matrix
import itertools
import matplotlib.pyplot as plt

train_path  = 'data/train'
valid_path  = 'data/valid'
test_path  = 'data/test'

train_batches = ImageDataGenerator(preprocessing_function=vgg16.preprocess_input).flow_from_directory(
    train_path, target_size=(224,224), batch_size=30)
valid_batches = ImageDataGenerator(preprocessing_function=vgg16.preprocess_input).flow_from_directory(
    valid_path, target_size=(224,224), batch_size=30)
test_batches = ImageDataGenerator(preprocessing_function=vgg16.preprocess_input).flow_from_directory(
    test_path, target_size=(224,224), batch_size=30)

base_model = vgg16.VGG16(weights = "imagenet", include_top=False, input_shape = (224,224, 3))

for layer in base_model.layers:
    layer.trainable = False

base_model.summary()
```


Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition – Transfer learning

```
from keras.layers import Dense, Flatten, Dropout, BatchNormalization
from keras.models import Model

# use "get_layer" method to save the last layer of the network
# save the output of the last layer to be the input of the next layer
last_layer = base_model.get_layer('block5_pool')
last_output = last_layer.output

# flatten the classifier input which is output of the last layer of VGG16 model
x = Flatten()(last_output)

# add FC layer, which has 64 units and relu activation
x = Dense(64, activation='relu', name='FC_2')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
# add our new softmax layer with 2 hidden units
x = Dense(2, activation='softmax', name='softmax')(x)

# instantiate a new_model using keras's Model class
new_model = Model(inputs=base_model.input, outputs=x)

# print the new_model summary
new_model.summary()

new_model.compile(Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])

new_model.fit_generator(train_batches, steps_per_epoch=4,
                        validation_data=valid_batches, validation_steps=2, epochs=20, verbose=2)
```


Copyright FAST CAMPUS Corp. All Rights Reserved

Object Recognition – Transfer learning

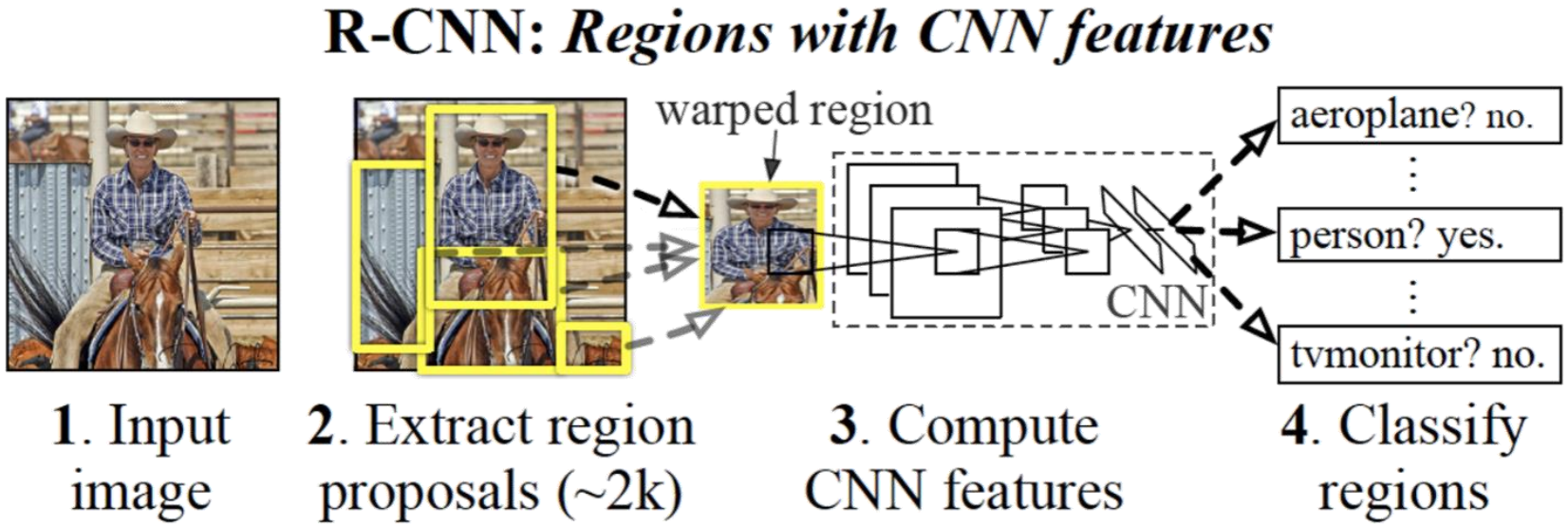
```
from sklearn.datasets import load_files
from keras.utils import np_utils
import numpy as np

def load_dataset(path):
    data = load_files(path)
    paths = np.array(data['filenames'])
    targets = np_utils.to_categorical(np.array(data['target']))
    return paths, targets

test_files, test_targets = load_dataset('data/test')

import tensorflow as tf
from keras.preprocessing import image
from keras.applications.vgg16 import preprocess_input
from tqdm import tqdm

def path_to_tensor(img_path):
    img = tf.keras.utils.load_img(img_path, target_size=(224, 224))
    x = tf.keras.utils.img_to_array(img)      # convert PIL.Image.Image type to 3D tensor with shape (224, 224, 3)
    return np.expand_dims(x, axis=0)      # convert 3D tensor to 4D tensor with shape (1, 224, 224, 3) and return 4D tensor

def paths_to_tensor(img_paths):
    list_of_tensors = [path_to_tensor(img_path) for img_path in tqdm(img_paths)]
    return np.vstack(list_of_tensors)

test_tensors = preprocess_input(paths_to_tensor(test_files))

print('₩nTesting loss: {:.4f}₩nTesting accuracy: {:.4f}'.format(*new_model.evaluate(test_tensors, test_targets)))

score = new_model.evaluate(test_tensors, test_targets)
print('₩n', 'Test accuracy:', score[1])
```


Copyright FAST CAMPUS Corp. All Rights Reserved

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

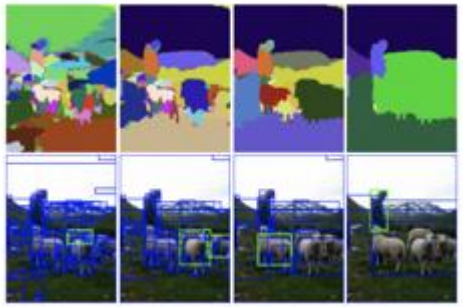
Object Detection

Object Detection – overview

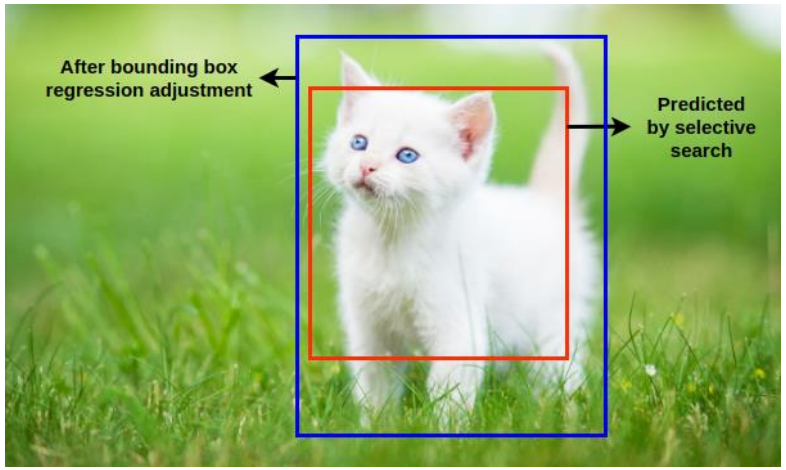


Link

1. Region Proposal



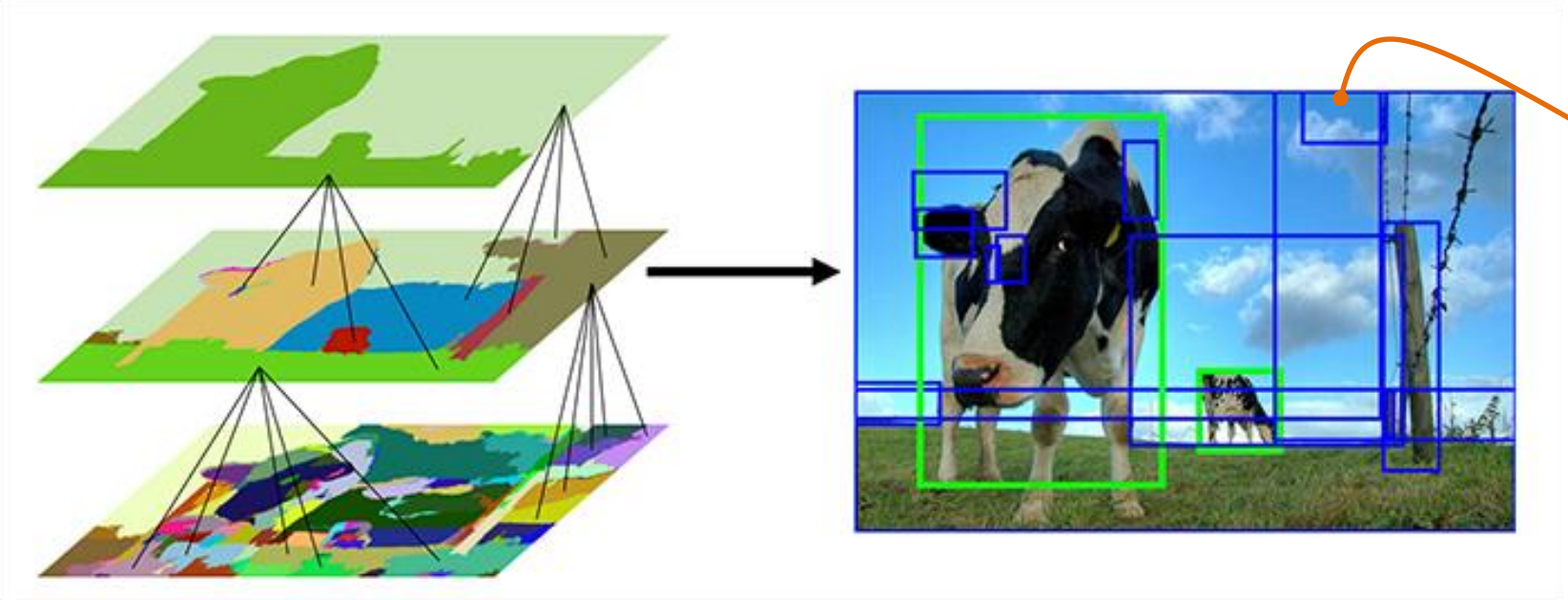
2. Region Classification



Ross Girshick, 2013, Rich feature hierarchies for accurate object detection and semantic segmentation, https://arxiv.org/pdf/1311.2524.pdf

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – algorithm



Object detection

1. RoI = Selective search
2. Object, No object = Objectness score
3. Bounding box prediction (Bbox regression)
4. Class prediction of each box
5. Non-maximum suppression (NMS)

NMS

1. If Class score < C. Return
2. Sort (prediction of predicted boxes)
3. boxes, IoU list = IoU (predicted boxes, ground truth boxes)
4. Remove (boxes, IoU list, NMS tolerance)

Vaibhaw Singh Chandel, 2017, Selective Search for Object Detection (C++ / Python), LearnOpenCV

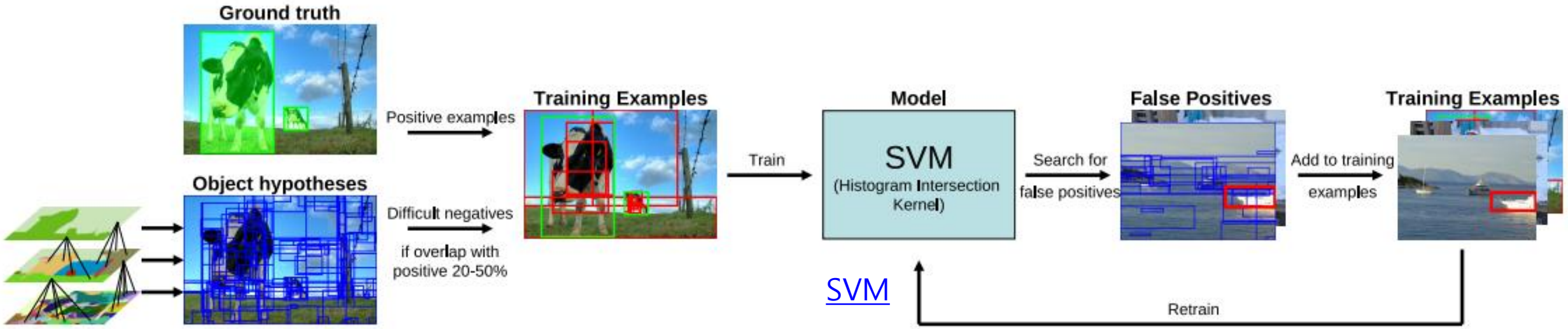
Adrian Rosebrock, 2020, OpenCV Selective Search for Object Detection, pyimagesearch.com/2020/06/29/opencv-selective-search-for-object-detection

Van de Sande, K. E., Uijlings, J. R., Gevers, T., & Smeulders, A. W. (2011, November). Segmentation as selective search for object recognition. In 2011 international conference on computer vision (pp. 1879-1886). IEEE.

Copyright FAST CAMPUS Corp. All Rights Reserved

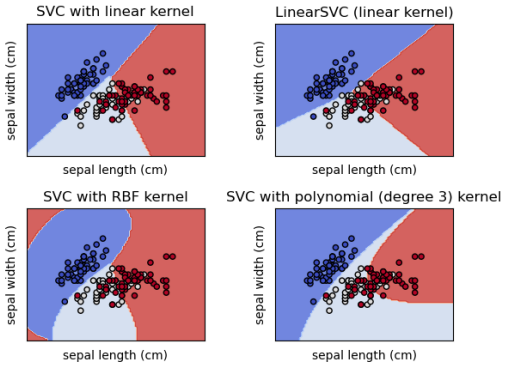
Object Detection – Regions of Interest(RoI)

Selective Search Link



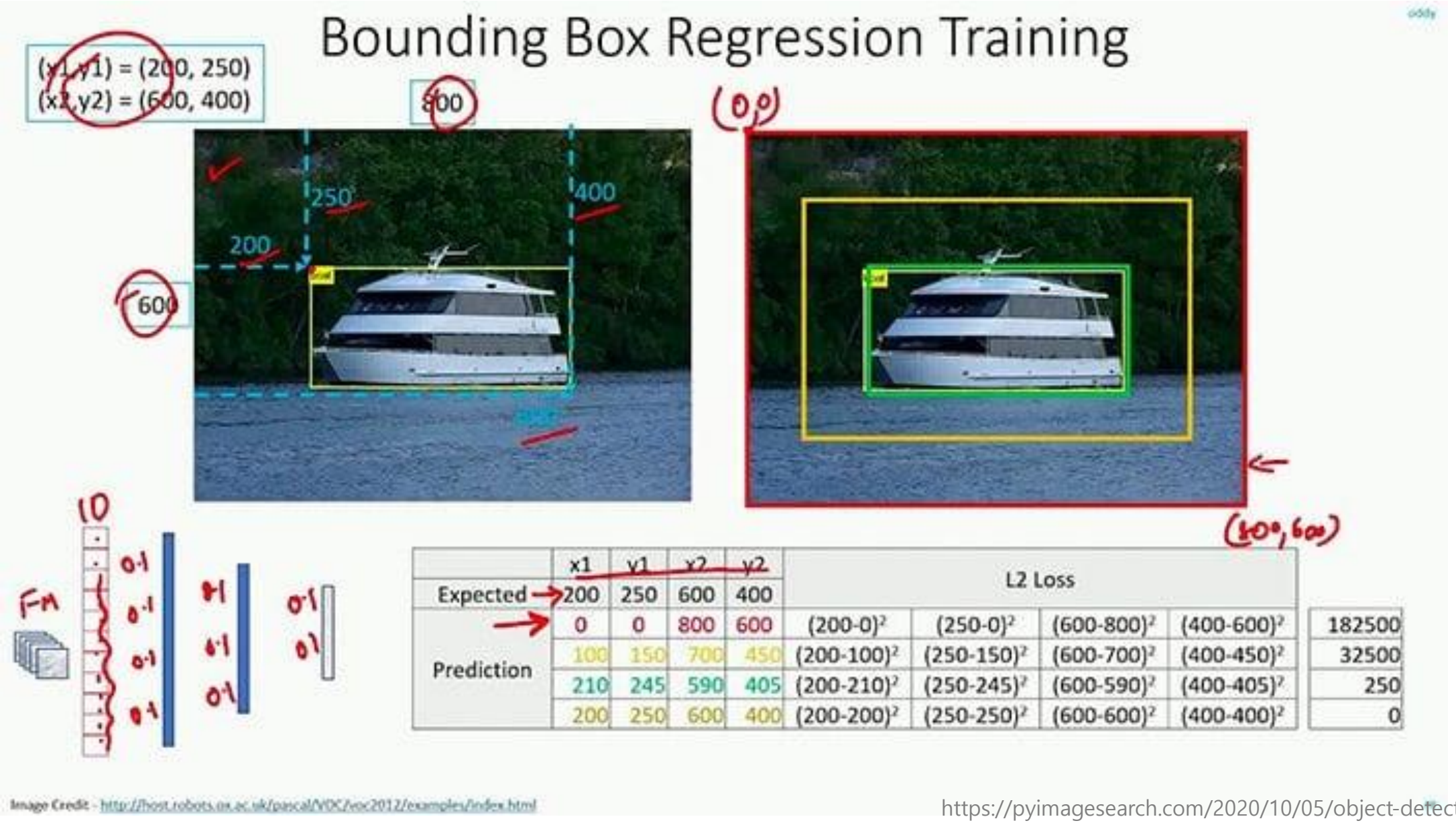
SVM

JRR Uijlings, 2012, Selective Search for Object Recognition, http://www.huppelen.nl/publications/selectiveSearchDraft.pdf



Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – bounding-box regressor

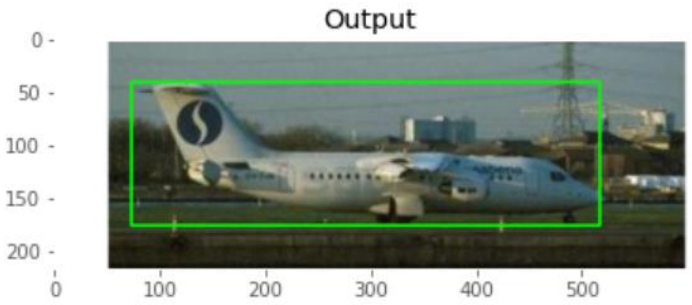


https://pyimagesearch.com/2020/10/05/object-detection-bounding-box-regression-with-keras-tensorflow-and-deep-learning/

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection

bounding-box regressor



Link

```
# load the image and preprocess it
image = load_img(imagePath, target_size=(224, 224))
image = img_to_array(image)

data.append(image)
targets.append((startX, startY, endX, endY))
filenames.append(filename)
...
split = train_test_split(data, targets, filenames, test_size=0.10,
  random_state=42)

(trainImages, testImages) = split[:2]
(trainTargets, testTargets) = split[2:4]
(trainFilenames, testFilenames) = split[4:]
```

https://pyimagesearch.com/2020/10/05/object-detection-bounding-box-regression-with-keras-tensorflow-and-deep-learning/

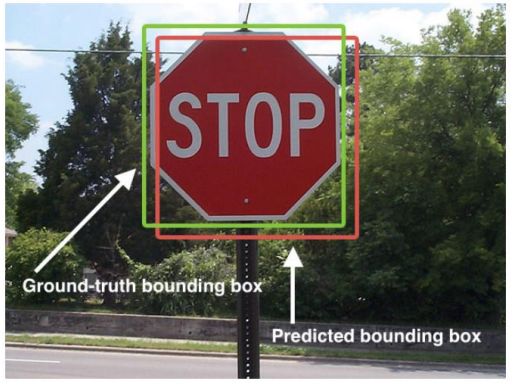
```
# flatten the max-pooling output of VGG
flatten = vgg.output
flatten = Flatten()(flatten)

# construct a fully-connected layer header to output the predicted
# bounding box coordinates
bboxHead = Dense(128, activation="relu")(flatten)
bboxHead = Dense(64, activation="relu")(bboxHead)
bboxHead = Dense(32, activation="relu")(bboxHead)
bboxHead = Dense(4, activation="sigmoid")(bboxHead)

# construct the model we will fine-tune for bounding box regression
model = Model(inputs=vgg.input, outputs=bboxHead)
opt = Adam(lr=config.INIT_LR)
model.compile(loss="mse", optimizer=opt)
print(model.summary())

# train the network for bounding box regression
print("[INFO] training bounding box regressor...")
H = model.fit(
        trainImages, trainTargets,
        validation_data=(testImages, testTargets),
        batch_size=config.BATCH_SIZE,
        epochs=config.NUM_EPOCHS,
        verbose=1)


print("[INFO] loading object detector...")
model = load_model(config.MODEL_PATH)

preds = model.predict(image)[0]
(startX, startY, endX, endY) = preds
```

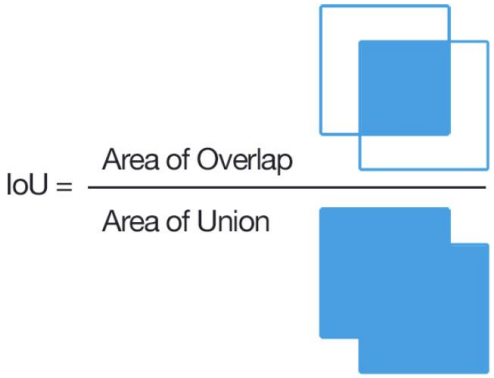
Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – Non-Maximum Suppression(NMS)

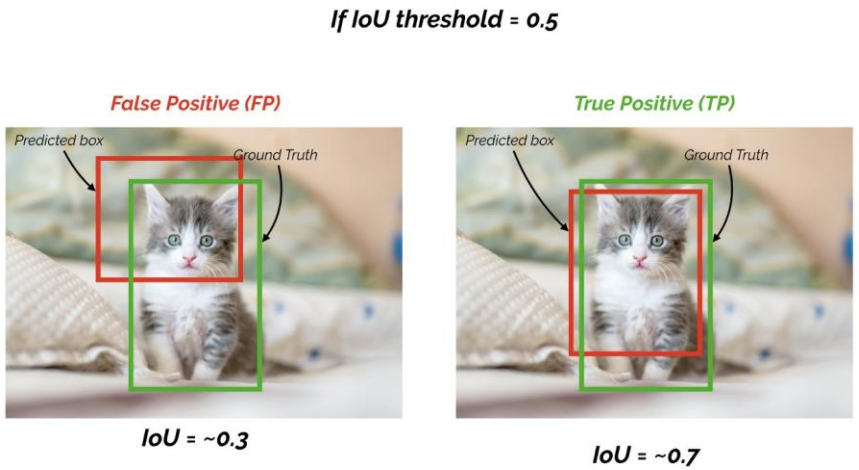
Dive

Intersection over union(IoU)






```
def calc_iou(b1, b2):
    # determine the (x, y)-coordinates of the intersection rectangle
    xA = max(b1[0], b2[0])
    yA = max(b1[1], b2[1])
    xB = min(b1[2], b2[2])
    yB = min(b1[3], b2[3])
    # compute the area of intersection rectangle
    area_intersect = (xB - xA) * (yB - yA)

    # Calculate area of boxes
    area_b1 = (b1[2] - b1[0]) * (b1[3] - b1[1])
    area_b2 = (b2[2] - b2[0]) * (b2[3] - b2[1])

    # areas - the intersection area
    epsilon = 1e-6
    iou = area_intersect / float(area_b1 + area_b2 - area_intersect +
epsilon)
    return iou
```



Ex. mAP@0.3 mAP@0.77

Kawshik Kumar Paul, Autonomous driving - Car detection, kawshikbuet17.github.io/Coursera-Deep-Learning/04-Convolutional-Neural-Networks/Codes/week3/Car%20detection%20for%20Autonomous%20Driving/Autonomous_driving_application_-_Car_detection_v3a.html

Copyright KAST CAMPUS Corp. All Rights Reserved

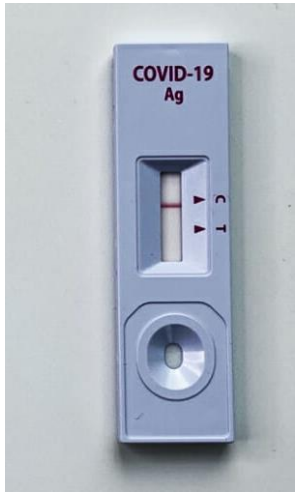
Object Detection – mean Average Precision(mAP)

FPS (Frames per second)

Recall and precision

TP(True Positive)=진양성 TN(True Negative)=진음성

FP(False Positive)=위양성 FN(False Negative)=위음성



$$recall = \frac{TP}{TP + FN}$$

재현율

$$precision = \frac{TP}{TP + FP}$$

정밀도

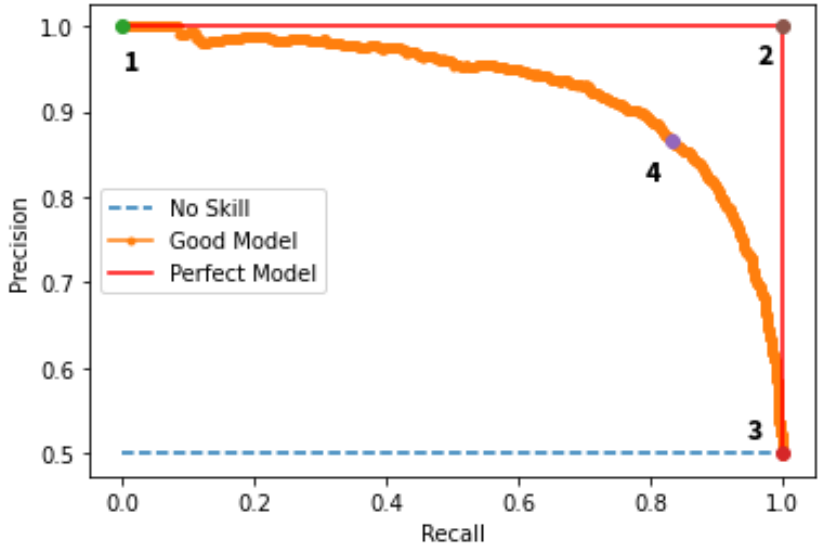
Recall > mistaken

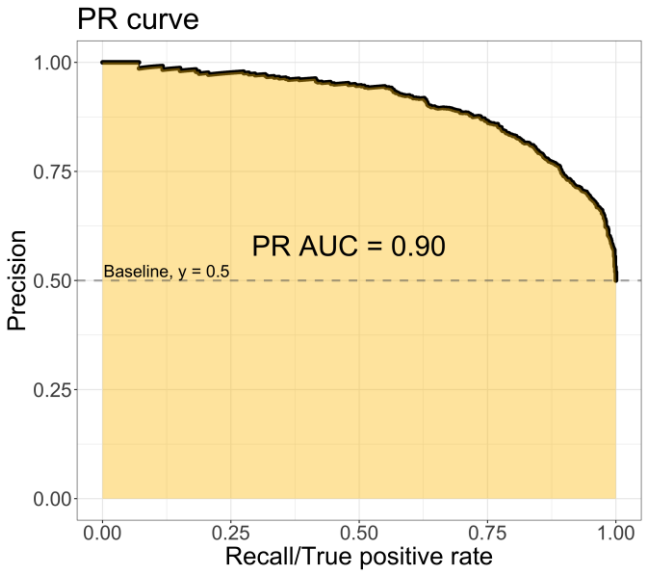
Deval Shah, 2022, Mean Average Precision (mAP) Explained: Everything You Need to Know, V7

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – mean Average Precision(mAP)

PR curve





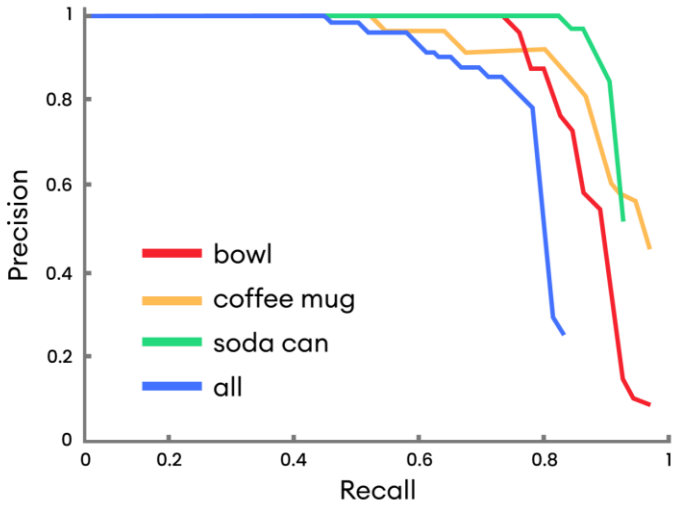
AP = AUC (Area Under the curve)

mAP

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$$AP_k = \text{ the AP of class } k$$

$$n = \text{ the number of classes}$$



1. Objectness score of each boxes
2. IoU
3. Class prediction
4. PR curve under IoU tolerance mAP@0.5
5. AUC
6. mAP

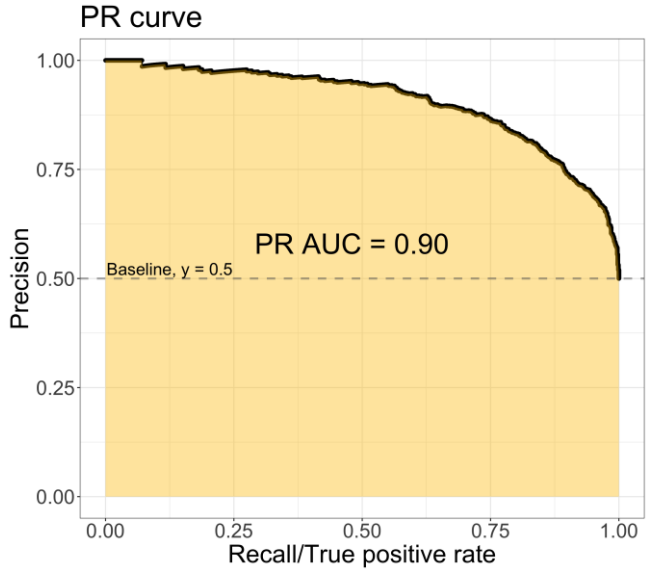
AISHIT DHARWAL, 2021.11, Complete Guide to Understanding Precision and Recall Curves

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox, 2011, Sparse Distance Learning for Object Recognition Combining RGB and Depth Information, IEEE

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – mean Average Precision(mAP)

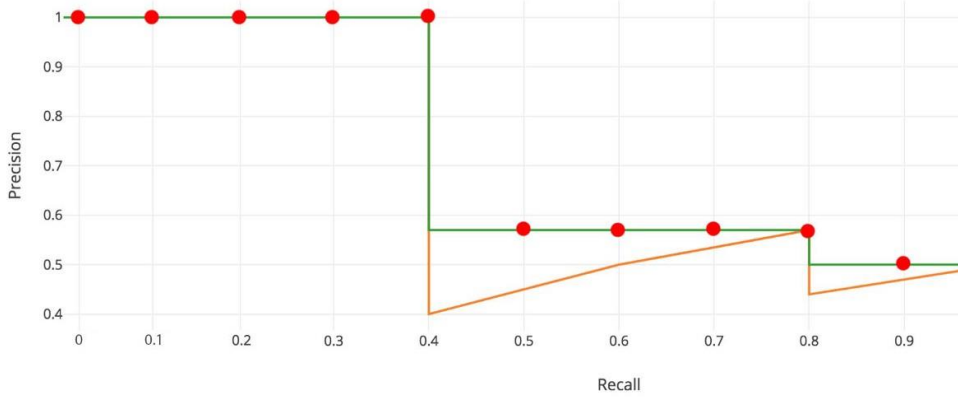
PR curve



AP = AUC (Area Under the curve)

Interpolated AP

Pascal VOC2008



$$AP = \frac{1}{11} \sum_{r \in \{0.0,\ldots,1.0\}} AP_r$$

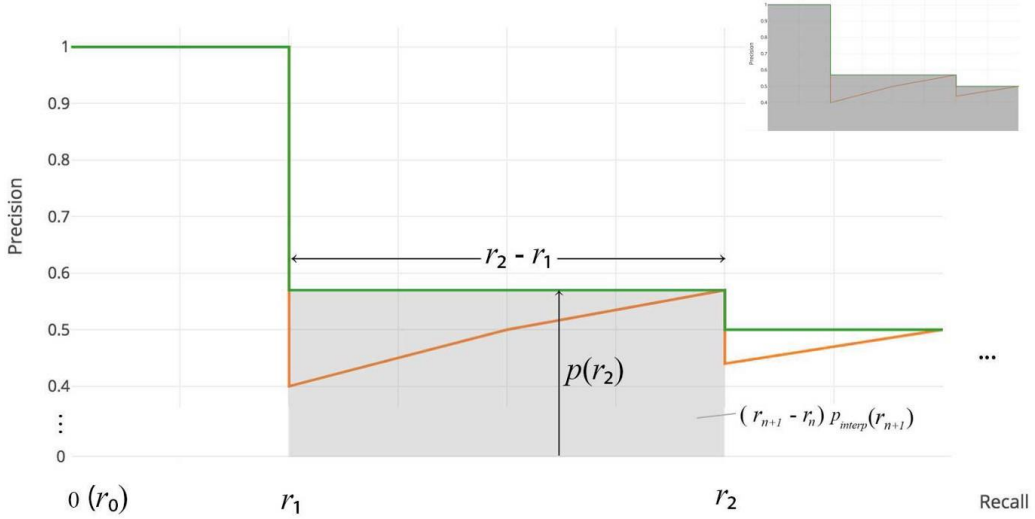
$$= \frac{1}{11} \sum_{r \in \{0.0,\ldots,1.0\}} p_{interp}(r)$$

$$AP = \frac{1}{11} \times \left(AP_r(0) + AP_r(0.1) + \ldots + AP_r(1.0)\right)$$

$$p_{interp}(r) = \max_{\tilde{r} \geq r} p(\tilde{r})$$

AP = (5 × 1.0 + 4 × 0.57 + 2 × 0.5)/11

r̃=recall



$$\mathrm{AP} = \Sigma\, (r_{n+1} - r_n)\, p_{interp}(r_{n+1})$$

$$p_{interp}(r_{n+1}) = \max_{\tilde{r} \geq r_{n+1}} p(\tilde{r})$$

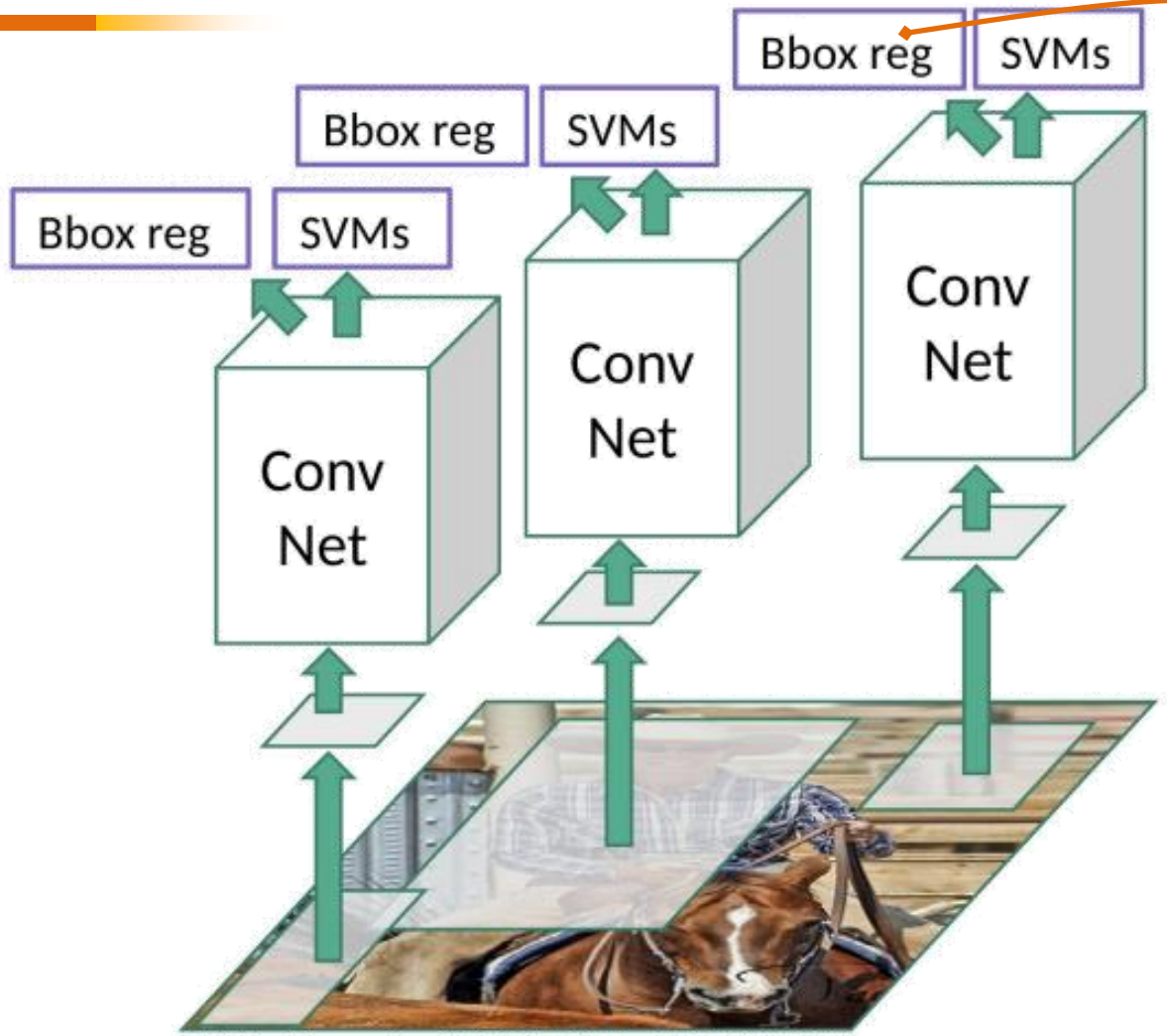
	backbone	AP	AP_{50}	AP_{75}
Two-stage methods				
Faster R-CNN+++ [3]	ResNet-101-C4	34.9	55.7	37.4
Faster R-CNN w FPN [6]	ResNet-101-FPN	36.2	59.1	39.0

COCO for YOLOv3

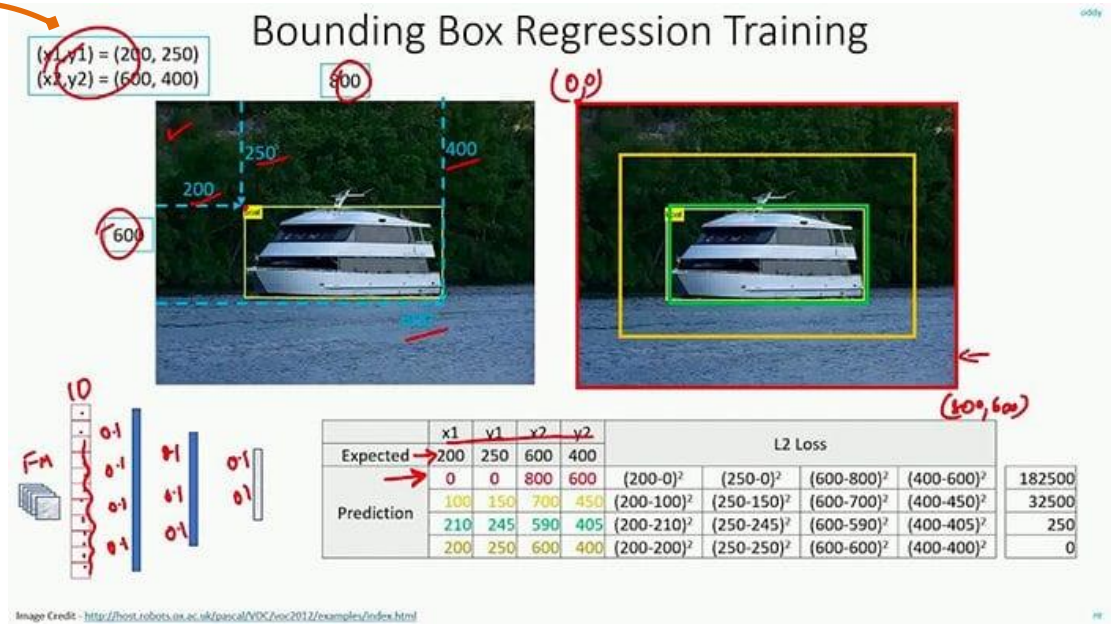
Jonathan Hui, 2018, mAP (mean Average Precision) for Object Detection

s Reserved

Object Detection – R-CNN region-based CNN



Ross Girshick, 2013, Rich feature hierarchies for accurate object detection and semantic segmentation, https://arxiv.org/pdf/1311.2524.pdf



https://pyimagesearch.com/2020/10/05/object-detection-bounding-box-regression-with-keras-tensorflow-and-deep-learning/

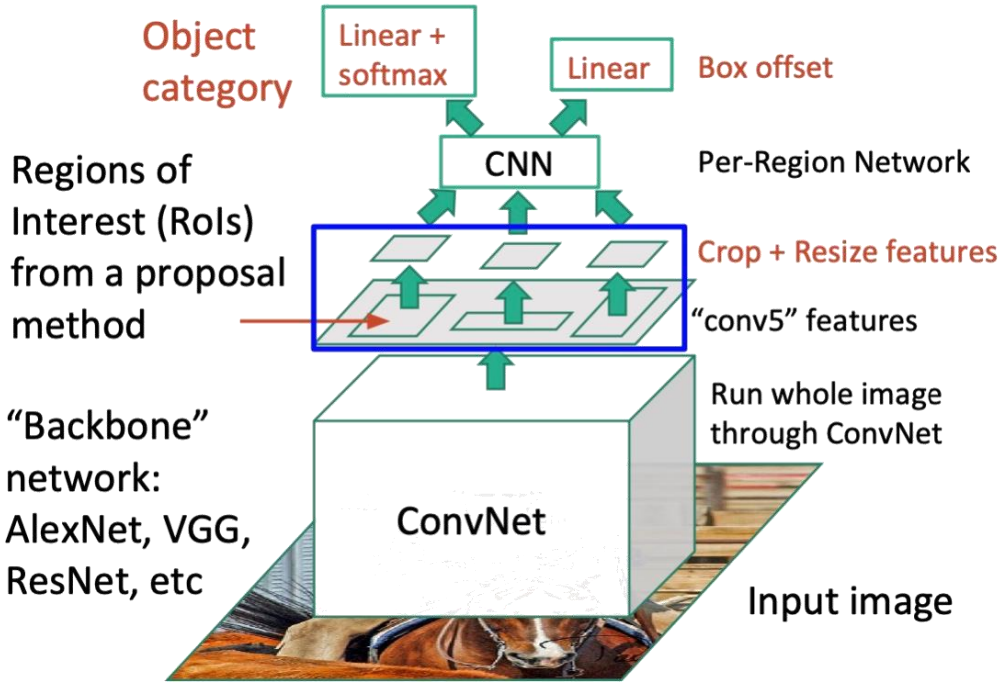
1. RoI Extraction
2. Feature Extraction with CNN
3. Classification with SVN (Support Vector Machine)
4. BBox prediction with regressor

Disadvantage

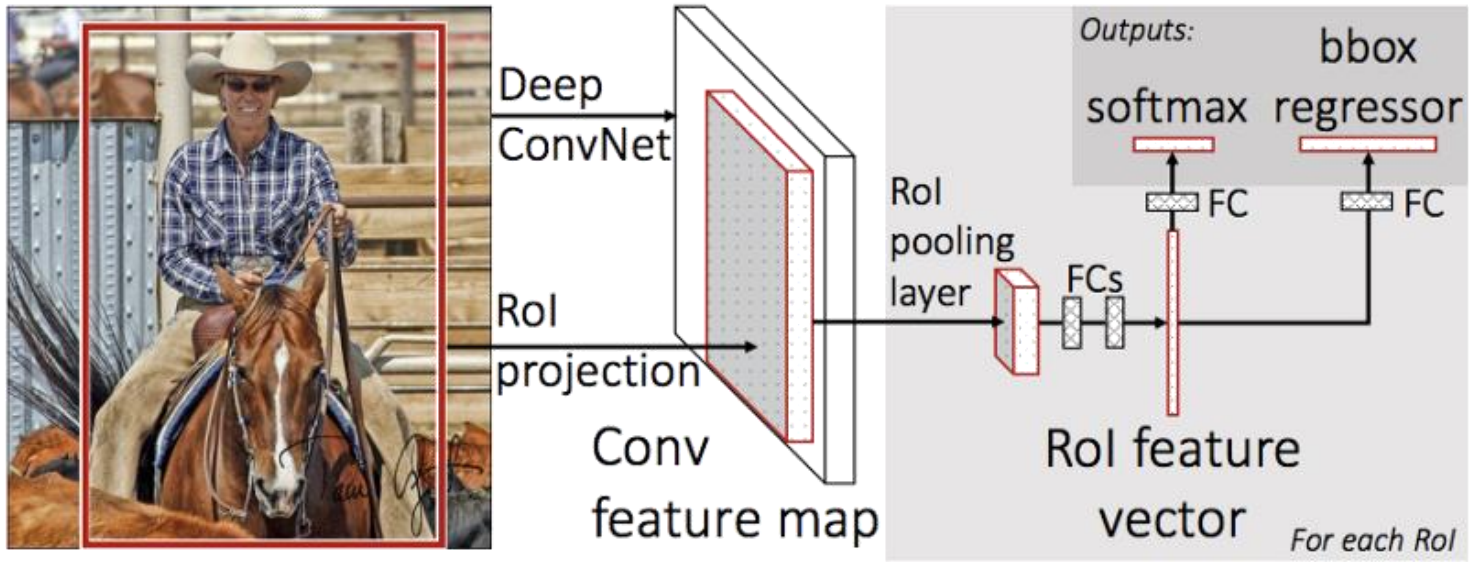
Slow. RoI over 2000 > CNN
Separated process. R-CNN, SVN, Bbox reg
Large storage volume of extracted features

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – Fast R-CNN



Girshick, "Fast R-CNN", ICCV 2015. Figure copyright Ross Girshick, 2015; source. Reproduced with permission.



Ross Girshick, 2015, Fast R-CNN, https://arxiv.org/abs/1504.08083

1. Full images > Feature extraction with ConvNet
2. RoI from feature map
3. RoI Pooling to fix region size (H x W) > FCs
4. FC > Softmax for classification probability of RoI list
5. FC > Bbox regression

Loss of classification(L_{cls}) + Loss of location(L_{loc})

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1]L_{\text{loc}}(t^u, v), \quad (1)$$

in which $L_{\text{cls}}(p, u) = -\log p_u$ is log loss for true class u.

The second task loss, L_{loc}, is defined over a tuple of true bounding-box regression targets for class u, $v = (v_{\text{x}}, v_{\text{y}}, v_{\text{w}}, v_{\text{h}})$, and a predicted tuple $t^u = (t^u_{\text{x}}, t^u_{\text{y}}, t^u_{\text{w}}, t^u_{\text{h}})$, again for class u. The Iverson bracket indicator function $[u \geq 1]$ evaluates to 1 when $u \geq 1$ and 0 otherwise. By convention the catch-all background class is labeled $u = 0$.

Disadvantage. Separated selective search.

Aman Chadha, aman.ai, https://aman.ai/cs231n/detection/

Fast R-CNN is written in Python and C++ (Caffe)

Copyright FAST CAMPUS Corp. All Rights Reserved

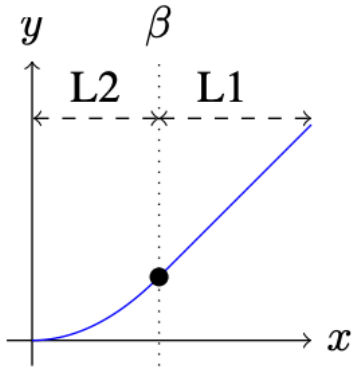
Object Detection – Fast R-CNN

Fast campus

Loss of location(L_{loc})

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i^u - v_i), \qquad (2)$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \qquad (3)$$



(a) Smooth L1

Model

```
# load the VGG16 network, ensuring the head FC layers are left off
vgg = VGG16(weights="imagenet", include_top=False,
            input_tensor=Input(shape=(224, 224, 3)))
# freeze all VGG layers so they will *not* be updated during the
# training process
vgg.trainable = False
# flatten the max-pooling output of VGG
flatten = vgg.output
flatten = Flatten()(flatten)
# construct a fully-connected layer header to output the predicted
# bounding box coordinates
bboxHead = Dense(128, activation="relu")(flatten)
bboxHead = Dense(64, activation="relu")(bboxHead)
bboxHead = Dense(32, activation="relu")(bboxHead)
bboxHead = Dense(4, activation="sigmoid")(bboxHead)
# construct the model we will fine-tune for bounding box regression
model = Model(inputs=vgg.input, outputs=bboxHead)
```

FC > Bbox regression

Ross Girshick
https://github.com/rbgirshick

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – Faster R-CNN

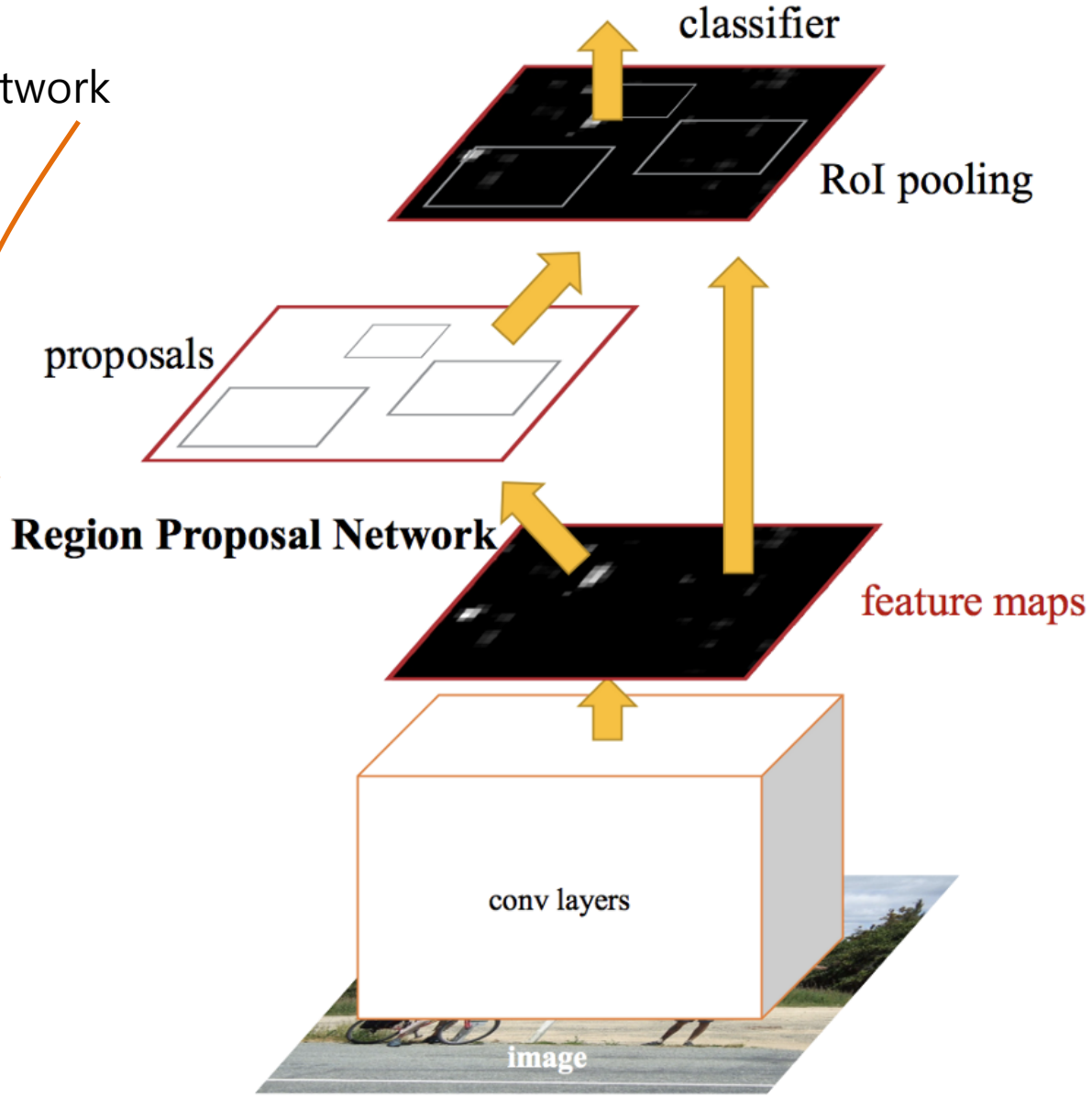
Selective search -> RPN (region proposal network
attention network)

RPN output = {objectness score, Bbox}

Model = VGG 16

(MobileNet, DenseNet, ResNet152)

1. Feature extraction
2. RPN
3. RoI pooling
4. Classification
5. Bbox regression



Shaoging Ren, 2015, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, https://arxiv.org/pdf/1506.01497.pdf

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – Faster R-CNN

RPN (region proposal network, attention network)

Conv3x3. 512 channel
Conv1x1 for background, foreground decision
Conv1x1 for Bbox regression

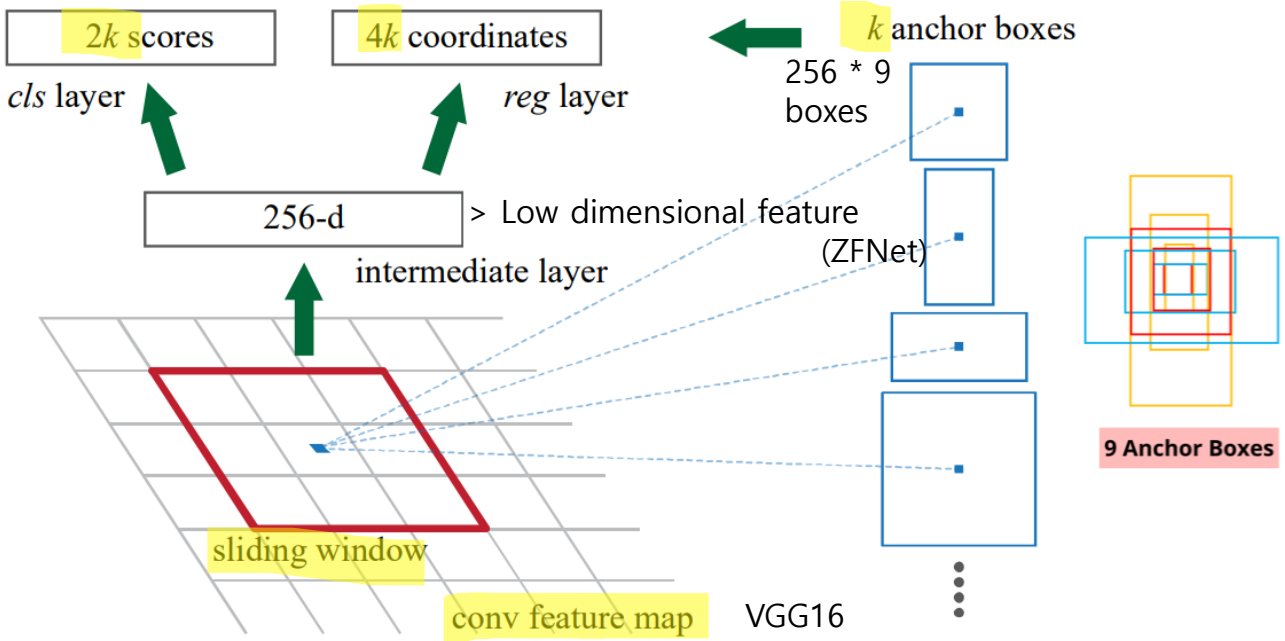


Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Exampl
VOC 2007 test. Our method detects objects in a wide range of

3.1.1 Anchors

At each sliding-window location, we simultaneously predict multiple region proposals, where the number of maximum possible proposals for each location is denoted as k. So the *reg* layer has $4k$ outputs encoding the coordinates of k boxes, and the *cls* layer outputs $2k$ scores that estimate probability of object or not object for each proposal[4]. The k proposals are parameterized *relative* to k reference boxes, which we call *anchors*. An anchor is centered at the sliding window in question, and is associated with a scale and aspect ratio (Figure 3, left). By default we use 3 scales and 3 aspect ratios, yielding $k = 9$ anchors at each sliding position. For a convolutional feature map of a size $W \times H$ (typically ~2,400), there are WHk anchors in total.

Shaoging Ren, 2015, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, https://arxiv.org/pdf/1506.01497.pdf

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – Faster R-CNN

Loss function for training RPN

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*).$$

L_{cls} = objectness (2 classes)

p=predicted object probability
**p=ground-truth*

$$p = \begin{cases} 1, IoU > 0.7 \\ -1, IoU < 0.3 \\ 0 \end{cases}$$

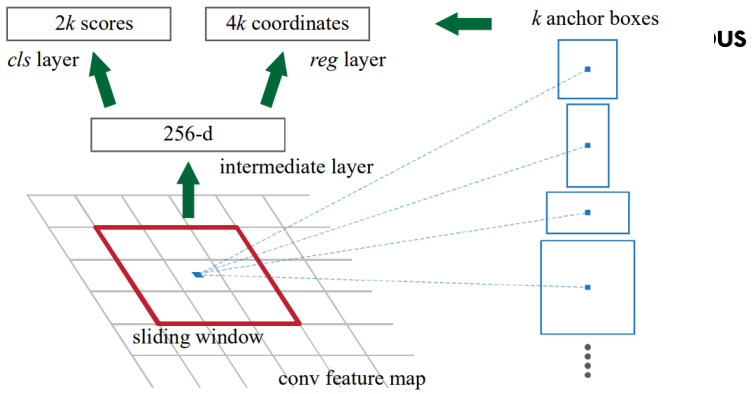


Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Exampl
VOC 2007 test. Our method detects objects in a wide range of

Here, i is the index of an anchor in a mini-batch and p_i is the predicted probability of anchor i being an object. The ground-truth label p_i^* is 1 if the anchor is positive, and is 0 if the anchor is negative. t_i is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t_i^* is that of the ground-truth box associated with a positive anchor. The classification loss L_{cls} is log loss over two classes (object *vs.* not object). For the regression loss, we use $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ where R is the robust loss function (smooth L_1) defined in [2]. The term $p_i^* L_{reg}$ means the regression loss is activated only for positive anchors ($p_i^* = 1$) and is disabled otherwise ($p_i^* = 0$). The outputs of the *cls* and *reg* layers consist of $\{p_i\}$ and $\{t_i\}$ respectively.

The two terms are normalized by N_{cls} and N_{reg} and weighted by a balancing parameter λ. In our current implementation (as in the released code), the *cls* term in Eqn.(1) is normalized by the mini-batch size (*i.e.*, $N_{cls} = 256$) and the *reg* term is normalized by the number of anchor locations (*i.e.*, $N_{reg} \sim 2,400$). By default we set $\lambda = 10$, and thus both *cls* and *reg* terms are roughly equally weighted. We show by experiments that the results are insensitive to the values of λ in a wide range (Table 9). We also note that the normalization as above is not required and could be simplified.

Shaoging Ren, 2015, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, https://arxiv.org/pdf/1506.01497.pdf

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – Faster R-CNN

Fast campus

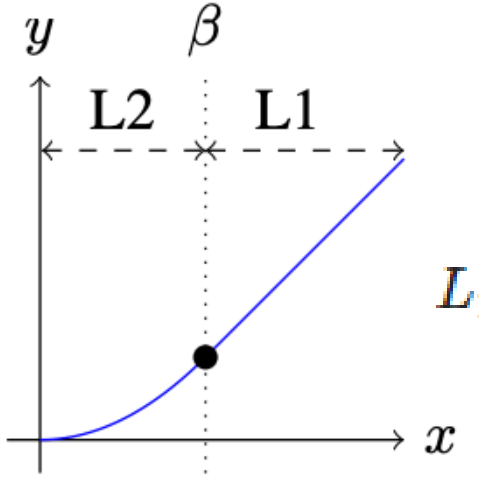
Multiple Loss Function

$$\mathcal{L}_{\text{cls}}(p_i, p_i^*) = -p_i^* \log p_i - (1 - p_i^*) \log(1 - p_i)$$

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}}$$

$\lambda \leq 10$

$$\mathcal{L}(\{p_i\}, \{t_i\}) = \frac{1}{N_{\text{cls}}} \sum_i \mathcal{L}_{\text{cls}}(p_i, p_i^*) + \frac{\lambda}{N_{\text{box}}} \sum_i p_i^* \cdot L_1^{\text{smooth}}(t_i - t_i^*)$$



$$L_{1;smooth} = \begin{cases} |x| & \text{if } |x| > \alpha; \\ \frac{1}{|\alpha|} x^2 & \text{if } |x| \leq \alpha \end{cases}$$

(a) Smooth L1

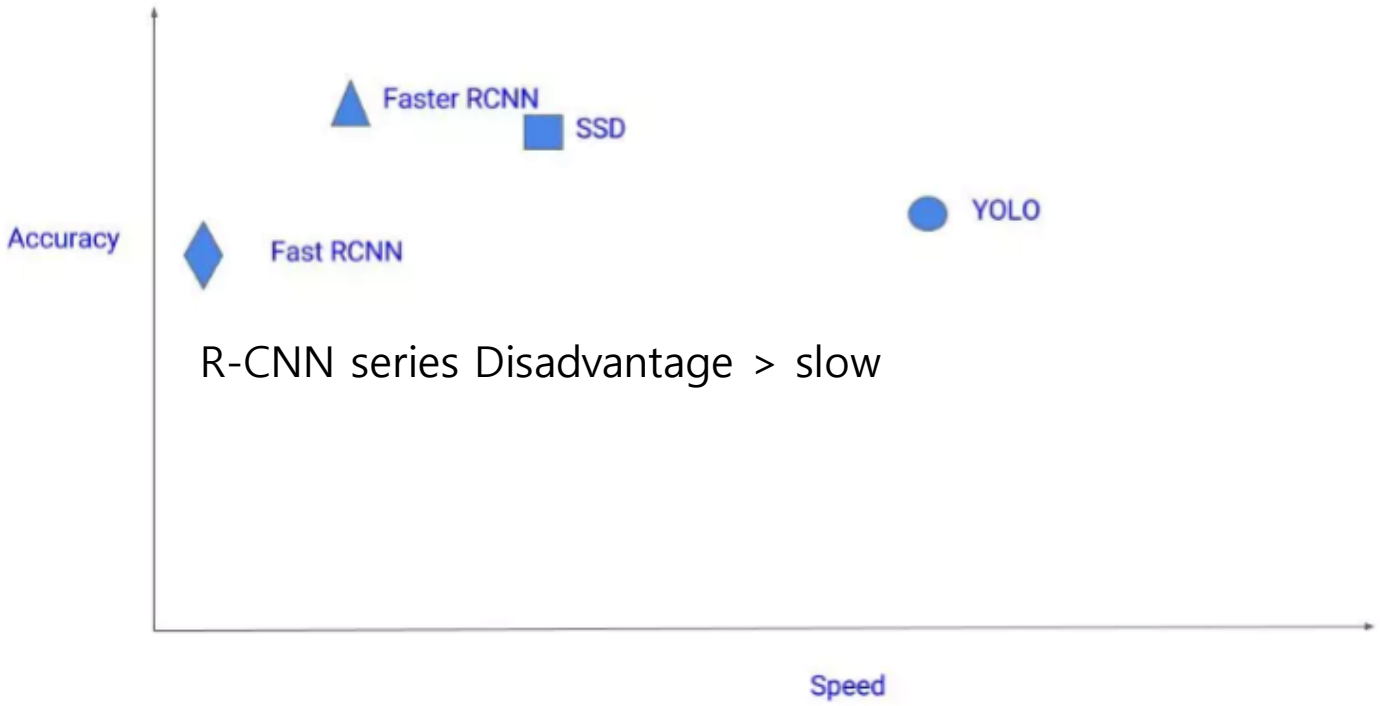
Pranav Raikote, 2021, Object Detection , Applied Singularity, appliedsingularity.com/2021/06/08/object-detection-part-5-faster-r-cnn/

Pranav Raikote, 2021, Object Detection , Applied Singularity, appliedsingularity.com/2021/06/08/object-detection-part-5-faster-r-cnn/

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – Faster R-CNN

	R-CNN	Fast R-CNN	Faster R-CNN
Test Time per Image	50 Seconds	2 Seconds	0.2 Seconds
Speed Up	1x	25x	250x



R-CNN series Disadvantage > slow

Ankit Sachan, Zero to Hero: Guide to Object Detection using Deep Learning: Faster R-CNN,YOLO,SSD. https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/

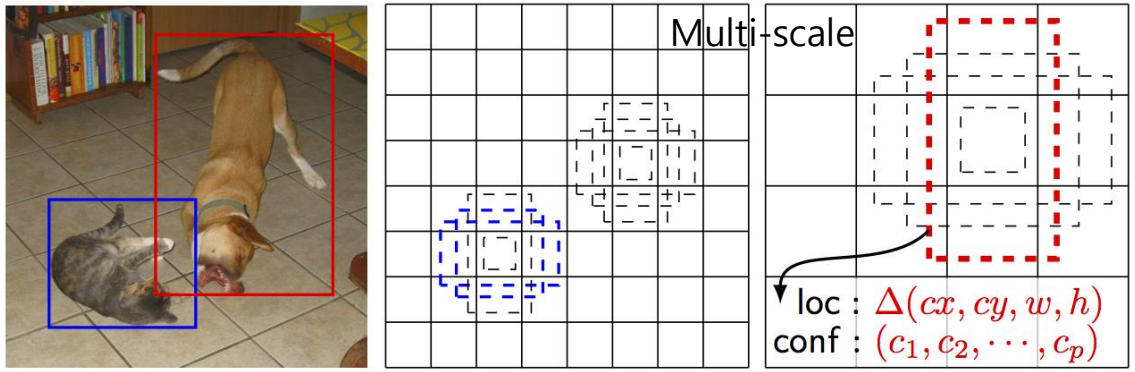
Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – Single-shot detector(SSD)

SSD300 = 59 fps, mAP 74.3%
SSD512 (512 x 512) = 22 fps, mAP 76.8%

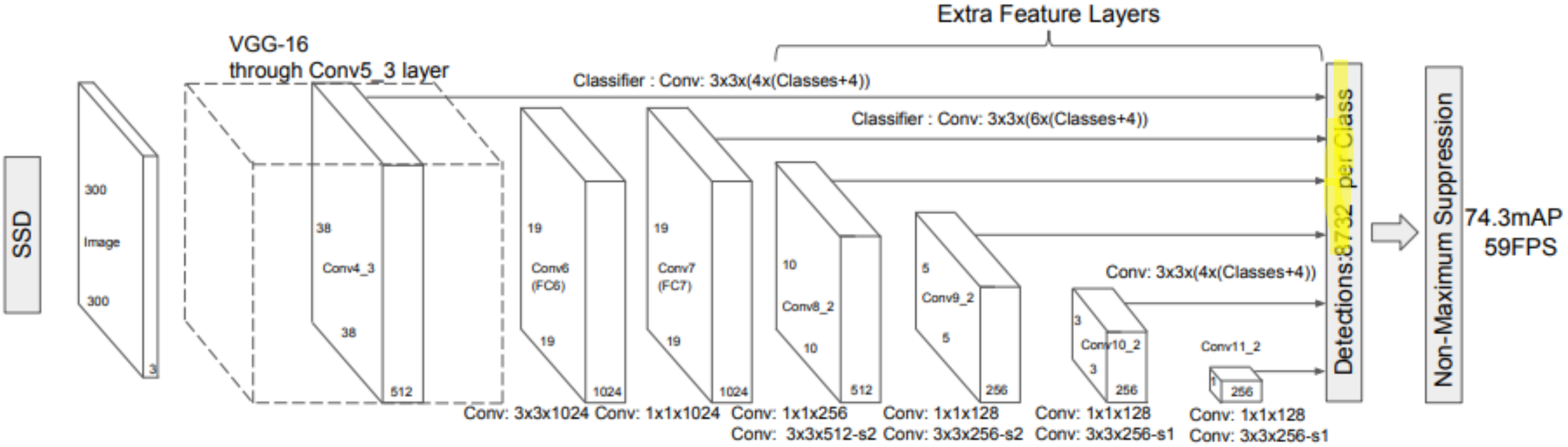
Feature extraction. VGG16
Multi-scale conv
NMS

Output = {cx, cy, w, h, objectness, C1…Cn}



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Conv4_3 = 38 x 38 x 4 = 5,776
Conv7 = 19 x 19 x 6 = 2,166
Conv8_2 = 10 x 10 x 6 = 600
Conv9_2 = 5 x 5 x 6 = 150
Conv10_2 = 3 x 3 x 4 = 36
Conv11_2 = 1 x 1 x 4 = 4



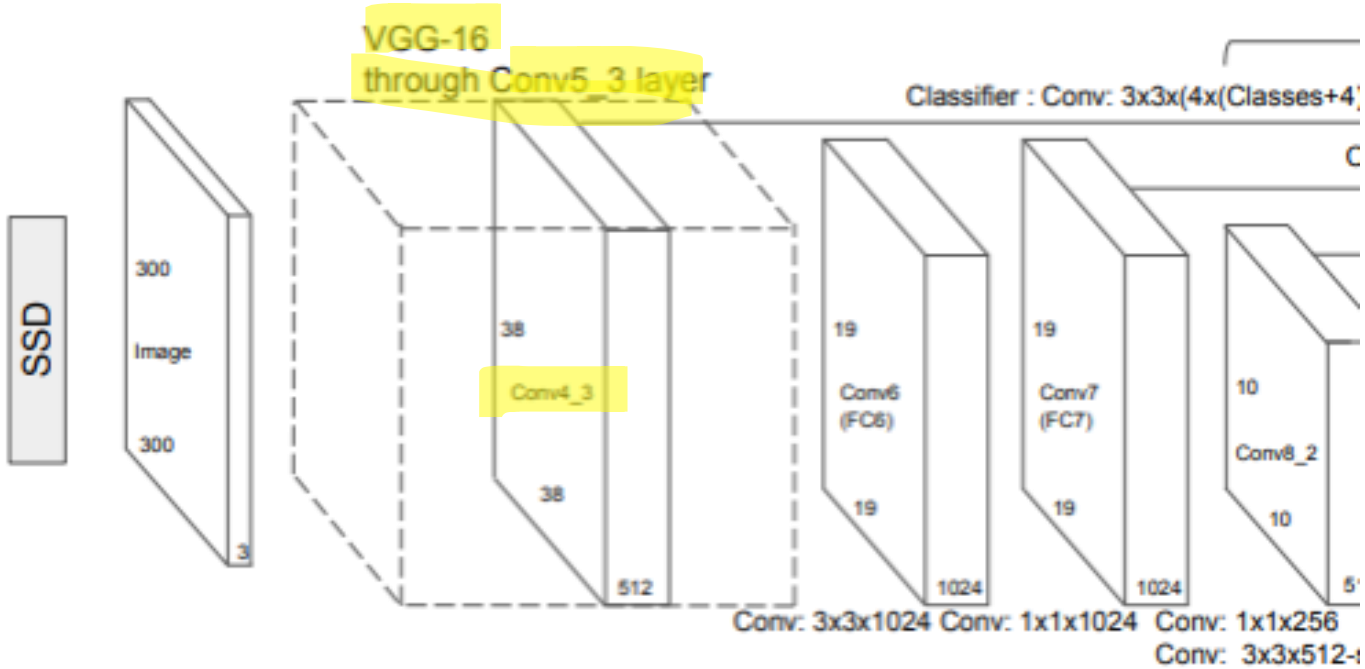
Wei Liu, 2016, SSD: Single Shot MultiBox Detector, https://arxiv.org/pdf/1512.02325.pdf

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – Single-shot detector(SSD)

Fast campus

Model




```python
def VGG16_SSD(img_height = 300, img_width = 300, img_channels = 3):
    x1 = Input(shape=(img_height, img_width, img_channels))
    l2_reg = 0.0005

    conv1_1 = Conv2D(64, (3, 3), activation='relu', padding='same',
    conv1_2 = Conv2D(64, (3, 3), activation='relu', padding='same',
    pool1 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding=

    conv2_1 = Conv2D(128, (3, 3), activation='relu', padding='same'
    conv2_2 = Conv2D(128, (3, 3), activation='relu', padding='same'
    pool2 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding=

    conv3_1 = Conv2D(256, (3, 3), activation='relu', padding='same'
    conv3_2 = Conv2D(256, (3, 3), activation='relu', padding='same'
    conv3_3 = Conv2D(256, (3, 3), activation='relu', padding='same'
    pool3 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding=

    conv4_1 = Conv2D(512, (3, 3), activation='relu', padding='same'
    conv4_2 = Conv2D(512, (3, 3), activation='relu', padding='same'
    conv4_3 = Conv2D(512, (3, 3), activation='relu', padding='same'
    pool4 = MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding=

    conv5_1 = Conv2D(512, (3, 3), activation='relu', padding='same'
    conv5_2 = Conv2D(512, (3, 3), activation='relu', padding='same'
    conv5_3 = Conv2D(512, (3, 3), activation='relu', padding='same'
    pool5 = MaxPooling2D(pool_size=(3, 3), strides=(1, 1), padding=

    model = Model(inputs=x1, outputs=pool5)
    return model
```

Loss function

$$L(x,c,l,g) = \frac{1}{N}(L_{conf}(x,c) + \alpha L_{loc}(x,l,g))$$

$N = matched\ boxes\ number, l = predicted\ box, g = ground\ truth\ box$

Let $x_{ij}^{p} = \{1,0\}$

indicator for matching the i-th default box to the j-th ground truth box of category p.

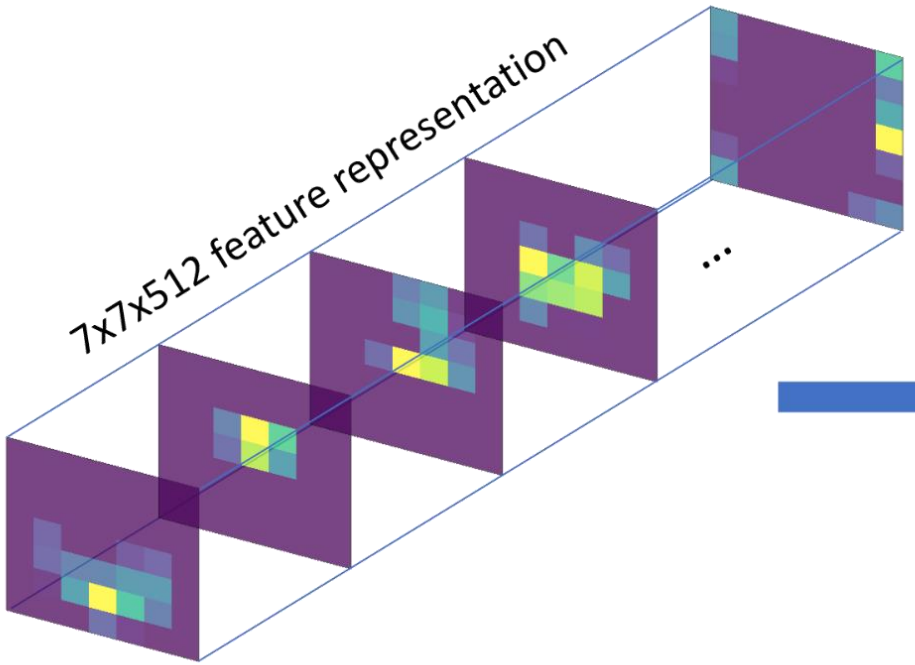
Eddie Forson, 2017, Understanding SSD MultiBox — Real-Time Object Detection In Deep Learning, towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

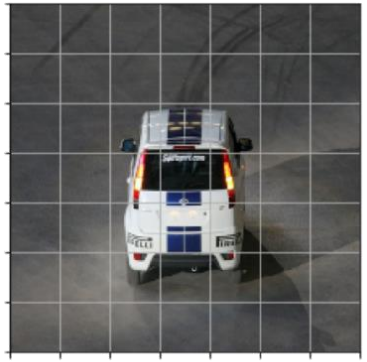
Copyright FAST-CAMPUS Corp. All Rights Reserved

Object Detection – Single-shot detector(SSD)

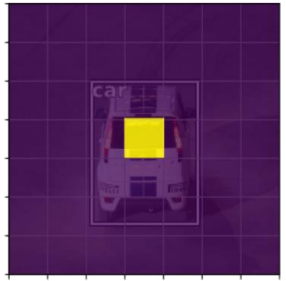
Multi-scale feature map

- Anchor boxes of each grid
- Classification of each box
- NMS

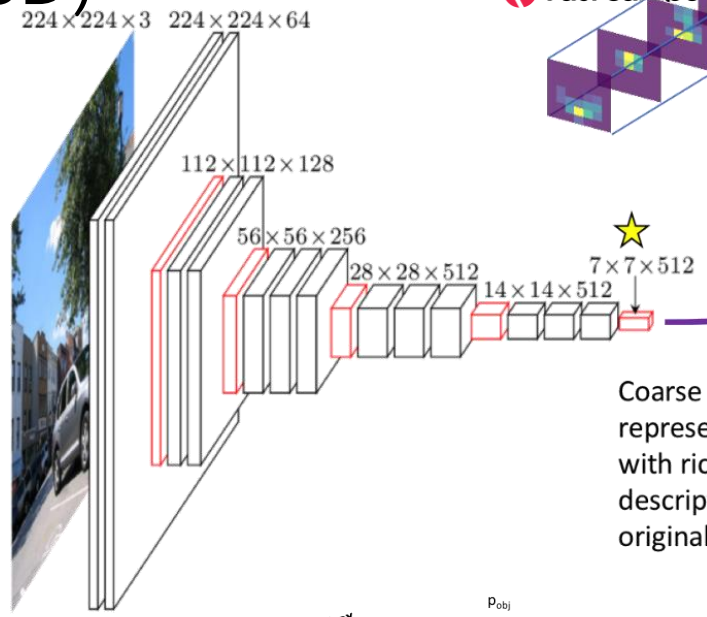


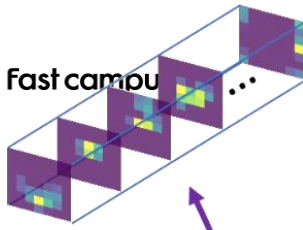


Visualizing the corresponding regions of each "pixel" in the 7x7 feature maps with the original 224x224 image

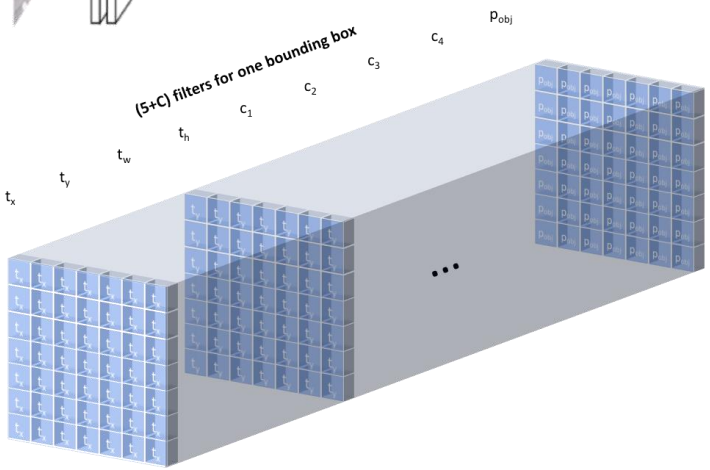


predict





Coarse spatial representation with rich feature description of original image

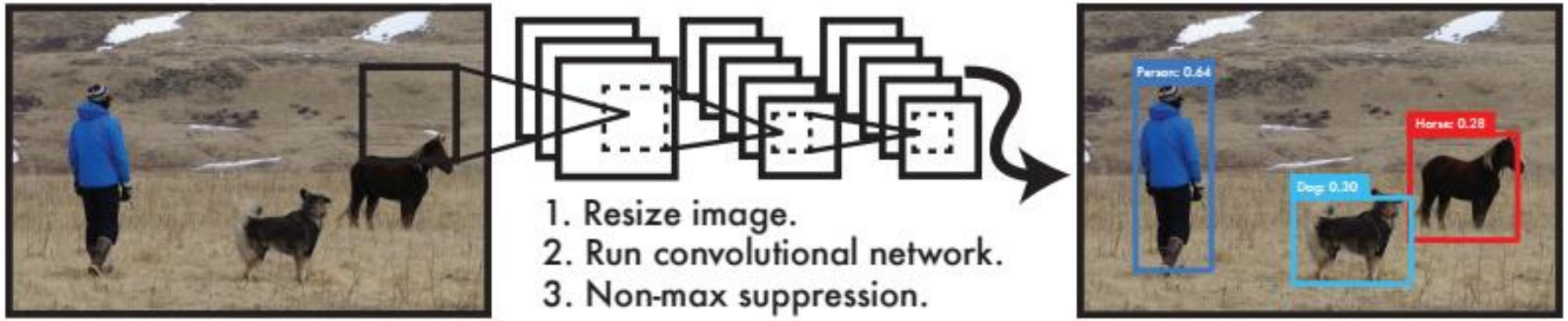


Learning 2 bounding boxes for each grid cell requires 18 channels (assuming 4 possible classes)

JEREMY JORDAN, 2018, An overview of object detection: one-stage methods. https://www.jeremyjordan.me/object-detection-one-stage/

Copyright FAST CAMPUS Corp. All Rights Reserved

Object Detection – YOLO



Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Joseph Redmon, 2016, You Only Look Once: Unified, Real-Time Object Detection, https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Redmon_You_Only_Look_CVPR_2016_paper.pdf

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO – paper review

Fast campus

You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

1. Introduction

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little con-

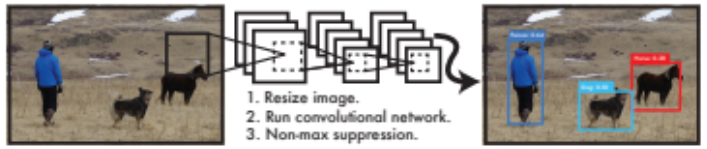


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bounding boxes, eliminate duplicate detections, and rescore the boxes based on other objects in the scene [13]. These complex pipelines are slow and hard to optimize because each individual component must be trained separately.

We reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using our system, you only look once (YOLO) at an image to predict what objects are present and where they are.

YOLO is refreshingly simple: see Figure 1. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection.

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO – paper review

Fast campus

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localization errors but is less likely to predict false positives on background. Finally, YOLO learns very general representations of objects. It outperforms other detection methods, including DPM and R-CNN, when generalizing from natural images to other domains like artwork.

Copyright **FAST CAMPUS** Corp. All Rights Reserved

YOLO – paper review

Our system divides the input image into an $S \times S$ grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.

Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as $\Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$. If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box.

Each grid cell also predicts C conditional class probabilities, $\Pr(\text{Class}_i|\text{Object})$. These probabilities are conditioned on the grid cell containing an object. We only predict one set of class probabilities per grid cell, regardless of the number of boxes B.

At test time we multiply the conditional class probabilities and the individual box confidence predictions,

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}} \quad (1)$$

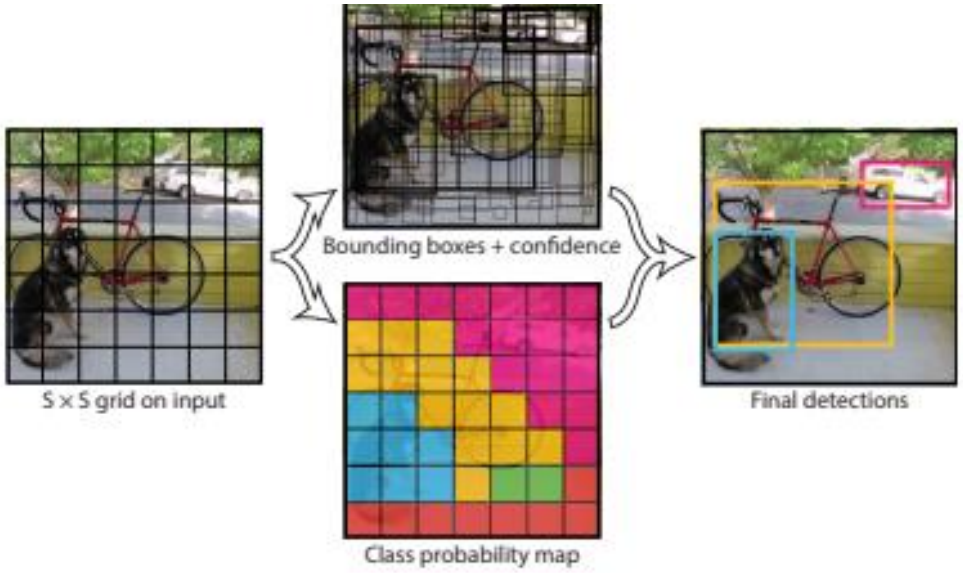
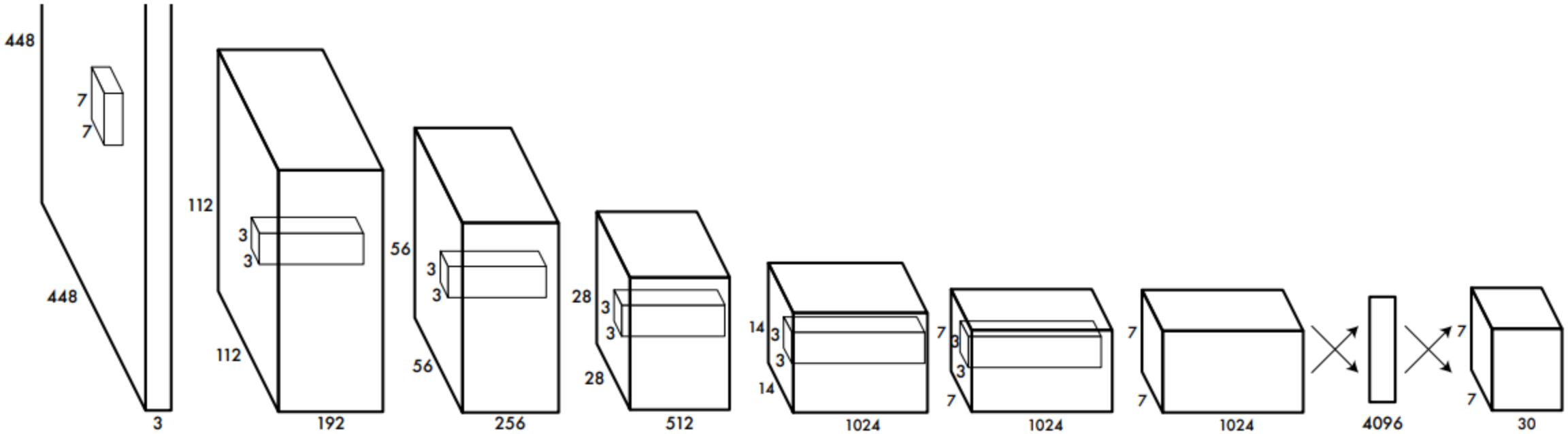


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

For evaluating YOLO on PASCAL VOC, we use $S = 7$, $B = 2$. PASCAL VOC has 20 labelled classes so $C = 20$. Our final prediction is a $7 \times 7 \times 30$ tensor.

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO – paper review



Conv. Layer
7x7x64-s-2
Maxpool Layer
2x2-s-2

Conv. Layer
3x3x192
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x128
3x3x256
1x1x256
3x3x512
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x256
3x3x512
} ×4
1x1x512
3x3x1024
Maxpool Layer
2x2-s-2

Conv. Layers
1x1x512
3x3x1024
} ×2
3x3x1024
3x3x1024-s-2

Conv. Layers
3x3x1024
3x3x1024

Conn. Layer

Conn. Layer

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO – paper review

Fast campus

Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.

We use a linear activation function for the final layer and all other layers use the following leaky rectified linear activation:

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases} \tag{2}$$

We optimize for sum-squared error in the output of our model. We use sum-squared error because it is easy to op-

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO – paper review

loss function:

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

where $\mathbb{1}_i^{\text{obj}}$ denotes if object appears in cell i and $\mathbb{1}_{ij}^{\text{obj}}$ denotes that the jth bounding box predictor in cell i is "responsible" for that prediction.

- $\mathbb{1}_{ij}^{\text{obj}}$: It indicates whether the j-th bounding box of the cell i is "responsible" for the object prediction (see Fig. 3). if an object is present in that grid cell, $\mathbb{1}_i^{\text{obj}} = 1$.
- C_{ij}: The confidence score of cell i, `Pr(containing an object) * IoU(pred, truth)` .
- $\hat{C}_{ij}$: The predicted confidence score.
- $\mathcal{C}$: The set of all classes.
- $p_i(c)$: The conditional probability of whether cell i contains an object of class $c \in \mathcal{C}$.
- $\hat{p}_i(c)$: The predicted conditional class probability.

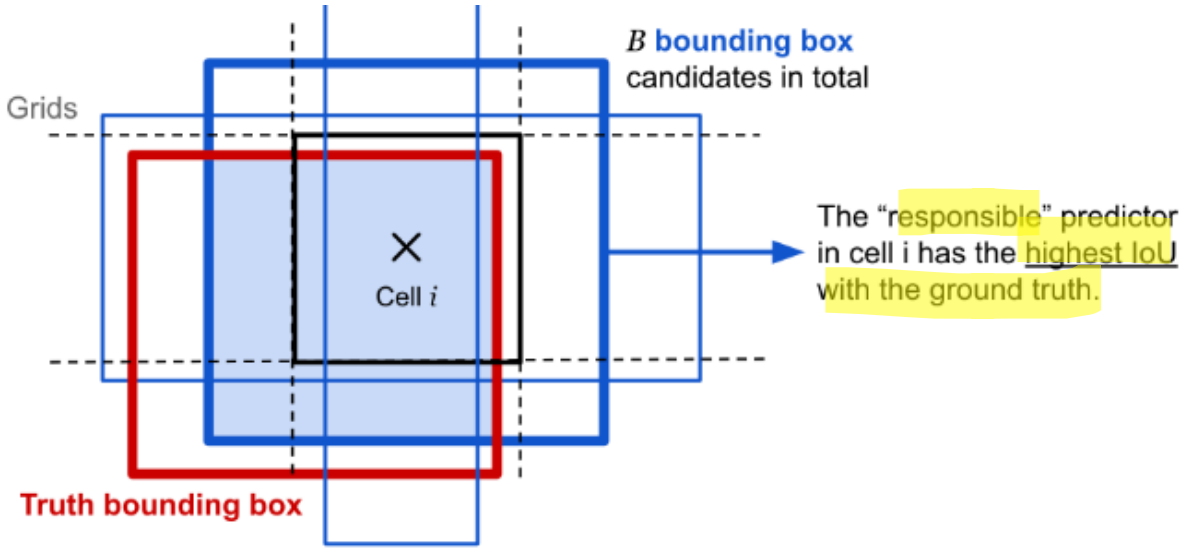


Fig. 3. At one location, in cell i, the model proposes B bounding box candidates and the one that has highest overlap with the ground truth is the "responsible" predictor.

Lilian Weng, 2018, Object Detection Part 4: Fast Detection Models, https://lilianweng.github.io/posts/2018-12-27-object-recognition-part-4/

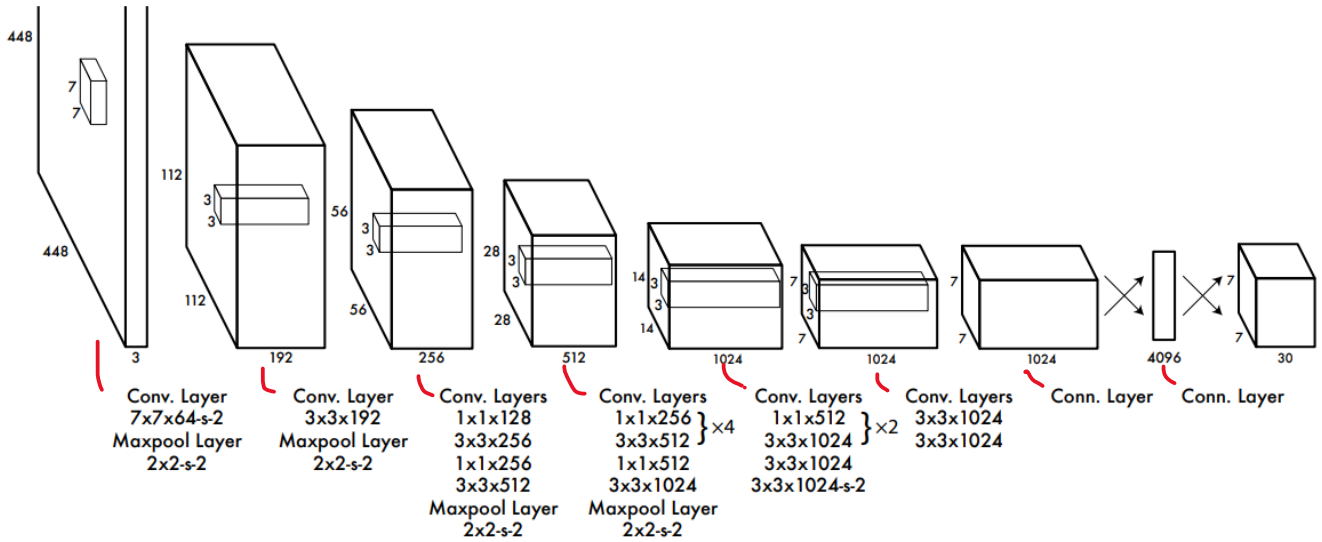
Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO - code

Implementing YOLOV1 from scratch using PyTorch

Fast campus

Model




```
architecture_config = [
    (7, 64, 2, 3),
    "M",
    (3, 192, 1, 1),
    "M",
    (1, 128, 1, 0),
    (3, 256, 1, 1),
    (1, 256, 1, 0),
    (3, 512, 1, 1),
    "M",
    [(1, 256, 1, 0), (3, 512, 1, 1), 4],
    (1, 512, 1, 0),
    (3, 1024, 1, 1),
    "M",
    [(1, 512, 1, 0), (3, 1024, 1, 1), 2],
    (3, 1024, 1, 1),
    (3, 1024, 2, 1),
    (3, 1024, 1, 1),
    (3, 1024, 1, 1),
]
```

```
CNNBlock(3, 64, kernel_size=7, strides=2, padding=3)
MaxPool2d(kernel_size=(2, 2), stride=(2,2))
CNNBlock(64, 192, kernel_size=3, strides=1, padding=1)
MaxPool2d(kernel_size=(2, 2), stride=(2,2))
CNNBlock(192, 128, kernel_size=1, strides=1, padding=0)
CNNBlock(128, 256, kernel_size=3, strides=1, padding=1)
CNNBlock(256, 256, kernel_size=1, strides=1, padding=0)
CNNBlock(256, 512, kernel_size=3, strides=1, padding=1)
MaxPool2d(kernel_size=(2, 2), stride=(2,2))
...
CNNBlock(512, 512, kernel_size=1, strides=1, padding=0)
CNNBlock(512, 1024, kernel_size=3, strides=1, padding=1)
MaxPool2d(kernel_size=(2, 2), stride=(2,2))
...
CNNBlock(1024, 1024, kernel_size=3, strides=1, padding=1)
CNNBlock(1024, 1024, kernel_size=3, strides=2, padding=1)
CNNBlock(1024, 1024, kernel_size=3, strides=1, padding=1)
CNNBlock(1024, 1024, kernel_size=3, strides=1, padding=1)

Flatten()
Linear(1024 * S * S, 4096)
LeakyReLU(0.1)
Linear(4096, S * S * (B * 5 + C)
```

CNNBlock(3, 64, kernel_size=7, strides=2, padding=3)
MaxPool2d(kernel_size=(2, 2), stride=(2,2))

```
class CNNBlock(nn.Module):
    def __init__(self, in_channels, out_channels, **kwargs):
        super(CNNBlock, self).__init__()
        self.conv = nn.Conv2d(in_channels, out_channels, bias=False, **kwargs)
        self.batchnorm = nn.BatchNorm2d(out_channels)
        self.leakyrelu = nn.LeakyReLU(0.1)

    def forward(self, x):
        return self.leakyrelu(self.batchnorm(self.conv(x)))
```

Aladdin Persson, 2020, Machine Learning Collection, https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/object_detection/YOLO

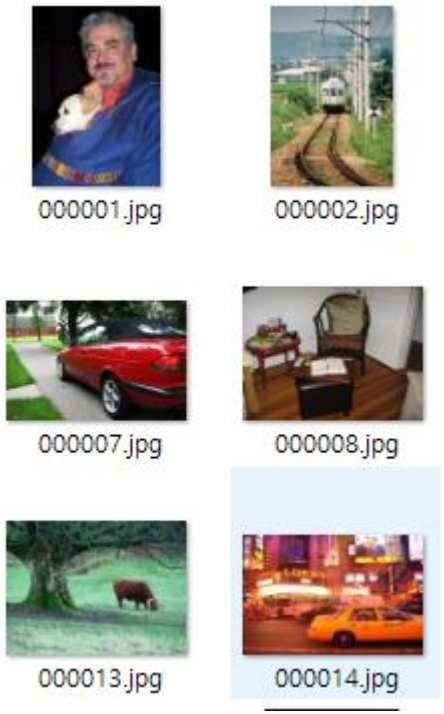
Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO - code

Implementing YOLOV1 from scratch using PyTorch

Fast campus

VOC Dataset & Label



	A	B
1	image	text
2	000007.jpg	000007.txt
3	000026.jpg	000026.txt
4	000032.jpg	000032.txt
5	000033.jpg	000033.txt
6	000034.jpg	000034.txt
7	000035.jpg	000035.txt
8	000036.jpg	000036.txt
9	000042.jpg	000042.txt
10	000089.jpg	000089.txt
11	000091.jpg	000091.txt
12	000104.jpg	000104.txt
13	000112.jpg	000112.txt
14	000122.jpg	000122.txt
15	000129.jpg	000129.txt
16	000133.jpg	000133.txt
17	000134.jpg	000134.txt
18	000187.jpg	000187.txt
102	000620.jpg	000620.txt
103	000622.jpg	000622.txt
104	000625.jpg	000625.txt

DbInputFormCBCMod.html × 000001.txt ×

```
11 0.34135977337110485 0.609 0.4164305949008499 0.262
14 0.5070821529745043 0.508 0.9745042492917847 0.972
```

```
class VOCDataset(torch.utils.data.Dataset):

    def __getitem__(self, index):
        label_path = os.path.join(self.label_dir, self.annotations.iloc[index, 1])
        with open(label_path) as f:
            for label in f.readlines():
                class_label, x, y, width, height = [...]
                boxes.append([class_label, x, y, width, height])

        image = Image.open(img_path)
        boxes = torch.tensor(boxes)

        image, boxes = self.transform(image, boxes) # box = class, cx, cy, w, h
                        # Compose([transforms.Resize((448, 448)), transforms.ToTensor(),])

        label_matrix = torch.zeros((self.S, self.S, self.C + 5 * self.B))
        for box in boxes:
            class_label, x, y, width, height = box.tolist()

            i, j = int(self.S * y), int(self.S * x)
            x_cell, y_cell = self.S * x - j, self.S * y - i
            ...

            box_coordinates = torch.tensor([x_cell, y_cell, width_cell, height_cell])

            label_matrix[i, j, 21:25] = box_coordinates
            label_matrix[i, j, class_label] = 1  #one hot encoding
```

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO - code

Loss function

loss function:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

```
for batch_idx, (x, y) in enumerate(loop):
    x, y = x.to(DEVICE), y.to(DEVICE)
    out = model(x)
    loss = loss_fn(out, y)
    mean_loss.append(loss.item())
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

```
class YoloLoss(nn.Module):  (16,7,7,30)
    def forward(self, predictions, target):  # 예측값과 라벨값 입력
        predictions = predictions.reshape(-1, self.S, self.S, self.C + self.B * 5)  (16,7,7,30)
        # IoU 계산
        iou_b1 = intersection_over_union(predictions[..., 21:25], target[..., 21:25])  (16,7,7,1) = (16,7,7,4), (16,7,7,4)
        iou_b2 = intersection_over_union(predictions[..., 26:30], target[..., 21:25])
        ious = torch.cat([iou_b1.unsqueeze(0), iou_b2.unsqueeze(0)], dim=0)  (2,16,7,7,1) = (1,16,7,7,1), (1,16,7,7,1)
        iou_maxes, bestbox = torch.max(ious, dim=0) # max iou box, the box index. bestbox[] == 1 or 0
        exists_box = target[..., 20].unsqueeze(3)  # in paper this is Iobj_i  (16,7,7,1), (16,7,7,1)
                (16,7,7,1)
        box_predictions = exists_box * ((bestbox * predictions[..., 26:30] + (1 - bestbox) * predictions[..., 21:25]))
        box_targets = exists_box * target[..., 21:25]  (16,7,7,4)
                (16,7,7,4)
        # Loss계산. bbox W, H loss = MSE(pred box, target box).
        box_predictions[..., 2:4] = torch.sign(box_predictions[..., 2:4]) * torch.sqrt(
            torch.abs(box_predictions[..., 2:4] + 1e-6)
        )
        box_targets[..., 2:4] = torch.sqrt(box_targets[..., 2:4])

        box_loss = self.mse(torch.flatten(box_predictions, end_dim=-2), torch.flatten(box_targets, end_dim=-2),)
        (16,7,7,1)                                  (16,7,7,4)                                  (16,7,7,4)
        pred_box = (bestbox * predictions[..., 25:26] + (1 - bestbox) * predictions[..., 20:21])
        object_loss = self.mse(torch.flatten(exists_box * pred_box), torch.flatten(exists_box * target[..., 20:21]),)

        no_object_loss = self.mse(torch.flatten((1 - exists_box) * predictions[..., 20:21], start_dim=1),
            torch.flatten((1 - exists_box) * target[..., 20:21], start_dim=1),)
        no_object_loss += self.mse(torch.flatten((1 - exists_box) * predictions[..., 25:26], start_dim=1),
            torch.flatten((1 - exists_box) * target[..., 20:21], start_dim=1))

        class_loss = self.mse(torch.flatten(exists_box * predictions[..., :20], end_dim=-2,),
            torch.flatten(exists_box * target[..., :20], end_dim=-2,),)

        loss = (self.lambda_coord * box_loss
            + object_loss
            + self.lambda_noobj * no_object_loss
            + class_loss
        )
```

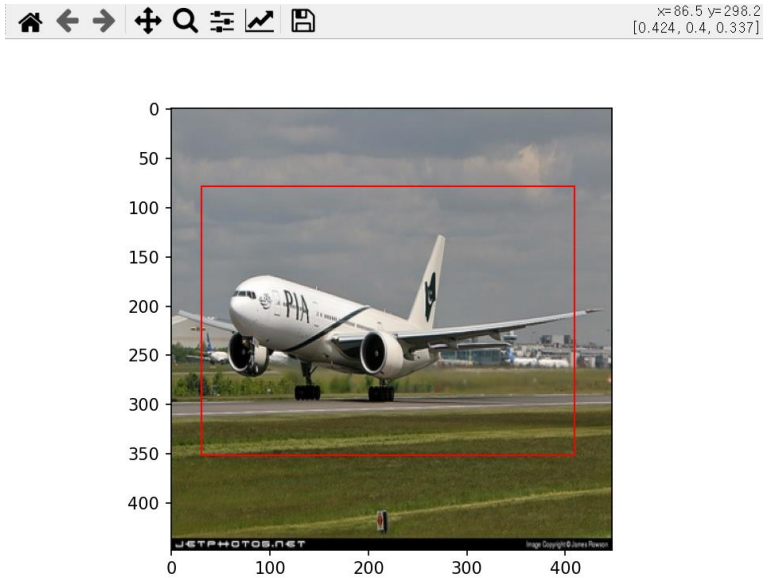
Aladdin Persson, 2020, Machine Learning Collection, https://github.com/aladdinpersson/Machine-Learning-Collection/tree/master/ML/Pytorch/object_detection/YOLO

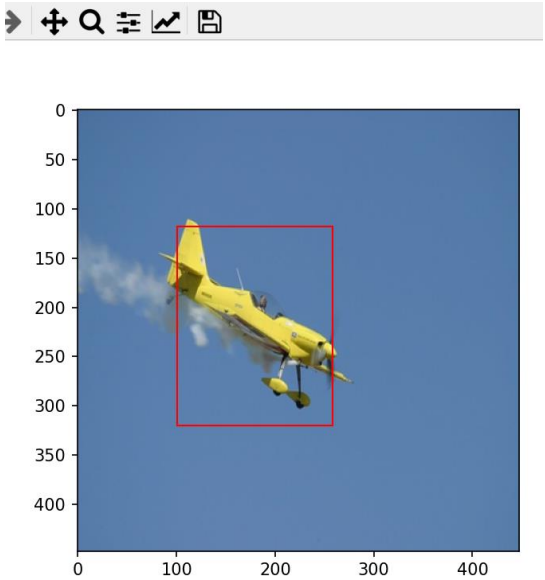
Copyright FAST CAMPUS Corp. All Rights Reserved

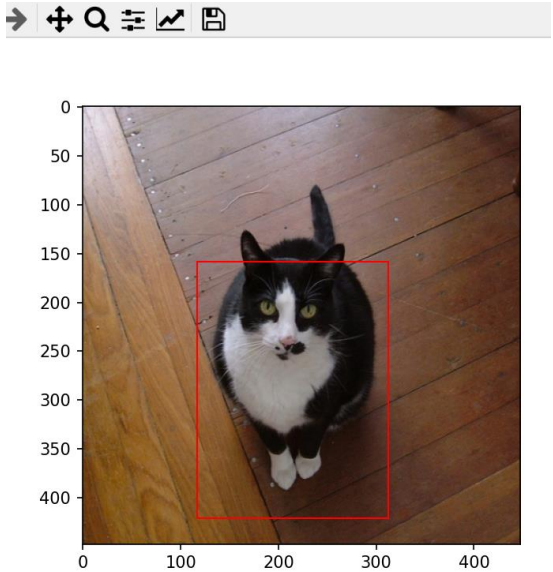
YOLO - code

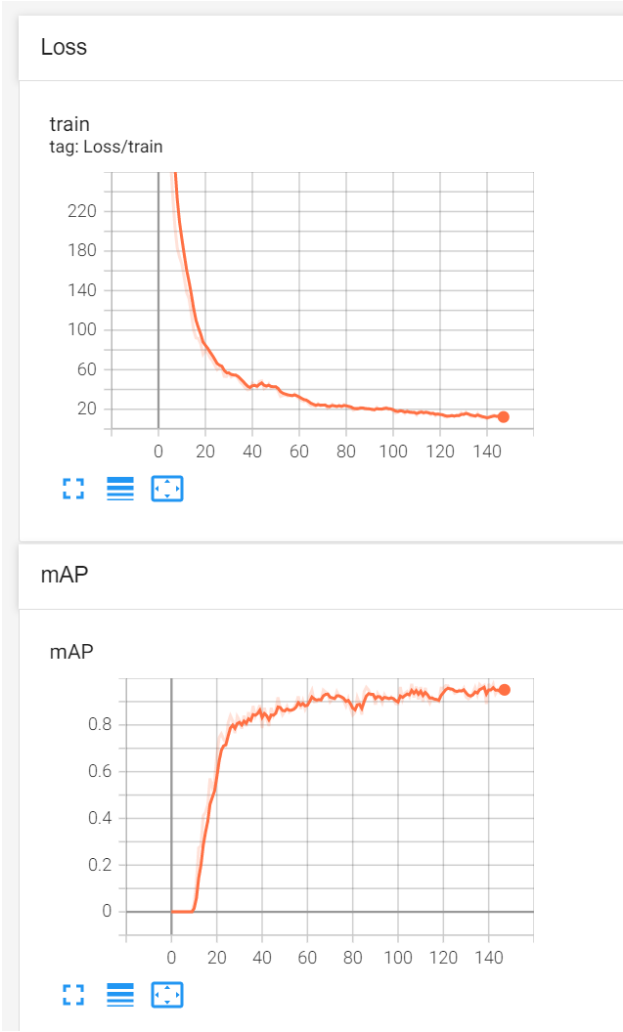
Implementing YOLOV1 from scratch using PyTorch

Loss function







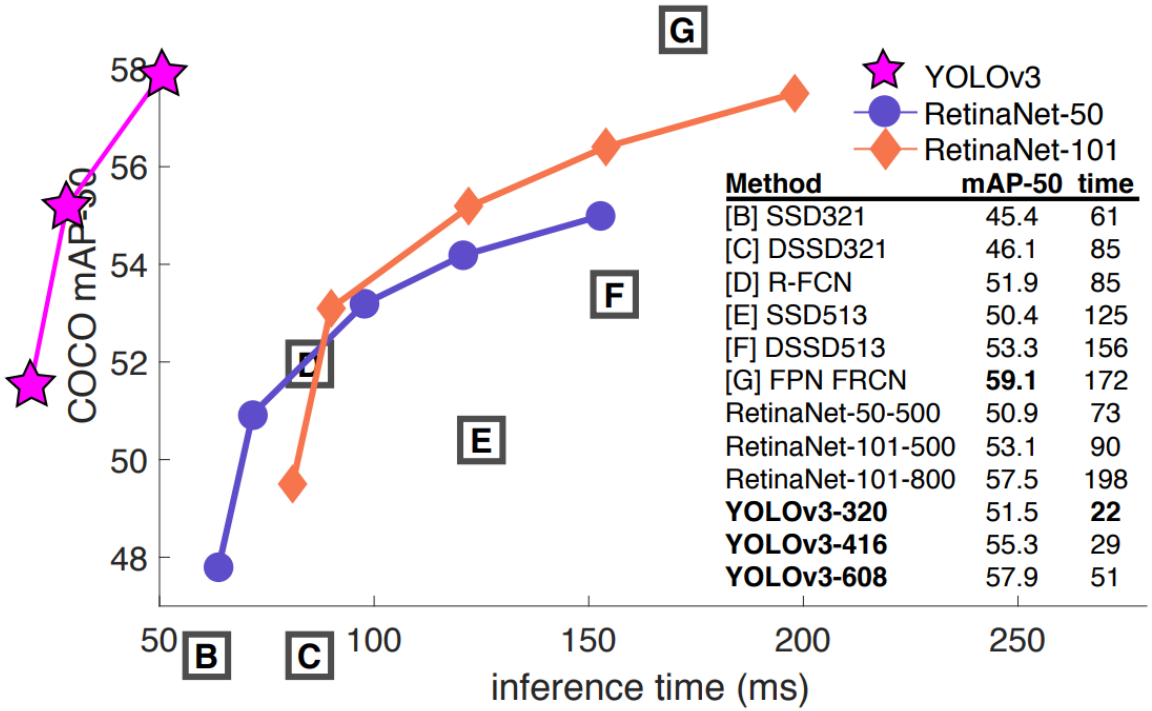


Copyright FAST CAMPUS Corp. All Rights Reserved

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

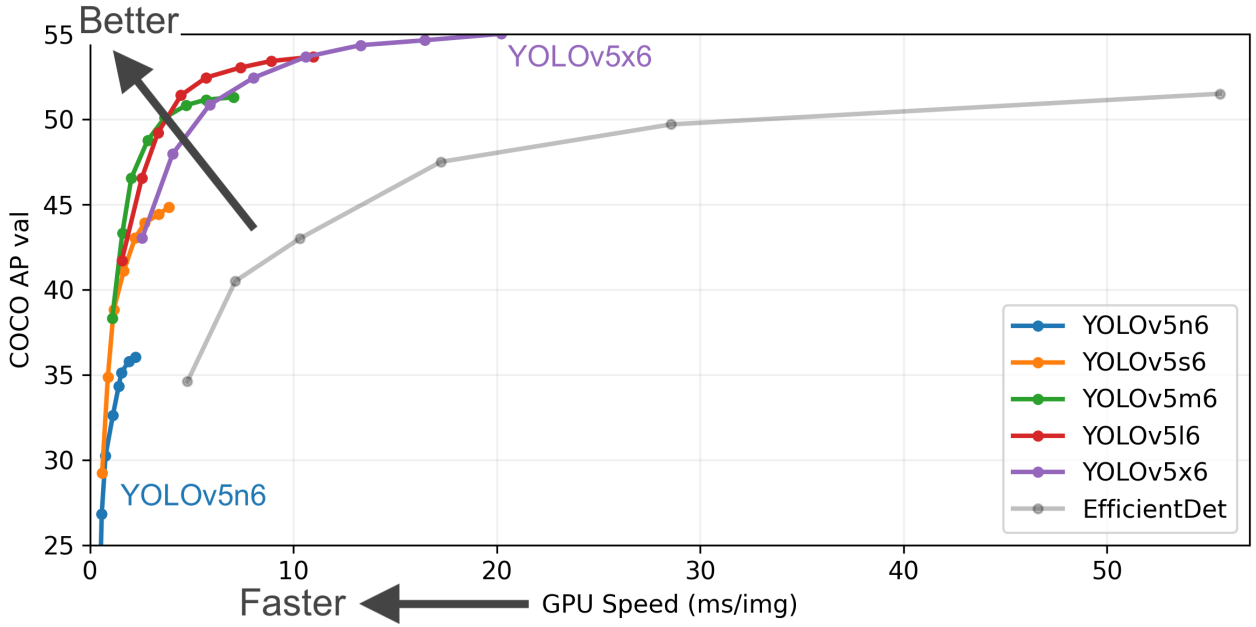
YOLO

YOLO v3, v5



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Figure 3. Again adapted from the [9], this time displaying speed/accuracy tradeoff on the mAP at .5 IOU metric. You can tell YOLOv3 is good because it's very high and far to the left. Can you cite your own paper? Guess who's going to try, this guy → [16]. Oh, I forgot, we also fix a data loading bug in YOLOv2, that helped by like 2 mAP. Just sneaking this in here to not throw off layout.



Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO3 – concept

YOLOv3: An Incremental Improvement

Joseph Redmon, Ali Farhadi
University of Washington

Abstract

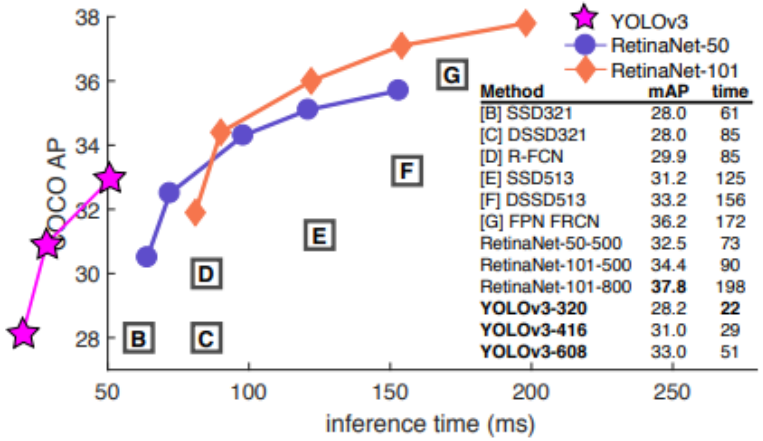
We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP_{50} in 51 ms on a Titan X, compared to 57.5 AP_{50} in 198 ms by RetinaNet, similar performance but 3.8× faster. As always, all the code is online at https://pjreddie.com/yolo/.

1. Introduction

Sometimes you just kinda phone it in for a year, you know? I didn't do a whole lot of research this year. Spent a lot of time on Twitter. Played around with GANs a little. I had a little momentum left over from last year [12] [1]; I managed to make some improvements to YOLO. But, honestly, nothing like super interesting, just a bunch of small changes that make it better. I also helped out with other people's research a little.

Actually, that's what brings us here today. We have a camera-ready deadline [4] and we need to cite some of the random updates I made to YOLO but we don't have a source. So get ready for a TECH REPORT!

The great thing about tech reports is that they don't need intros, y'all know why we're here. So the end of this intro-



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 1. We adapt this figure from the Focal Loss paper [9]. YOLOv3 runs significantly faster than other detection methods with comparable performance. Times from either an M40 or Titan X, they are basically the same GPU.

2.1. Bounding Box Prediction

Following YOLO9000 our system predicts bounding boxes using dimension clusters as anchor boxes [15]. The network predicts 4 coordinates for each bounding box, t_x, t_y, t_w, t_h. If the cell is offset from the top left corner of the image by (c_x, c_y) and the bounding box prior has width and height p_w, p_h, then the predictions correspond to:

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

Joseph Redmon, Ali Farhadi, 2018, YOLOv3: An Incremental Improvement, https://pjreddie.com/media/files/papers/YOLOv3.pdf

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO3 – concept

Backbone model

YOLOv3 predicts boxes at 3 different scales. Our system extracts features from those scales using a similar concept to feature pyramid networks [8]. From our base feature extractor we add several convolutional layers. The last of these predicts a 3-d tensor encoding bounding box, objectness, and class predictions. In our experiments with COCO [10] we predict 3 boxes at each scale so the tensor is $N \times N \times [3 * (4 + 1 + 80)]$ for the 4 bounding box offsets, 1 objectness prediction, and 80 class predictions.

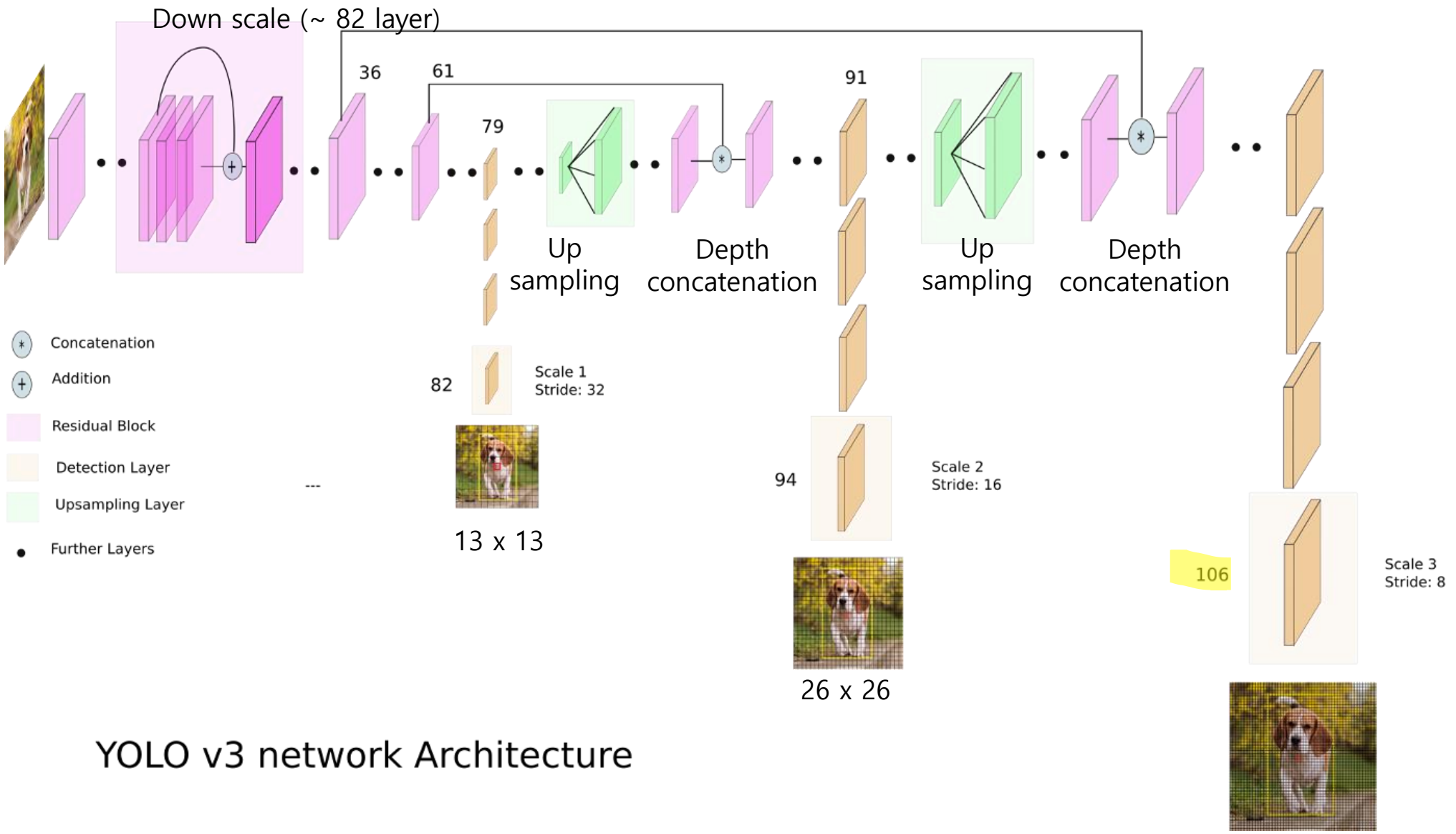
network stuff. Our network uses successive 3×3 and 1×1 convolutional layers but now has some shortcut connections as well and is significantly larger. It has 53 convolutional layers so we call it.... wait for it..... Darknet-53!

	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.**

erved

YOLO v3 – concept



YOLO v3 network Architecture

Ayoosh Kathuria, 2018, What's new in YOLO v3, Towards Data Science
https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b

github.com/pjreddi

```
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=leaky
```

https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO v3 – concept

Grid

Divide S x S grid
objectness P0 = Pr(object) x IoU(predict, ground)
classification with sigmoid function for multi-labels

Output = S x S x (5B + K)
K = class confidence

Anchor box & Multi-scale

Multi-scale = 32, 16, 8 strides.
ex) 416 x 416 image = 13 x 13, 26 x 26, 52 x 52

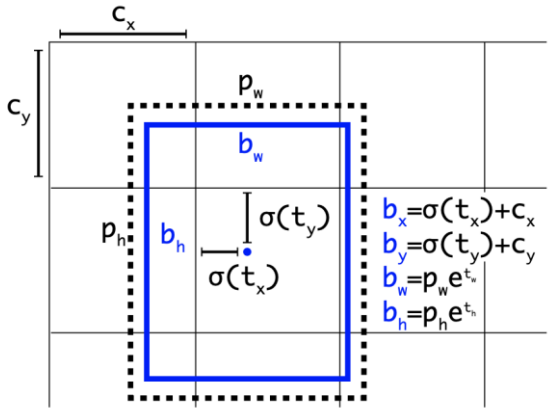
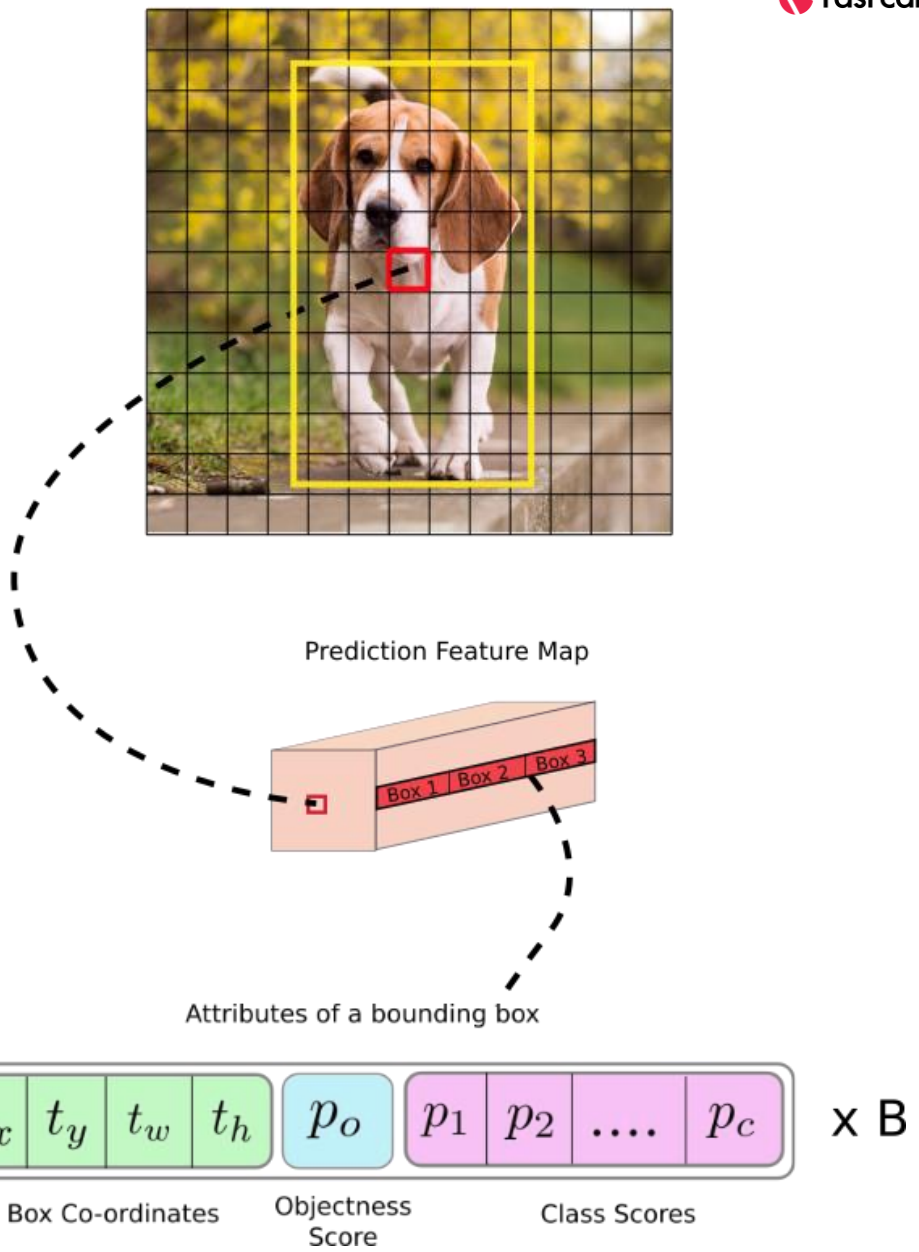


Image Grid. The Red Grid is responsible for detecting the dog

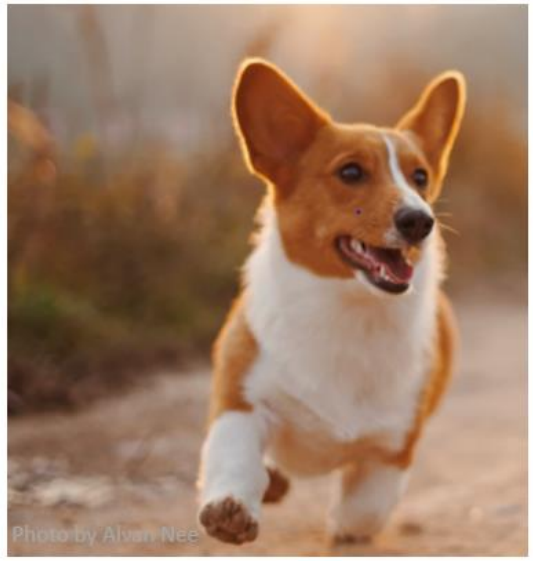


Ayoosh Kathuria, 2018, paperspace, How to implement a YOLO (v3) object detector from scratch in PyTorch, https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

Copyright FAST CAMPUS Corp. All Rights Reserved

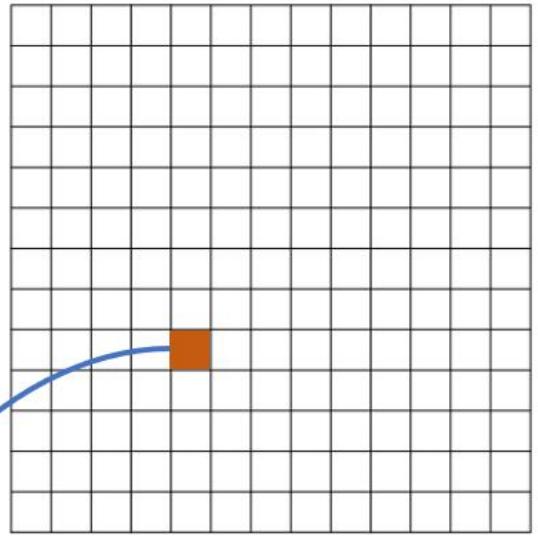
YOLO3 – concept

3 Anchor boxes



Pre-processing Image
(416, 416, 3)

YOLO v3
Network



Objectness
score

Encoding*
(13, 13, **3**, (4+1+80))**

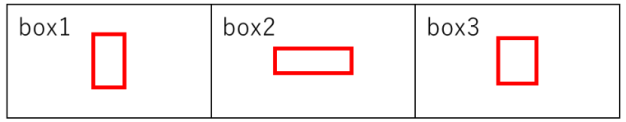
3 anchor boxes per
one grid cell***

Bounding box
offset: x, y, w, h

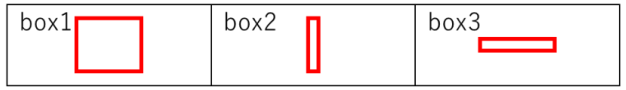
Class
probabilities

* In YOLO v3, the final output of detectors will be in shape 13x13, 26x26, 52x52. The example above is one of the output in 13x13 for illustration purposes only
** (13, 13, 3x(4+1+80)) are flatten into (13, 13, 255)
*** In YOLO v3, there are 3 anchor boxes per grid cell

scale1 : Detection Result



scale2 : Detection Result



scale3 : Detection Result

$$3 \times (4 + 1 + class_n)$$

Natashia Natashia, 2020, YOLO v3 Object Detection with Keras,
https://towardsdatascience.com/yolo-v3-object-detection-with-keras-461d2cfccef6

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO v3 – concept

loss function:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Localization loss

$$\lambda_{coord} * \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(x_i - x_i' \right)^2 + \left(y_i - y_i' \right)^2 \right]$$

$$+ \lambda_{coord} * \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(w_i - w_i' \right)^2 + \left(h_i - h_i' \right)^2 \right]$$

Confidence loss

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} * BCE\left(C_i, C_i' \right)$$

$$+ \lambda_{noobj} * \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} * BCE\left(C_i, C_i' \right) * \left(1 - \mathbb{1}_{ij}^{obj} \right)$$

Classification loss

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{ij}^{obj} \sum_{c \in classes} BCE\left(p_i(c), p_i'(c) \right)$$

Binary cross entropy

Vignesh, 2020, Using YOLOv3 for detection of traffic signs, medium.com

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO v3 – one shot object detection implementation

Fast campus

YOLO3 keras.ipynb

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유

+ 코드 + 텍스트

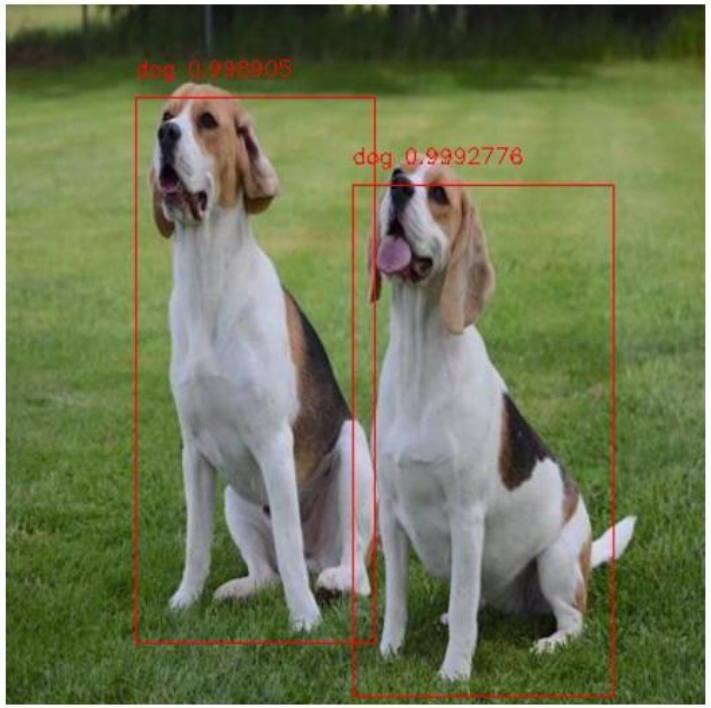
RAM 디스크 수정 가능

```
loading weights of convolution #57
```

[32]
```
!ls
```

```
Beagle_Dogs.jpg  input_detected.jpg  input.jpg  sample_data  yolov3.weights
```

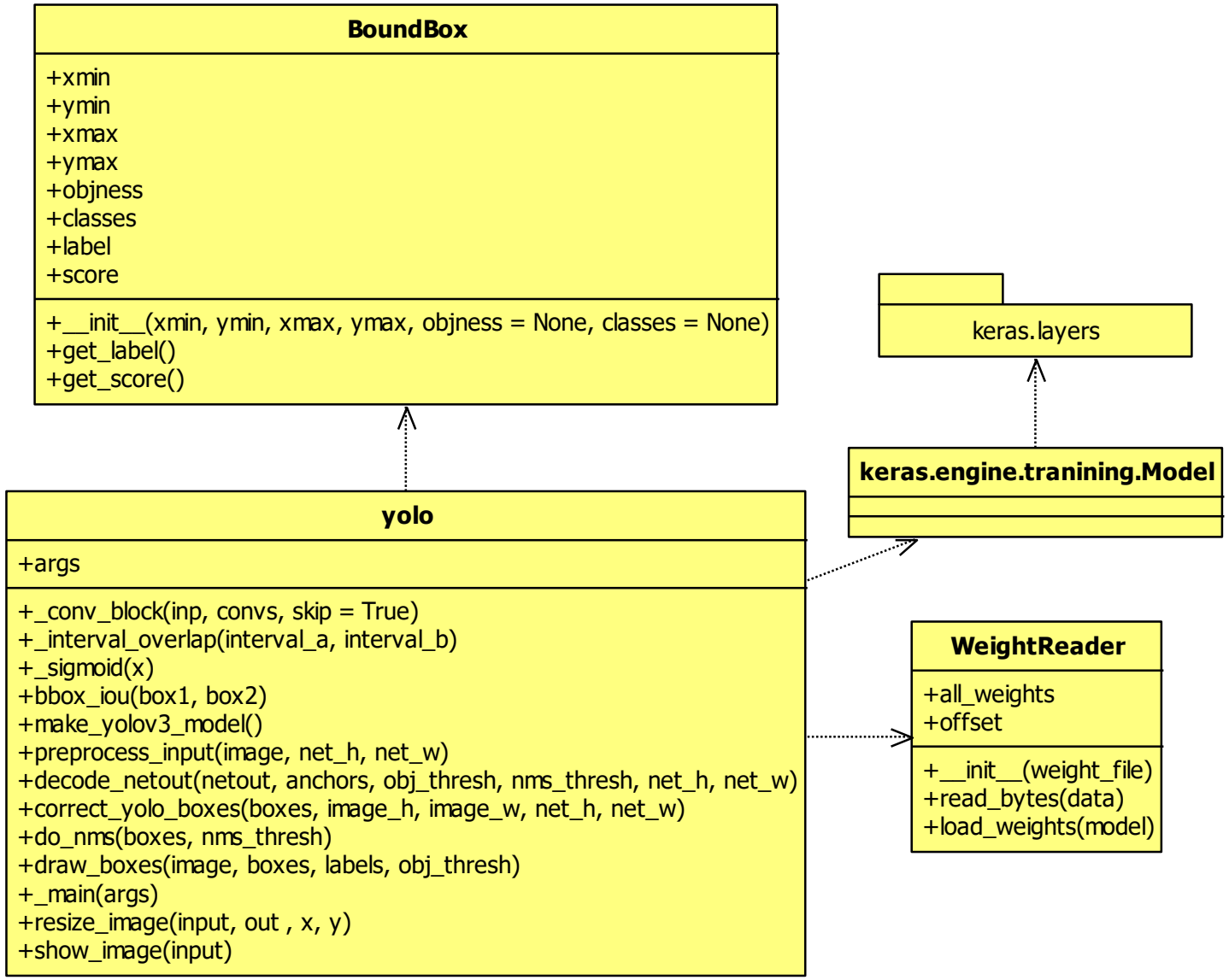
```
from IPython.display import Image, display
display(Image('input_detected.jpg'))
```



https://github.com/experiencor/keras-yolo3

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO v3 - code review. one shot object detection



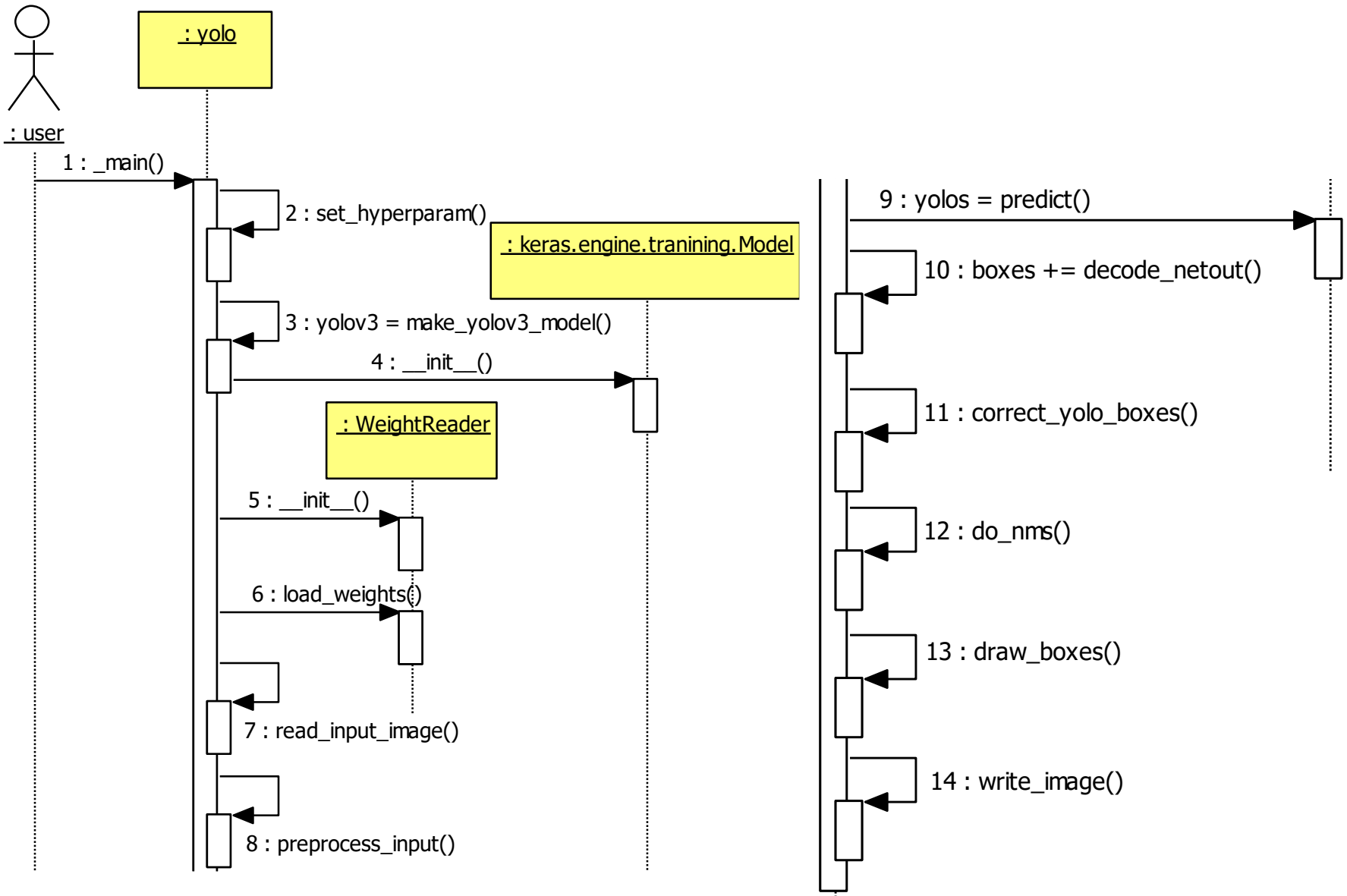
UML distilled

YOLO3 (Detection, Training, and Evaluation), https://github.com/experiencor/keras-yolo3

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO v3 – code review

```python
def _main_(args):
    weights_path = args.weights
    image_path   = args.image

    net_h, net_w = 416, 416
    obj_thresh, nms_thresh = 0.5, 0.45
    anchors =
[[116,90,  156,198,  373,326],  [30,61,
62,45,  59,119], [10,13,  16,30,  33,23]]
    labels = ["person", "bicycle", "car",
"motorbike", "aeroplane", "bus", "train", "truck”]

    yolov3 = make_yolov3_model()

    weight_reader = WeightReader(weights_path)
    weight_reader.load_weights(yolov3)

    image = cv2.imread(image_path)
    image_h, image_w, _ = image.shape
    new_image = preprocess_input(image, net_h,
net_w)

    # run the prediction
    yolos = yolov3.predict(new_image)
    boxes = []

    for i in range(len(yolos)):
        # decode the output of the network
        boxes += decode_netout(yolos[i][0],
anchors[i], obj_thresh, nms_thresh, net_h, net_w)

    # correct the sizes of the bounding boxes
    correct_yolo_boxes(boxes, image_h, image_w,
net_h, net_w)

    # suppress non-maximal boxes
    do_nms(boxes, nms_thresh)

    # draw bounding boxes on the image using labels
    draw_boxes(image, boxes, labels, obj_thresh)

    # write the image with bounding boxes to file
    cv2.imwrite(image_path[:-4] + '_detected' +
image_path[-4:], (image).astype('uint8'))
```



Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO v3 – code review

```python
def make_yolov3_model():
    input_image = Input(shape=(None, None, 3))
    # Layer  0 => 4
    x = _conv_block(input_image, [{'filter': 32, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 0},
                                  {'filter': 64, 'kernel': 3, 'stride': 2, 'bnorm': True, 'leaky': True, 'layer_idx': 1},
                                  {'filter': 32, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 2},
                                  {'filter': 64, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 3}])
    # Layer  5 => 8
    x = _conv_block(x, [{'filter': 128, 'kernel': 3, 'stride': 2, 'bnorm': True, 'leaky': True, 'layer_idx': 5},
                        {'filter':  64, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 6},
                        {'filter': 128, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 7}])
    x = _conv_block(x, [{'filter':  64, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 9},
                        {'filter': 128, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 10}])
    x = _conv_block(x, [{'filter': 256, 'kernel': 3, 'stride': 2, 'bnorm': True, 'leaky': True, 'layer_idx': 12},
                        {'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 13},
                        {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 14}])
    for i in range(7):
        x = _conv_block(x, [{'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 16+i*3},
                            {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 17+i*3}])
    skip_36 = x
    x = _conv_block(x, [{'filter': 512, 'kernel': 3, 'stride': 2, 'bnorm': True, 'leaky': True, 'layer_idx': 37},
                        {'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 38},
                        {'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 39}])
    for i in range(7):
        x = _conv_block(x, [{'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 41+i*3},
                            {'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 42+i*3}])
    skip_61 = x
    x = _conv_block(x, [{'filter': 1024, 'kernel': 3, 'stride': 2, 'bnorm': True, 'leaky': True, 'layer_idx': 62},
                        {'filter':  512, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 63},
                        {'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 64}])
    for i in range(3):
        x = _conv_block(x, [{'filter':  512, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 66+i*3},
                            {'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 67+i*3}])
    x = _conv_block(x, [{'filter':  512, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 75},
                        {'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 76},
                        {'filter':  512, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 77},
                        {'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 78},
                        {'filter':  512, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 79}], skip=False)
    yolo_82 = _conv_block(x, [{'filter': 1024, 'kernel': 3, 'stride': 1, 'bnorm': True,  'leaky': True,  'layer_idx': 80},
                              {'filter':  255, 'kernel': 1, 'stride': 1, 'bnorm': False, 'leaky': False, 'layer_idx': 81}], skip=False)
    x = _conv_block(x, [{'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 84}], skip=False)
    x = UpSampling2D(2)(x)
    x = concatenate([x, skip_61])
    x = _conv_block(x, [{'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 87},
                        {'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 88},
                        {'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 89},
                        {'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 90},
                        {'filter': 256, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True, 'layer_idx': 91}], skip=False)
    yolo_94 = _conv_block(x, [{'filter': 512, 'kernel': 3, 'stride': 1, 'bnorm': True,  'leaky': True,  'layer_idx': 92},
                              {'filter': 255, 'kernel': 1, 'stride': 1, 'bnorm': False, 'leaky': False, 'layer_idx': 93}], skip=False)
    x = _conv_block(x, [{'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True, 'leaky': True,   'layer_idx': 96}], skip=False)
    x = UpSampling2D(2)(x)
    x = concatenate([x, skip_36])
    # Layer 99 => 106
    yolo_106 = _conv_block(x, [{'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True,  'leaky': True,  'layer_idx': 99},
                               {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True,  'leaky': True,  'layer_idx': 100},
                               {'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True,  'leaky': True,  'layer_idx': 101},
                               {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True,  'leaky': True,  'layer_idx': 102},
                               {'filter': 128, 'kernel': 1, 'stride': 1, 'bnorm': True,  'leaky': True,  'layer_idx': 103},
                               {'filter': 256, 'kernel': 3, 'stride': 1, 'bnorm': True,  'leaky': True,  'layer_idx': 104},
                               {'filter': 255, 'kernel': 1, 'stride': 1, 'bnorm': False, 'leaky': False, 'layer_idx': 105}], skip=False)
    model = Model(input_image, [yolo_82, yolo_94, yolo_106])
    return model
```

	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

https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg

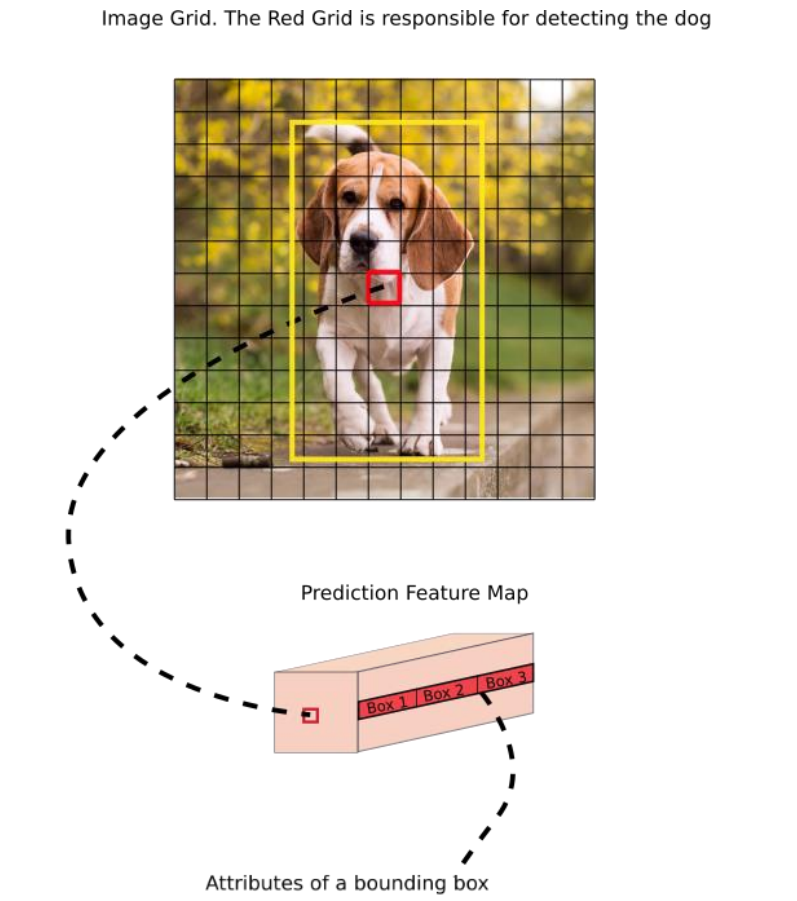
Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO v3 – code review

```python
def decode_netout(netout, anchors, obj_thresh, nms_thresh, net_h, net_w):
    grid_h, grid_w = netout.shape[:2] # netout은 (13,13,255)텐서크기임. grid_h, grid_w는 격자갯수 13임(1st scale).
    nb_box = 3
    netout = netout.reshape((grid_h, grid_w, nb_box, -1)) # (격자 H,격자 W,앵커박스, …)인 (13,13,3,85)텐서차원 변경
    nb_class = netout.shape[-1] - 5 # 마지막 텐서 차원에서 -5(bbox와 objectness)하면 클래스갯수 80임.

    boxes = []
    netout[..., :2]  = _sigmoid(netout[..., :2]) # bbox x,y 텐서 slicing(13,13,3,2). x,y는 loss()전 sig()로 0-1변환됨.
    netout[..., 4:]  = _sigmoid(netout[..., 4:]) # objectness, classes 텐서 slicing (13,13,3,81)
    # objectness 텐서를 classes 텐서(80)과 곱함. objectness가 0이면 classes는 0이 됨.
    netout[..., 5:]  = netout[..., 4][..., np.newaxis] * netout[..., 5:]
    netout[..., 5:] *= netout[..., 5:] > obj_thresh # objectness threshold 이상일 경우에만, 예측된 classes 유효함.

    # [box][x,y,w,h,objectness,classes] 텐서임. Bbox별로 열거해 처리함.
    for i in range(grid_h*grid_w):
        row = i / grid_w    # grid_w=13. 격자 별 인덱스 계산
        col = i % grid_w

        for b in range(nb_box): # 앵커박스 nb_box = 3 인 경우 3번 루프.
            # 앵커박스의 objectness 획득
            objectness = netout[int(row)][int(col)][b][4]
            # objectness가 objectness threshhold값 이하면 아래 로직 처리 안함.
            if(objectness.all() <= obj_thresh): continue  #
https://numpy.org/doc/stable/reference/generated/numpy.all.html

            # 앵커박스의 텐서 0-3까지는 bbox x, y, w, h임. x, y는 각 격자의 상대좌표값임.
            x, y, w, h = netout[int(row)][int(col)][b][:4]

            x = (col + x) / grid_w # 격자 13x13 좌표계 기준으로 x, y값을 계산해 정규화(0-1)
            y = (row + y) / grid_h
            w = anchors[2 * b + 0] * np.exp(w) / net_w # w를 앵커 스케일 팩터와 곱하고 416 픽셀 해상도 기준 정규화(0-1)
            h = anchors[2 * b + 1] * np.exp(h) / net_h # h를 위와 동일하게 계산

            # 클래스 확률(80개) 획득
            classes = netout[int(row)][col][b][5:]

            # x, y는 0-1로 정규화된 격자 중심좌표임. w/2, h/2를 계산해, xmin, ymin, xmax, ymax를 얻음.
            box = BoundBox(x-w/2, y-h/2, x+w/2, y+h/2, objectness, classes)
            boxes.append(box)

    return boxes
```

* In YOLO v3, the final output of detectors will be in shape 13x13, 26x26, 52x52. The example above is one of the output in 13x13 for illustration purposes only

** (13, 13, 3x(4+1+80)) are flatten into (13, 13, 255)

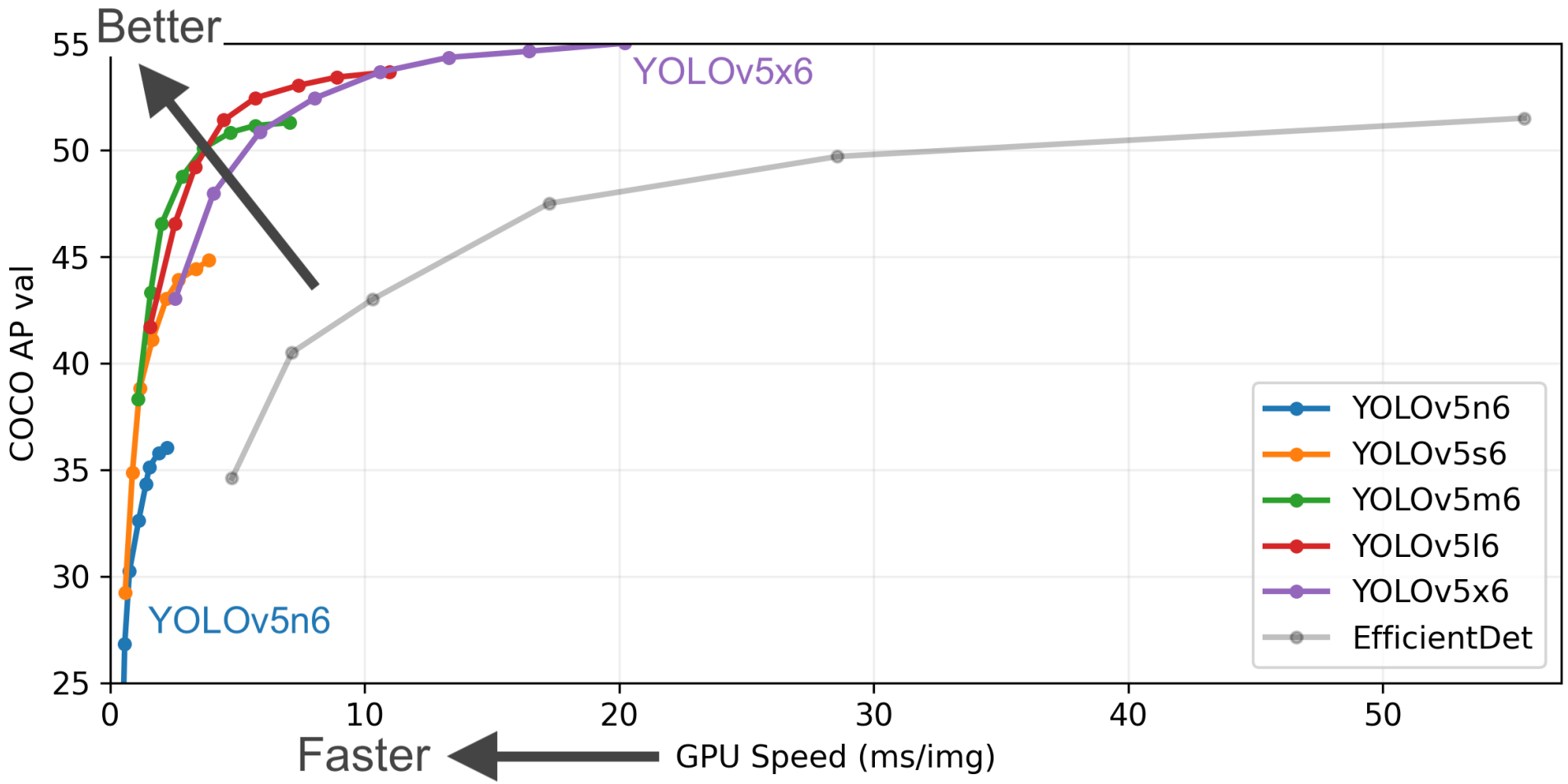


Copyright FAST CAMPUS Corp. All Rights Reserved

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

YOLO in practice

YOLO v5 – History



Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 with CoLab

https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb

https://ultralytics.com/about https://github.com/ultralytics/yolov5

YOLOv5 Tutorial

파일 수정 보기 삽입 런타임 도구 도움말 변경사항을 저장할 수 없음

공유

+ 코드 + 텍스트 Drive로 복사 RAM 디스크 수정 가능

YOLOv5 by

Download on the App Store

Coming Soon on Google Play

This is the **official YOLOv5** 🚀 **notebook** by **Ultralytics**, and is freely available for re under the GPL-3.0 license. For more information please visit https://github.com/ultralytics/yolov5 and https://ultralytics.com. Thank you!

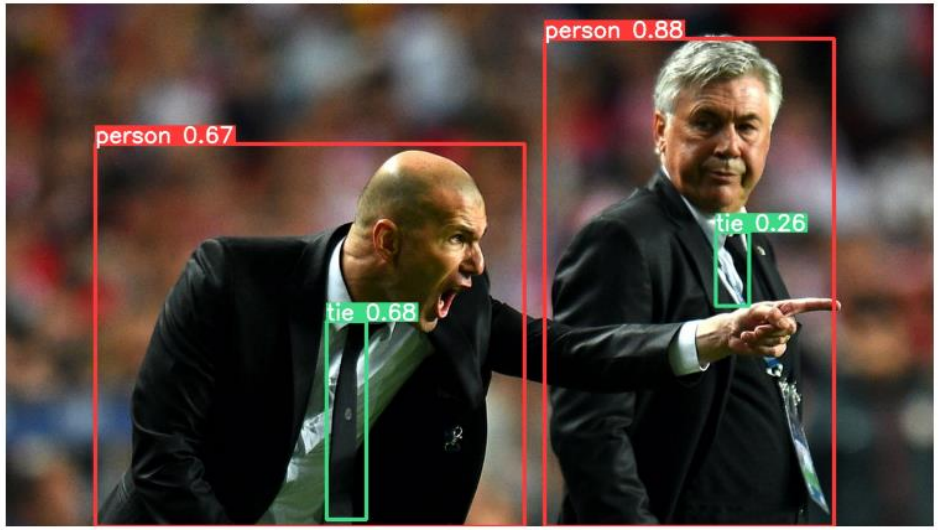
About Us

YOLO5 with CoLab

+ 코드 + 텍스트 Drive로 복사 RAM 디스크 수정 가능

```
!python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source data/images
display.Image(filename='runs/detect/exp/zidane.jpg', width=600)
```

```
detect: weights=['yolov5s.pt'], source=data/images, data=data/coco128.yaml, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, device=, view
YOLOv5 🚀 v6.1-401-g3456fe6 Python-3.7.13 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

Fusing layers...
YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients
image 1/2 /content/yolov5/data/images/bus.jpg: 640x480 4 persons, 1 bus, Done. (0.012s)
image 2/2 /content/yolov5/data/images/zidane.jpg: 384x640 2 persons, 2 ties, Done. (0.013s)
Speed: 0.5ms pre-process, 12.8ms inference, 1.0ms NMS per image at shape (1, 3, 640, 640)
Results saved to runs/detect/exp3
```



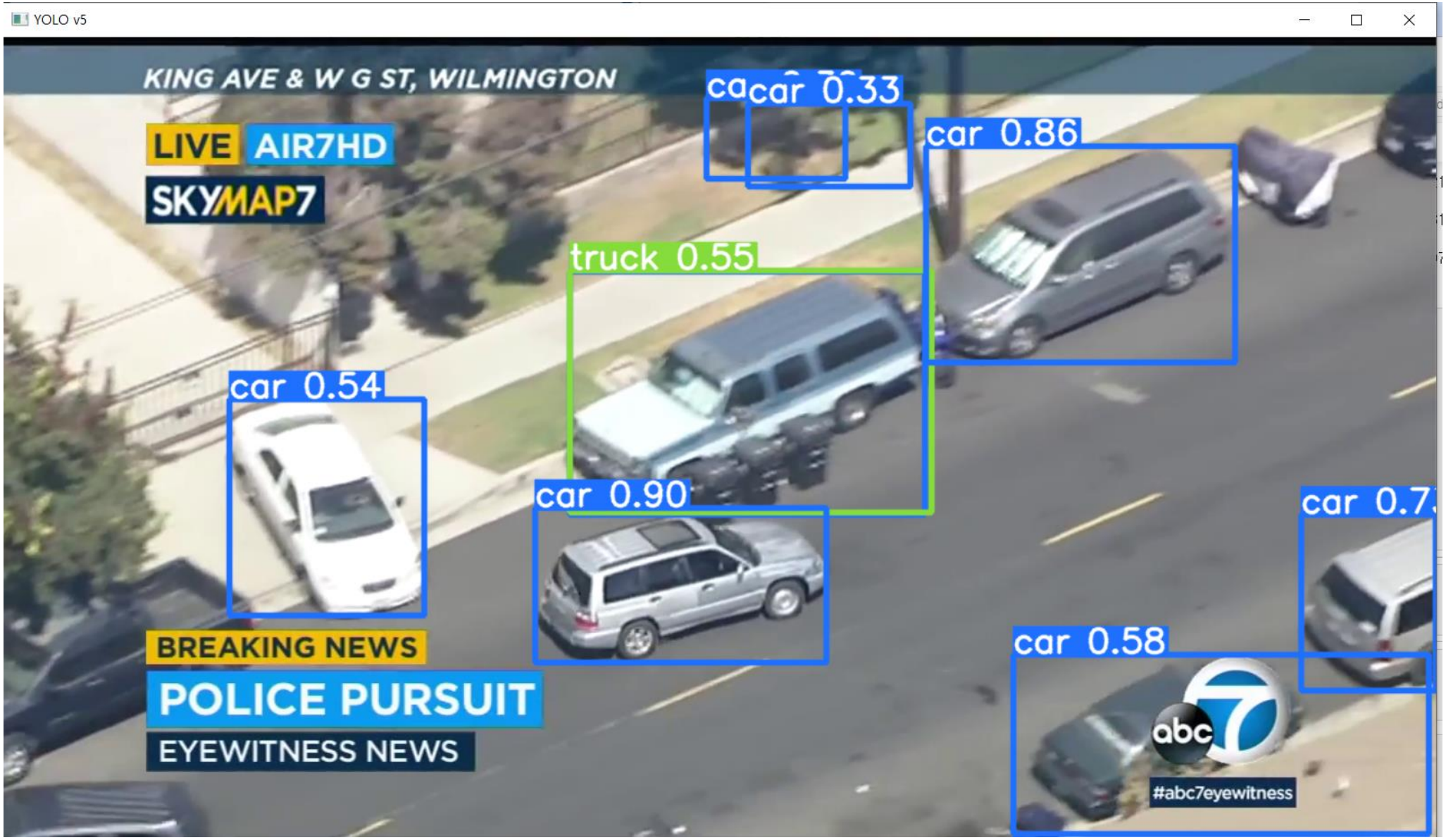
Webcam on CoLab

```
cap = cv2.VideoCapture(0)
while cap.isOpened():
  ret, frame = cap.read()
  cv2.imshow('YOLO', frame)
  if cv2.waitKey(10) && 0xFF == crd('q'):
    break
cap.release()
cv2.destoryAllWindows()
```

Example. Realtime Object Inference on Raspberry Pi 4 Webcam

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 with CoLab



Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 with CoLab

Fast campus

Manual - https://docs.ultralytics.com/

docs.ultralytics.com

YOLOv5 Documentation | Search | ultralytics/yolov5 v6.2 31.7k

YOLOv5 Documentation
- Introduction
- Quick Start
- Environments
- Tutorials
- FAQ

Introduction

To get started right now check out the Quick Start Guide

What is YOLOv5

YOLO an acronym for 'You only look once', is an object detection algorithm that divides images into a grid system. Each cell in the grid is responsible for detecting objects within itself.

YOLO is one of the most famous object detection algorithms due to its speed and accuracy.

The History of YOLO

YOLOv5

Shortly after the release of YOLOv4 Glenn Jocher introduced YOLOv5 using the Pytorch framework.
The open source code is available on GitHub

Author: Glenn Jocher
Released: 18 May 2020

Table of contents
- What is YOLOv5
- The History of YOLO
  - YOLOv5
  - YOLOv4
  - YOLOv3
  - YOLOv2
  - YOLOv1

Download dataset from torch hub

Download COCO val 2017 dataset (1GB - 5000 images), and test model accuracy.

[5]
```
# Download COCO val
torch.hub.download_url_to_file('https://ultralytics.com/assets/coco2017val.zip', 'tmp.zip')
!unzip -q tmp.zip -d ../datasets && rm tmp.zip
```

100% 780M/780M [00:38<00:00, 15.7MB/s]

```
# Run YOLOv5x on COCO val
!python val.py --weights yolov5x.pt --data coco.yaml --img 640 --iou 0.65 --half
```

```
val: data=/content/yolov5/data/coco.yaml, weights=['yolov5x.pt'], batch_size=32, imgsz=640, conf_thres=0.001, iou_thres=0.65, task=val, device=, workers=8
YOLOv5 🚀 v6.1-401-g3456fe6 Python-3.7.13 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

Downloading https://github.com/ultralytics/yolov5/releases/download/v6.1/yolov5x.pt to yolov5x.pt...
100% 166M/166M [00:06<00:00, 27.5MB/s]

Fusing layers...
YOLOv5x summary: 444 layers, 86705005 parameters, 0 gradients
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
100% 755k/755k [00:00<00:00, 17.6MB/s]
val: Scanning '/content/datasets/coco/val2017' images and labels...4952 found, 48 missing, 0 empty, 0 corrupt: 100% 5000/5000 [00:02<00:00, 2039.19it/s]
val: New cache created: /content/datasets/coco/val2017.cache
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95:  65% 102/157 [01:19<00:43,  1.26it/s]
```

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 in practice using Jupyter Notebook, vscode

Fast campus

detection

python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source 0

```
$ python detect.py --source OPTION
```

Replace OPTION with your selection, to detect from:

- **Webcam** : (OPTION = 0) *For live object detection from your connected webcam*
- **Image** : (OPTION = filename.jpg) *Create a copy of the image with an object detection overlay*
- **Video** : (OPTION = filename.mp4) *Create a copy of the video with an object detection overlay*
- **Directory** : (OPTION = directory_name/) *Create a copy of all file with an object detection overlay*
- **Global File Type** (OPTION = directory_name/*.jpg) *Create a copy of all file with an object detection overlay*
- **RTSP stream** : (OPTION = rtsp://170.93.143.139/rtplive/470011e600ef003a004ee33696235daa) *For live object detection from a stream*
- **RTMP stream** : (OPTION = rtmp://192.168.1.105/live/test) *For live object detection from a stream*
- **HTTP stream** : (OPTION = http://112.50.243.8/PLTV/88888888/224/3221225900/1.m3u8) *For live object detection from a stream*

The following file formats are currently supported:

- **Images:** bmp, jpg, jpeg, png, tif, tiff, dng, webp, mpo
- **Videos:** mov, avi, mp4, mpg, mpeg, m4v, wmv, mkv

https://docs.ultralytics.com/quick-start/

Copyright **FAST CAMPUS** Corp. All Rights Reserved

YOLO5 with CoLab

train

```
# Train YOLOv5s on COCO128 for 3 epochs
!python train.py --img 640 --batch 16 --epochs 3 --data coco128.yaml --weights yolov5s.pt --cache
```

```
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1    229245  models.yolo.Detect                      [80, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 19
Model summary: 270 layers, 7235389 parameters, 7235389 gradients, 16.6 GFLOPs

Transferred 349/349 items from yolov5s.pt
```

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 with CoLab

Fast campus

validation

YOLOv5 Tutorial

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

댓글 공유

목차

- Setup
- Inference
- Validate
  - COCO val
  - COCO test
- Train
- Visualize
  - ClearML Logging and Automation NEW
  - Weights & Biases Logging
  - Local Logging
- Environments
- Status
- Appendix

섹션

+ 코드 + 텍스트

RAM 디스크 수정 가능

[5]

10분

[6]

8분

```
# Run YOLOv5x on COCO test
!python val.py --weights yolov5x.pt --data coco.yaml --img 640 --iou 0.65 --half --task test
```

```
val: data=/content/yolov5/data/coco.yaml, weights=['yolov5x.pt'], batch_size=32, imgsz=640, conf_thres=0.001, iou_thres=0.65, task=test, device=, workers=
YOLOv5 🚀 v6.2-21-g1cd3e75 Python-3.7.13 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

Fusing layers...
YOLOv5x summary: 444 layers, 86705005 parameters, 0 gradients
test: Scanning '/content/datasets/coco/test-dev2017' images and labels...0 found, 20288 missing, 0 empty, 0 corrupt: 100% 20288/20288 [00:33<00:00, 612.56
test: WARNING: No labels found in /content/datasets/coco/test-dev2017.cache. See https://github.com/ultralytics/yolov5/wiki/Train-Custom-Data
test: New cache created: /content/datasets/coco/test-dev2017.cache
               Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 634/634 [06:35<00:00,  1.60it/s]
                 all      20288          0          0          0          0          0
WARNING: no labels found in test set, can not compute metrics without labels ⚠️
Speed: 0.1ms pre-process, 16.3ms inference, 1.3ms NMS per image at shape (32, 3, 640, 640)

Evaluating pycocotools mAP... saving runs/val/exp2/yolov5x_predictions.json...
loading annotations into memory...
Done (t=0.51s)
creating index...
index created!
Loading and preparing results...
pycocotools unable to run: Results do not correspond to current coco set
Results saved to runs/val/exp2
```

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 with CoLab

test

COCO test

Download COCO test2017 dataset (7GB - 40,000 images), to test model accuracy on test-dev set (**20,000 images, no labels**). Results are saved to a `*.json` file which should be **zipped** and submitted to the evaluation server at https://competitions.codalab.org/competitions/20794.

```
# Download COCO test-dev2017
!bash data/scripts/get_coco.sh --test
```

```
Downloading https://github.com/ultralytics/yolov5/releases/download/v1.0/coco2017labels.zip  ...
Downloading http://images.cocodataset.org/zips/test2017.zip ...
  %   % TTotal    %ot Ralec  ei  ve% d Re% ceXfiveredd  %  Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 -A-:-v-:-er- ag--e :-Sp-:ee--d  -  -:Ti--me:-  -     Ti  me0     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 67.7M  100 67.7M    0     0  27.6M      0  0:00:02  0:00:02 --:--:-- 34.1M
replace ../datasets/coco/LICENSE? [y]es, [n]o, [A]ll, [N]one, [r]ename:  NULL
(EOF or read error, treating as "[N]one" ...)
 60 6339M   60 3818M    0     0  70.7M      0  0:01:29  0:00:54  0:00:35 84.1M
```

```
# Run YOLOv5x on COCO test
!python val.py --weights yolov5x.pt --data coco.yaml --img 640 --iou 0.65 --half --task test
```

YOLOv5 Tutorial

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

목차

- Setup
- Inference
- Validate
  - COCO val
  - COCO test
- Train
- Visualize
  - ClearML Logging and Automation NEW
  - Weights & Biases Logging
  - Local Logging
- Environments
- Status
- Appendix

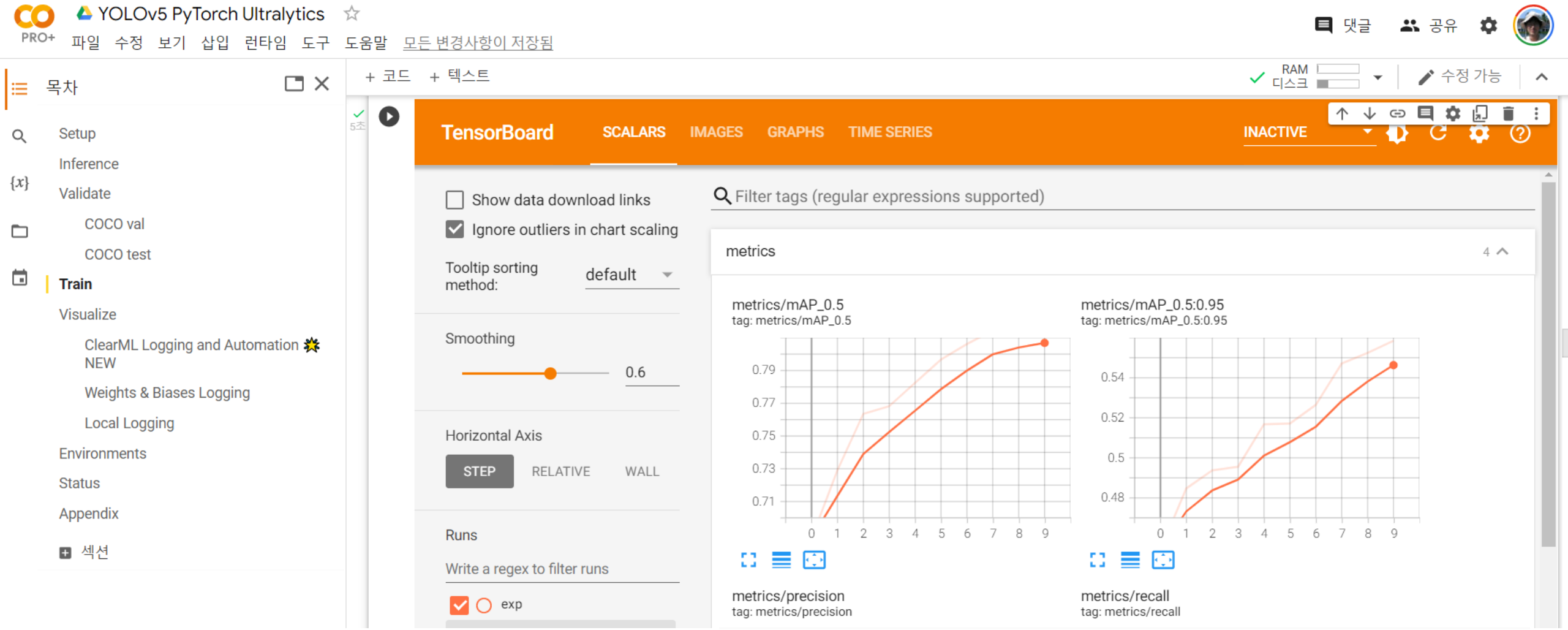
+ 코드 + 텍스트

```
100% 780M/780M [02:21<00:00, 8.52MB/s]
```

```
# Run YOLOv5x on COCO val
!python val.py --weights yolov5x.pt --data coco.yaml --img 640 --iou 0.65 --half
```

```
val: data=/content/yolov5/data/coco.yaml, weights=['yolov5x.pt'], batch_size=32, imgsz=640, conf_thres=0.001, iou_thres=0.65, task=val, device=, workers=8
YOLOv5 🚀 v6.2-21-g1cd3e75 Python-3.7.13 torch-1.12.1+cu113 CUDA:0 (Tesla T4, 15110MiB)

Downloading https://github.com/ultralytics/yolov5/releases/download/v6.2/yolov5x.pt to yolov5x.pt...
100% 166M/166M [00:30<00:00, 5.65MB/s]

Fusing layers...
YOLOv5x summary: 444 layers, 86705005 parameters, 0 gradients
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
100% 755k/755k [00:00<00:00, 130MB/s]
val: Scanning '/content/datasets/coco/val2017' images and labels...4952 found, 48 missing, 0 empty, 0 corrupt: 100% 5000/5000 [00:02<00:00, 2110.14it/s]
val: New cache created: /content/datasets/coco/val2017.cache
                 Class     Images     Labels          P          R     mAP@.5 mAP@.5:.95: 100% 157/157 [02:01<00:00,  1.29it/s]
                   all       5000      36335      0.743      0.625      0.683      0.504
Speed: 0.1ms pre-process, 16.0ms inference, 1.4ms NMS per image at shape (32, 3, 640, 640)

Evaluating pycocotools mAP... saving runs/val/exp/yolov5x_predictions.json...
loading annotations into memory...
Done (t=0.41s)
creating index...
index created!
Loading and preparing results...
DONE (t=5.75s)
creating index...
index created!
Running per image evaluation...
```

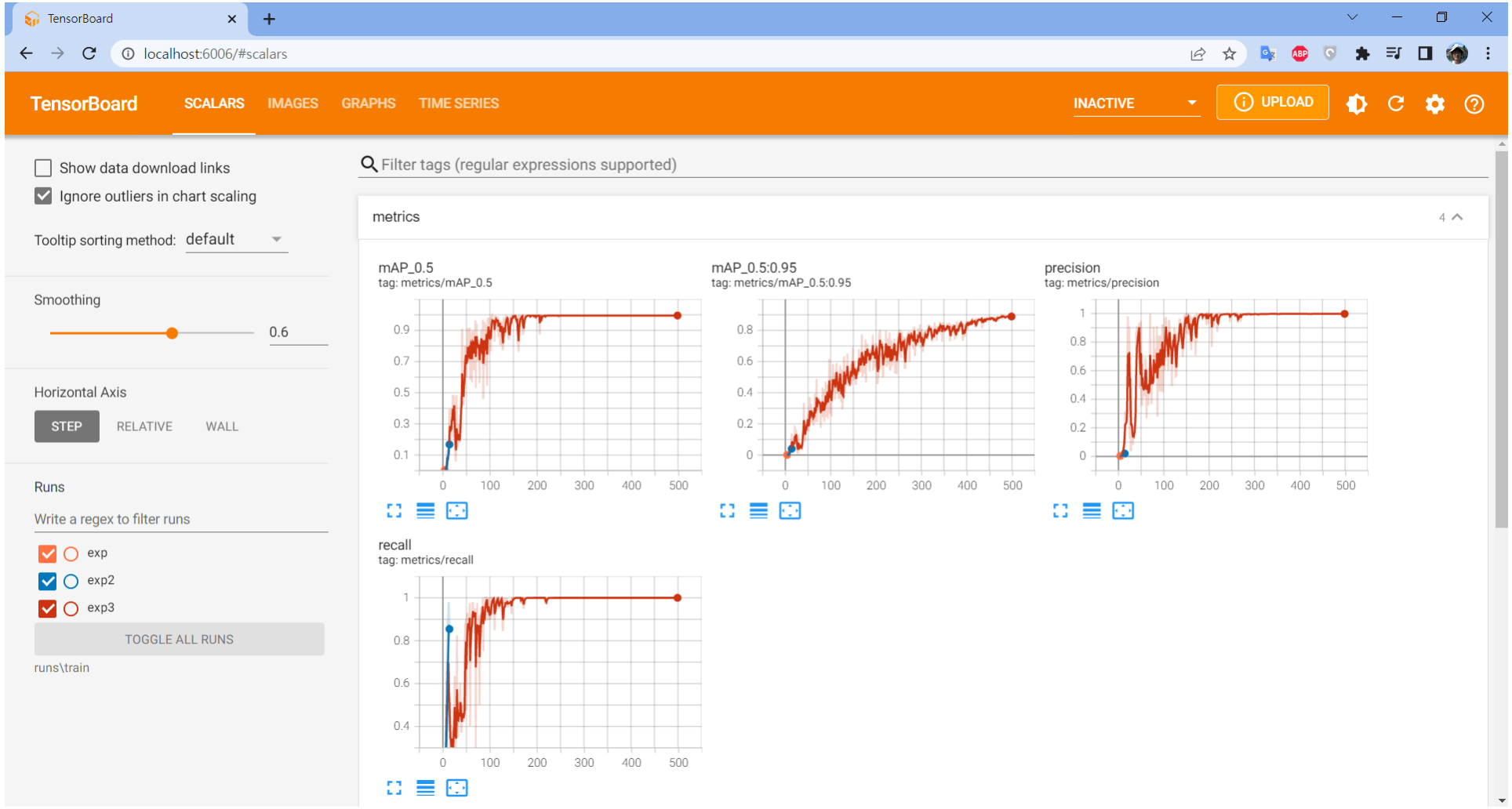
Copyright FAST CAMPUS Corp. All Rights Reserved


YOLOv5 PyTorch Ultralytics
PRO+
파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨
댓글 공유
목차
Setup
Inference
Validate
COCO val
COCO test
Train
Visualize
ClearML Logging and Automation NEW
Weights & Biases Logging
Local Logging
Environments
Status
Appendix
섹션
+ 코드 + 텍스트
RAM
디스크
수정 가능
TensorBoard
SCALARS
IMAGES
GRAPHS
TIME SERIES
INACTIVE
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method:
default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
exp
Filter tags (regular expressions supported)
metrics
metrics/mAP_0.5
tag: metrics/mAP_0.5
metrics/mAP_0.5:0.95
tag: metrics/mAP_0.5:0.95
metrics/precision
tag: metrics/precision
metrics/recall
tag: metrics/recall

monitoring

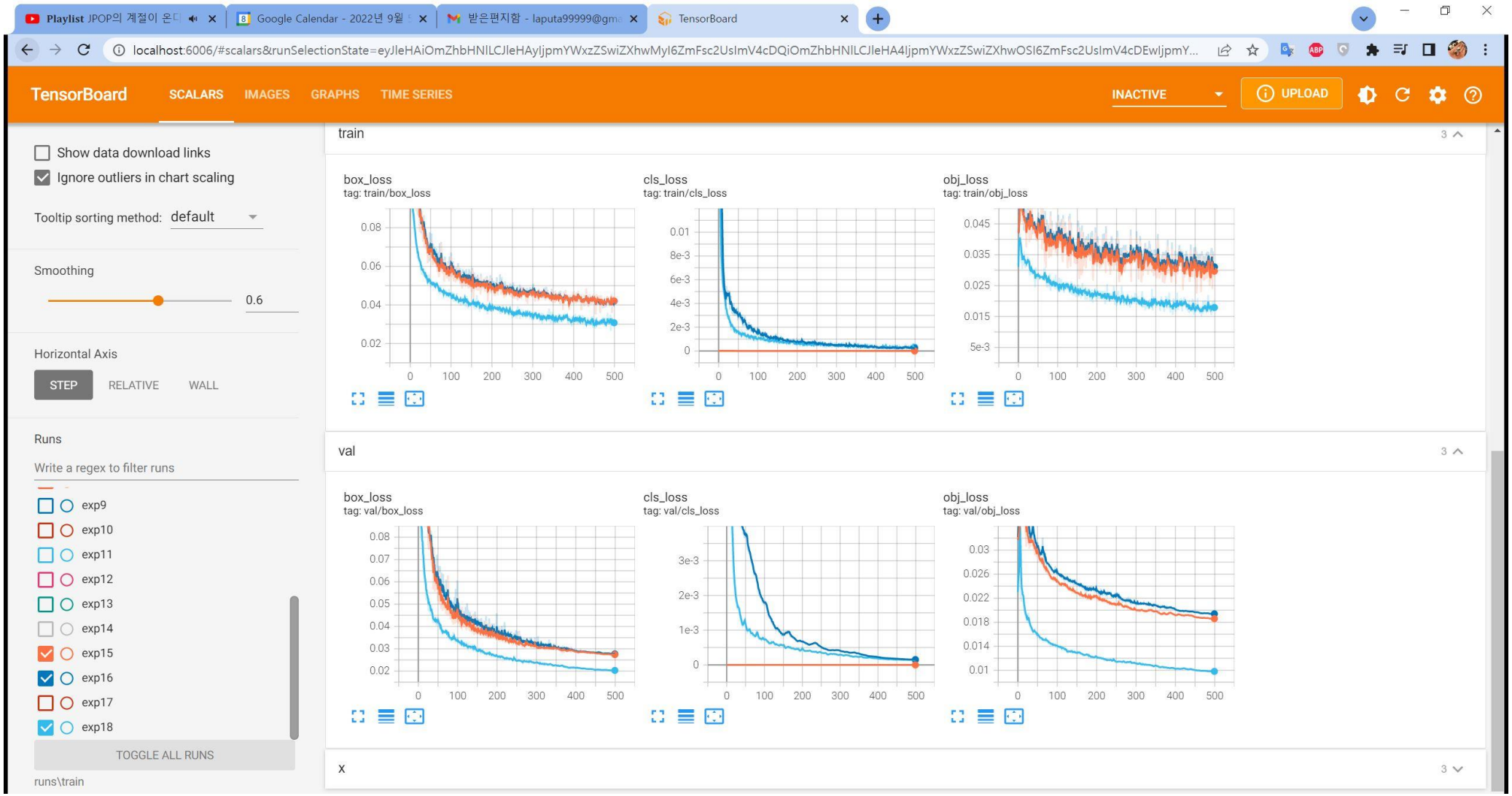
YOLO5 with CoLab

monitoring



Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO 5 in practice – Data Augmentation

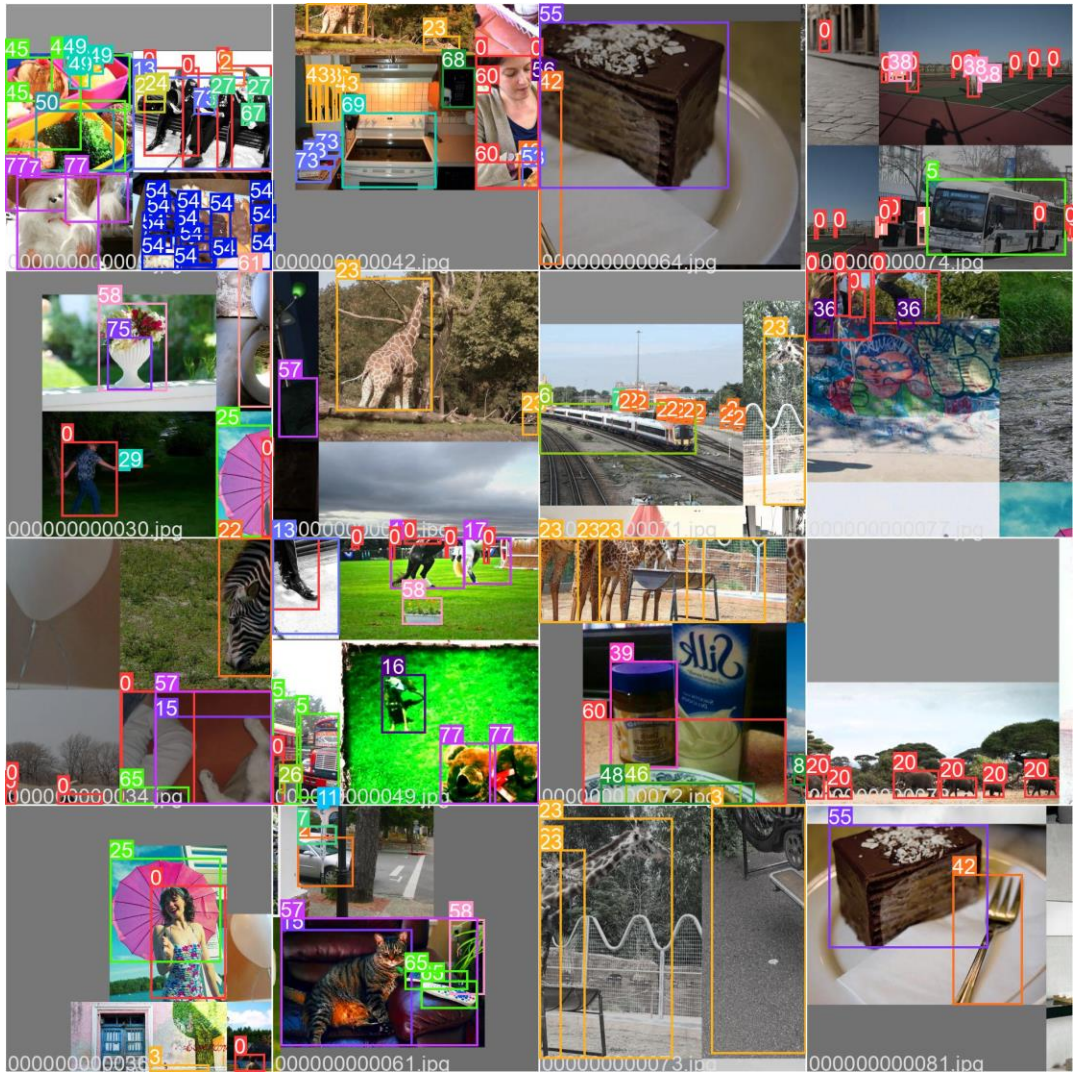


Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 in practice – Data Augmentation

augmentation

https://docs.ultralytics.com/FAQ/augmentation/



https://github.com/albumentations-team/albumentations

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 in practice using Jupyter Notebook, vscode

Fast campus

Hyper parameter

Hyperparameter Evolution

```
# Hyperparameter Evolution Results
# Generations: 1000
#                   P         R     mAP.5 mAP.5:.95       box       obj       cls
# Metrics:     0.4761      0.79     0.763    0.4951   0.01926   0.03286  0.003559

lr0: 0.01  # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.2  # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937  # SGD momentum/Adam beta1
weight_decay: 0.0005  # optimizer weight decay 5e-4
warmup_epochs: 3.0  # warmup epochs (fractions ok)
warmup_momentum: 0.8  # warmup initial momentum
warmup_bias_lr: 0.1  # warmup initial bias lr
box: 0.05  # box loss gain
cls: 0.5  # cls loss gain
cls_pw: 1.0  # cls BCELoss positive_weight
obj: 1.0  # obj loss gain (scale with pixels)
obj_pw: 1.0  # obj BCELoss positive_weight
iou_t: 0.20  # IoU training threshold
anchor_t: 4.0  # anchor-multiple threshold
anchors: 0  # anchors per output grid (0 to ignore)
fl_gamma: 0.0  # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015  # image HSV-Hue augmentation (fraction)
hsv_s: 0.7  # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4  # image HSV-Value augmentation (fraction)
degrees: 0.0  # image rotation (+/- deg)
translate: 0.1  # image translation (+/- fraction)
scale: 0.5  # image scale (+/- gain)
shear: 0.0  # image shear (+/- deg)
perspective: 0.0  # image perspective (+/- fraction), range 0-0.001
flipud: 0.0  # image flip up-down (probability)
fliplr: 0.5  # image flip left-right (probability)
mosaic: 1.0  # image mosaic (probability)
mixup: 0.0  # image mixup (probability)
```

We recommend a **minimum** of 300 generations of evolution for best results. Note that **evolution is generally expensive and time consuming**, as the base scenario is trained hundreds of times, possibly requiring hundreds or thousands of GPU hours.

Copyright **FAST CAMPUS** Corp. All Rights Reserved

YOLO5 with CoLab

Fast campus

experience

```
[9] !ls
```

```
coco2017labels.zip  export.py   __pycache__      setup.cfg       val.py
CONTRIBUTING.md     hubconf.py  README.md        train.py        yolov5s.pt
data                LICENSE     requirements.txt tutorial.ipynb  yolov5x.pt
detect.py           models      runs             utils
```

```
[20] %cd exp2
     !ls
```

```
/content/yolov5/runs/val/exp2
confusion_matrix.png   val_batch1_labels.jpg  val_batch2_pred.jpg
val_batch0_labels.jpg  val_batch1_pred.jpg    yolov5x_predictions.json
val_batch0_pred.jpg    val_batch2_labels.jpg
```

```
display.Image(filename='val_batch2_pred.jpg', width=600)
```

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 in practice – ML Ops process

Fast campus

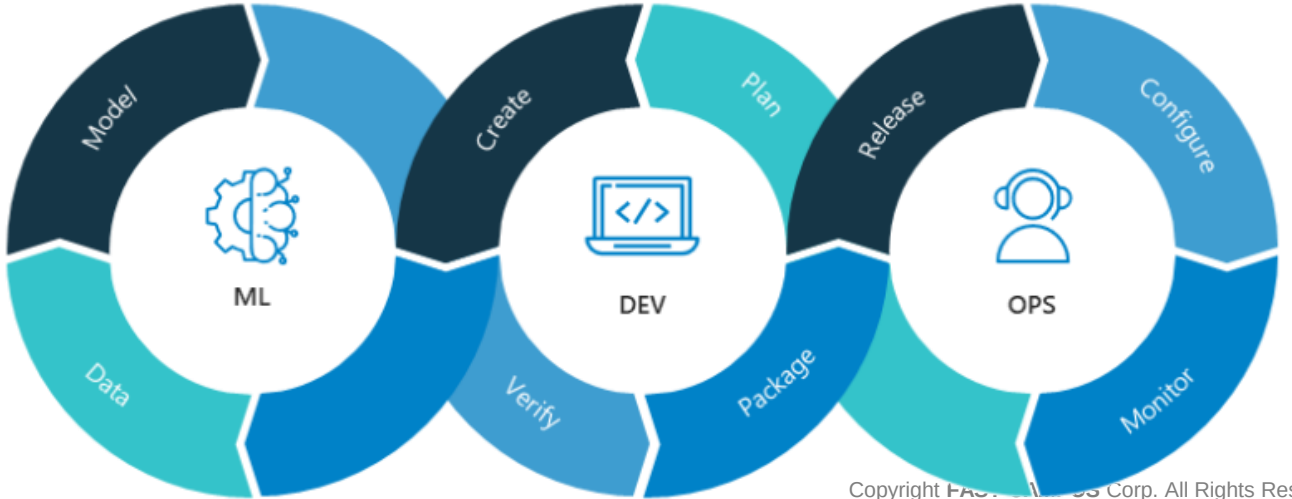
ML service development process

Define user requirement → Define service → Select model & dataset → Develop model & service → Test

ML Ops pipeline

Collect dataset, update & cleansing → Define hyperparameters → Train → Monitoring & review

Tuning hyperparam → Package & deploy model, service → Test → Release & monitoring service



Copyright FAST CAMPUS Corp. All Rights Reserved

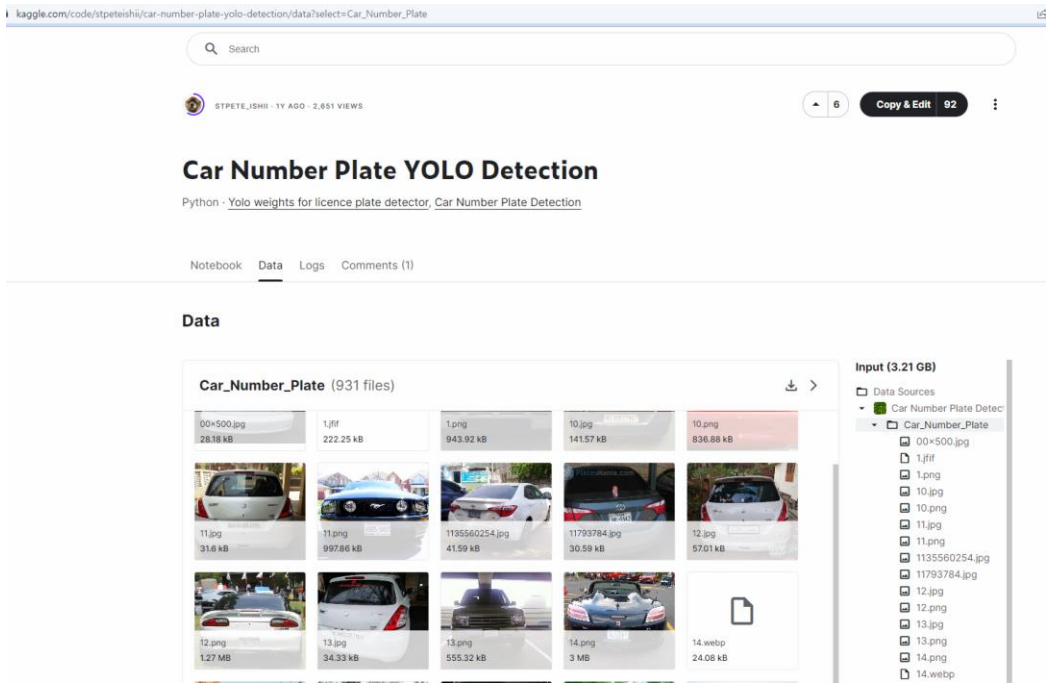
YOLO5 in practice – prepare dataset

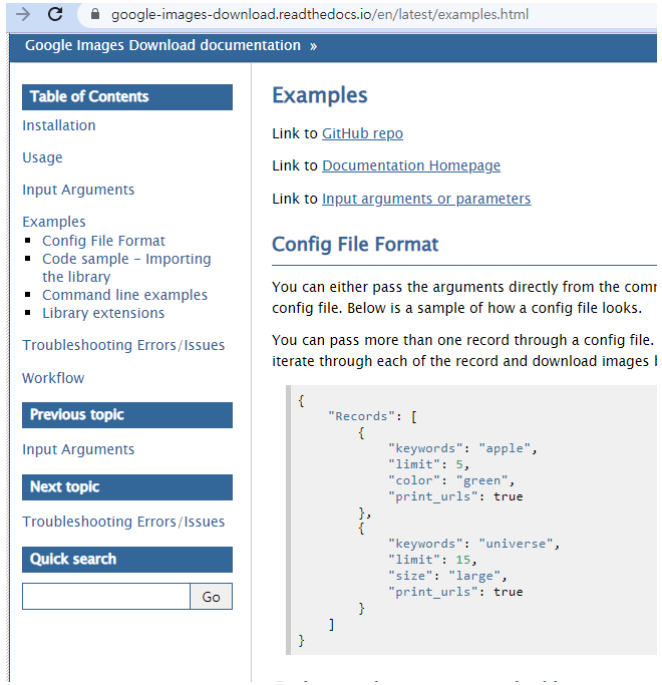
Data collection

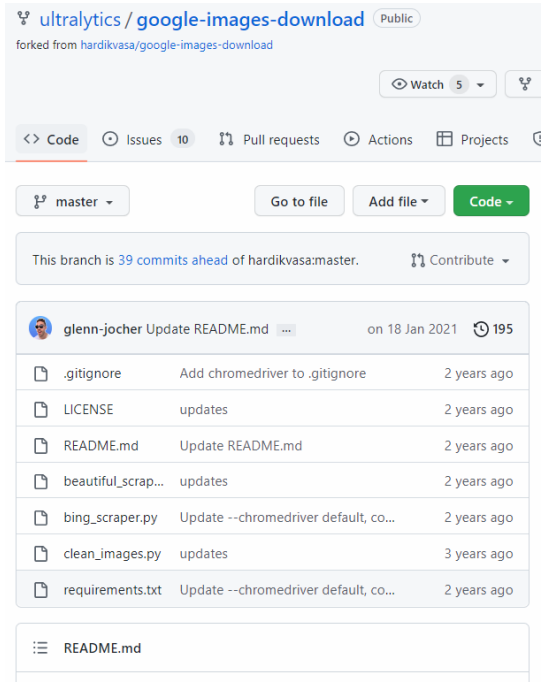
pip install git+https://github.com/Joeclinton1/google-images-download.git

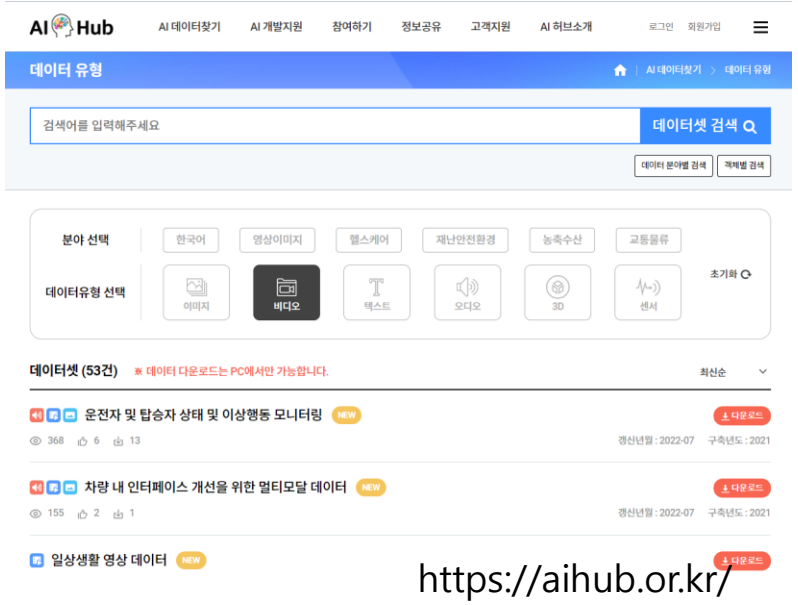
googleimagesdownload --keywords "construction" -sk "dozer" --limit 20

https://google-images-download.readthedocs.io/en/latest/examples.html









https://aihub.or.kr/

Copyright FAST CAMPUS Corp. All Rights Reserved

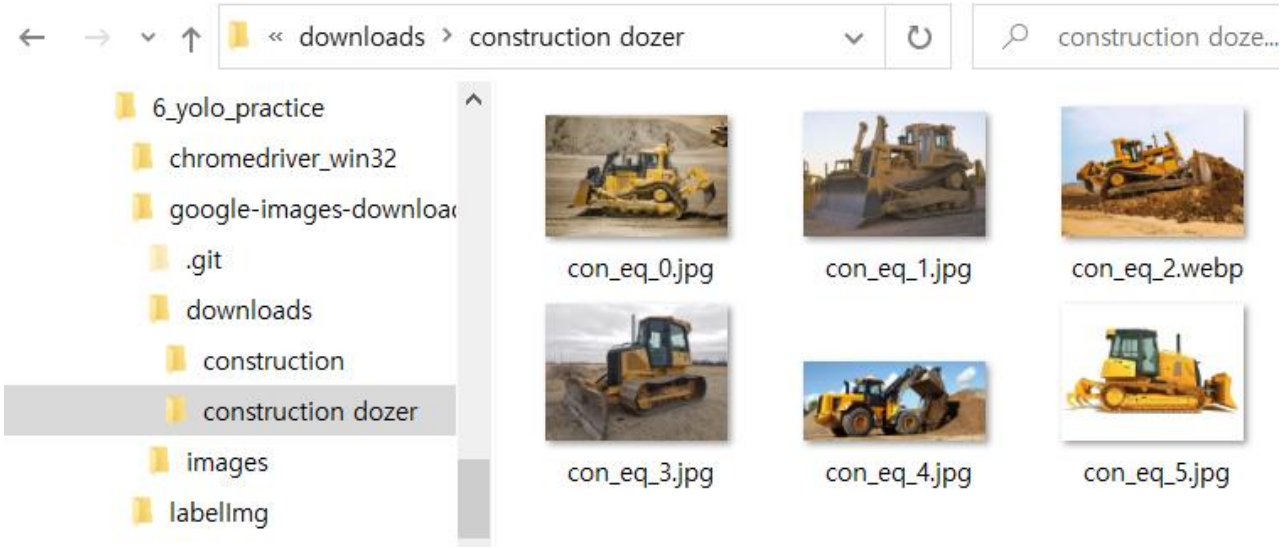
YOLO5 in practice – prepare dataset

Data collection

Data naming

python rename_fname.py --path ./downloads/"construction dozer"/

```
import os
import argparse

def rename_multiple_files(path,obj):
    i=0
    for filename in os.listdir(path):
        try:
            f,extension = os.path.splitext(path+filename)
            src=path+filename
            dst=path+obj+str(i)+extension
            os.rename(src,dst)
            i+=1
            print('Rename successful.')
        except BaseException as e:
            print(e)
            i+=1

parser = argparse.ArgumentParser()
parser.add_argument('--path', default='./downloads/construction/', help='function definition.
ex) earthwork, property, report')
parser.add_argument('--tag', default='con_eq_', help='model data file')
args = parser.parse_args()

rename_multiple_files(args.path, args.tag)
```



Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 in practice – labeling

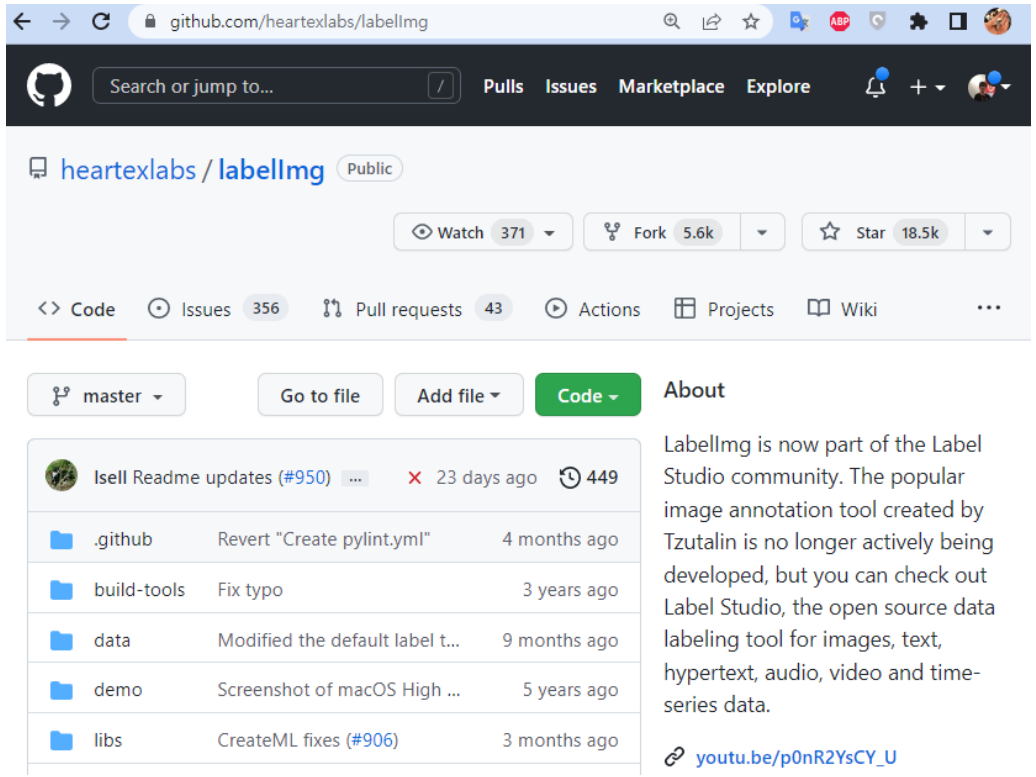
Labeling Tool

https://github.com/heartexlabs/labelImg

Windows

Install Python, PyQt5 and install lxml.

```
pip install PyQt5
pip install lxml

pyrcc5 -o libs/resources.py resources.qrc
For pyqt5, pyrcc5 -o libs/resources.py resources.qrc

python labelImg.py
python labelImg.py [IMAGE_PATH] [PRE-DEFINED CLASS FILE]
```



Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 in practice – labeling

Fast campus

README.rst

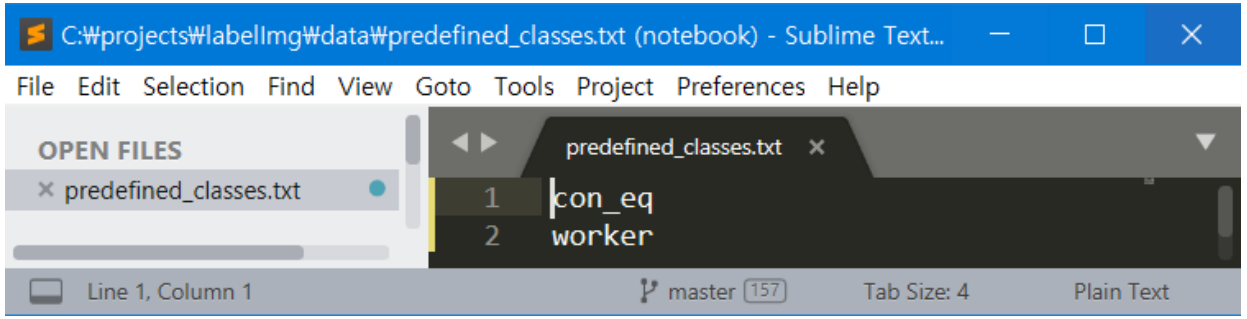
3. Select image in File List, it will appear the bounding box and label for all objects in that image.

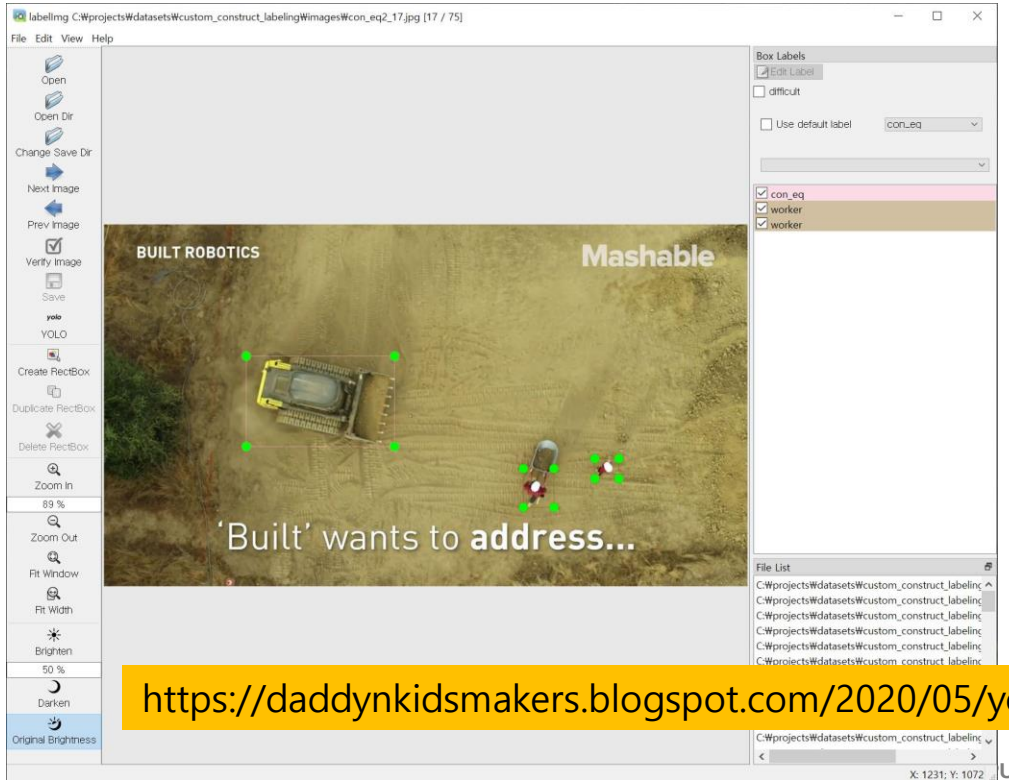
(Choose Display Labels mode in View to show/hide lablels)

Hotkeys

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
Ctrl + Shift + d	Delete the current image
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

Edit the data/predefined_classes.txt to load pre-defined classes

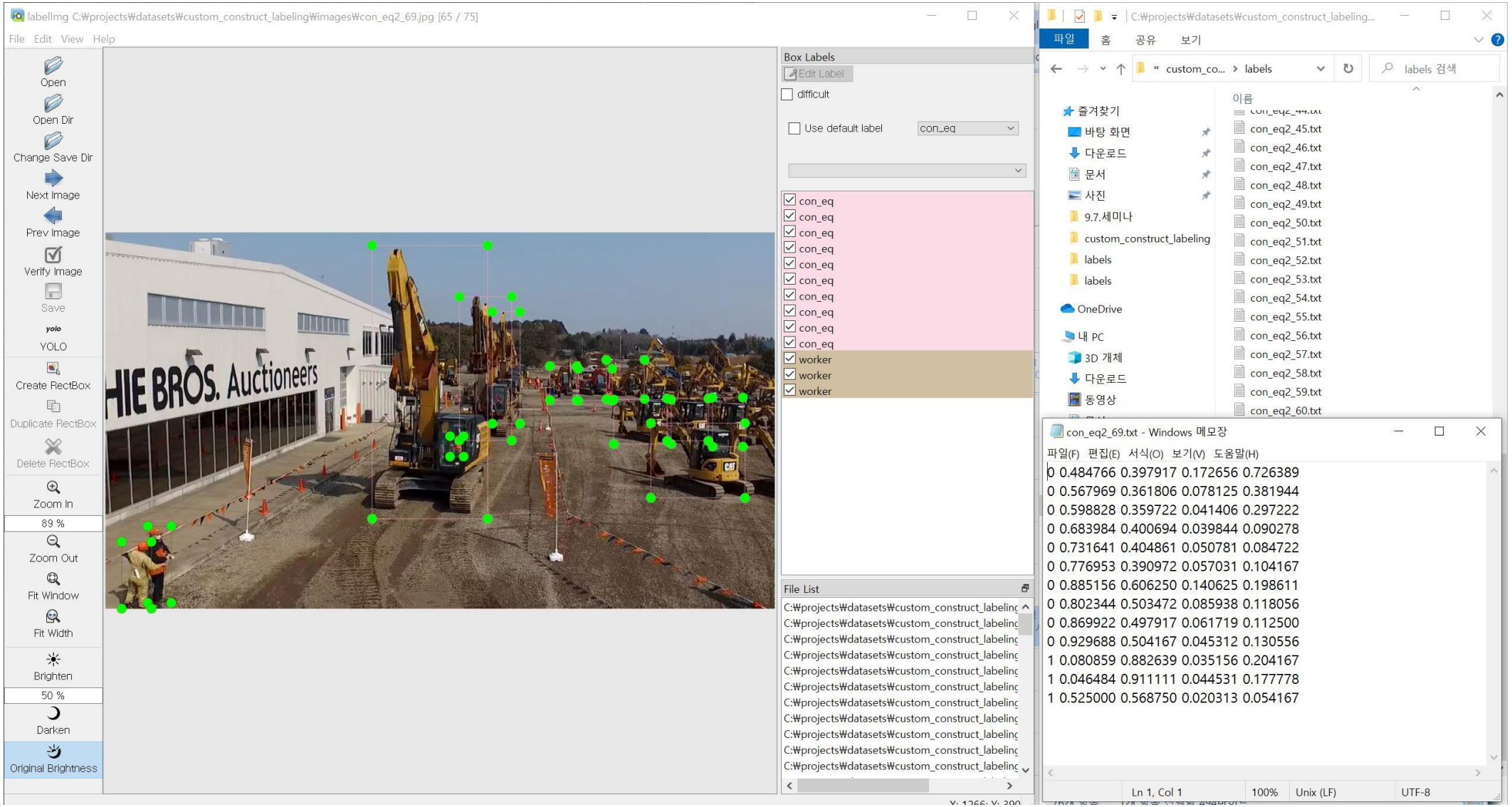




https://daddynkidsmakers.blogspot.com/2020/05/yolo.html

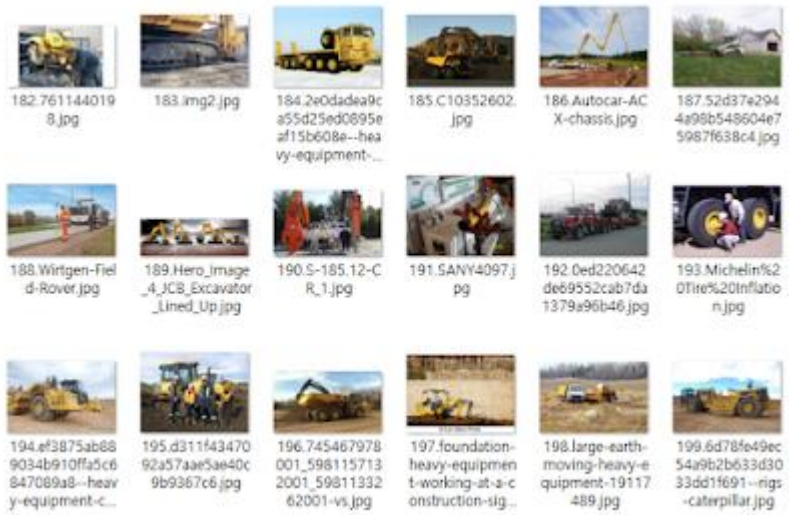
US Corp. All Rights Reserved

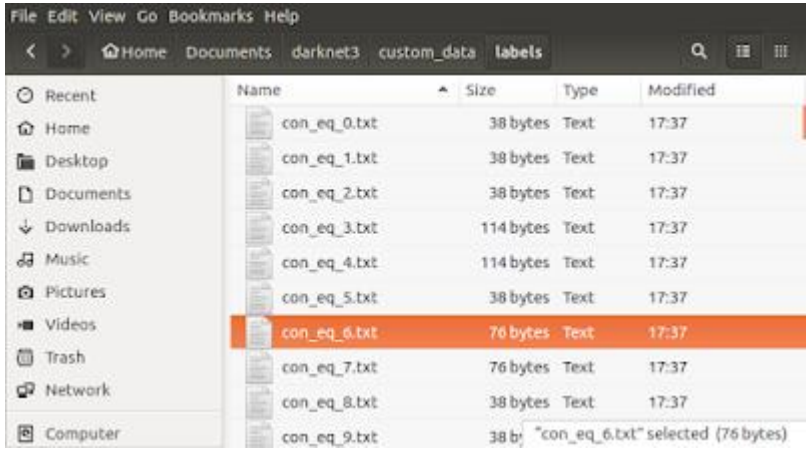
YOLO5 in practice – labeling

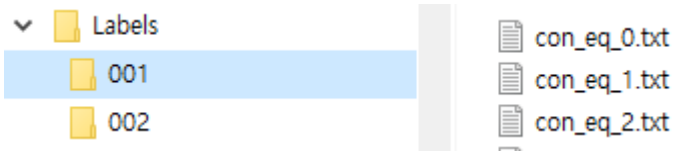


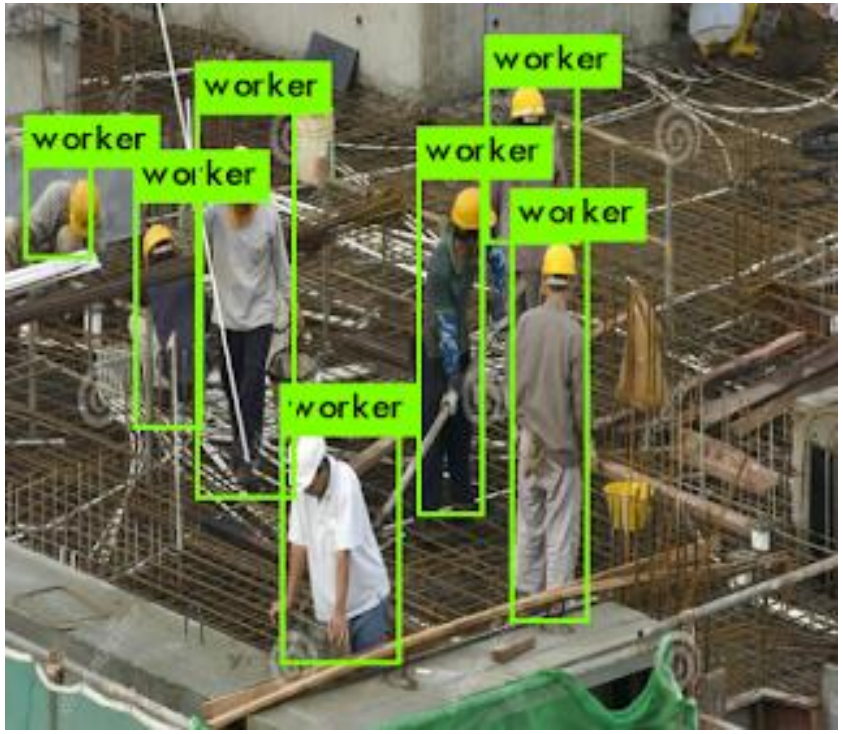
Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 in practice – labeling









Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – Train & Prediction

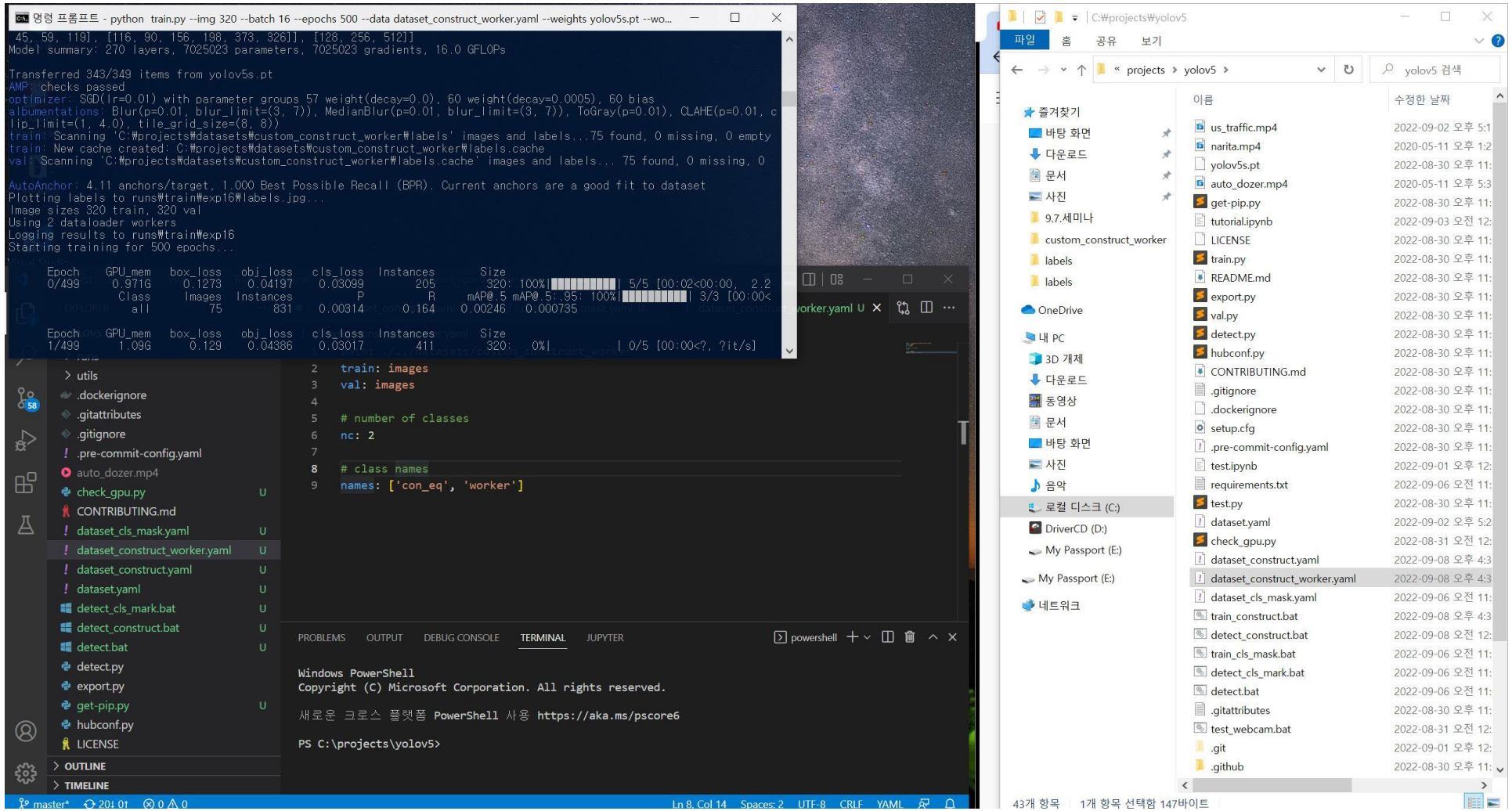
```
train: weights=yolov5s.pt, cfg=, data=dataset_construct_worker.yaml, hyp=data₩hyps₩hyp.scratch-low.yaml, epochs=500, batch_size=16, imgsz=320, rect=False, resume=False, nosave=False, noval=False, no
autoanchor=False, noplots=False, evolve=None, bucket=, cache=None, image_weights=False, device=, multi_scale=False, single_cls=False, optimizer=SGD, sync_bn=False, workers=2, project=runs₩train, nam
e=exp, exist_ok=False, quad=False, cos_lr=False, label_smoothing=0.0, patience=100, freeze=[0], save_period=-1, seed=0, local_rank=-1, entity=None, upload_dataset=False, bbox_interval=-1, artifact_a
lias=latest
github:  YOLOv5 is out of date by 23 commits. Use `git pull` or `git clone https://github.com/ultralytics/yolov5` to update.
YOLOv5  v6.2-77-g4a37381 Python-3.9.12 torch-1.12.1+cu113 CUDA:0 (NVIDIA GeForce RTX 3080 Laptop GPU, 8192MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.5, cls_pw=1.0, obj=1.0, obj_pw=1.0, iou_t=0.2, a
nchor_t=4.0, fl_gamma=0.0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0
Weights & Biases: run 'pip install wandb' to automatically track and visualize YOLOv5  runs in Weights & Biases
ClearML: run 'pip install clearml' to automatically track, visualize and remotely train YOLOv5  in ClearML
TensorBoard: Start with 'tensorboard --logdir runs₩train', view at http://localhost:6006/
Overriding model.yaml nc=80 with nc=2

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1     18879  models.yolo.Detect                      [2, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 198, 373, 326]], [128, 256, 512]]
Model summary: 270 layers, 7025023 parameters, 7025023 gradients, 16.0 GFLOPs

Transferred 343/349 items from yolov5s.pt
AMP: checks passed
optimizer: SGD(lr=0.01) with parameter groups 57 weight(decay=0.0), 60 weight(decay=0.0005), 60 bias
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01), CLAHE(p=0.01, clip_limit=(1, 4.0), tile_grid_size=(8, 8))
train: Scanning 'C:₩projects₩datasets₩custom_construct_worker₩labels'images and labels...186 found, 0 missing,
```

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO5 in practice – Train & Prediction

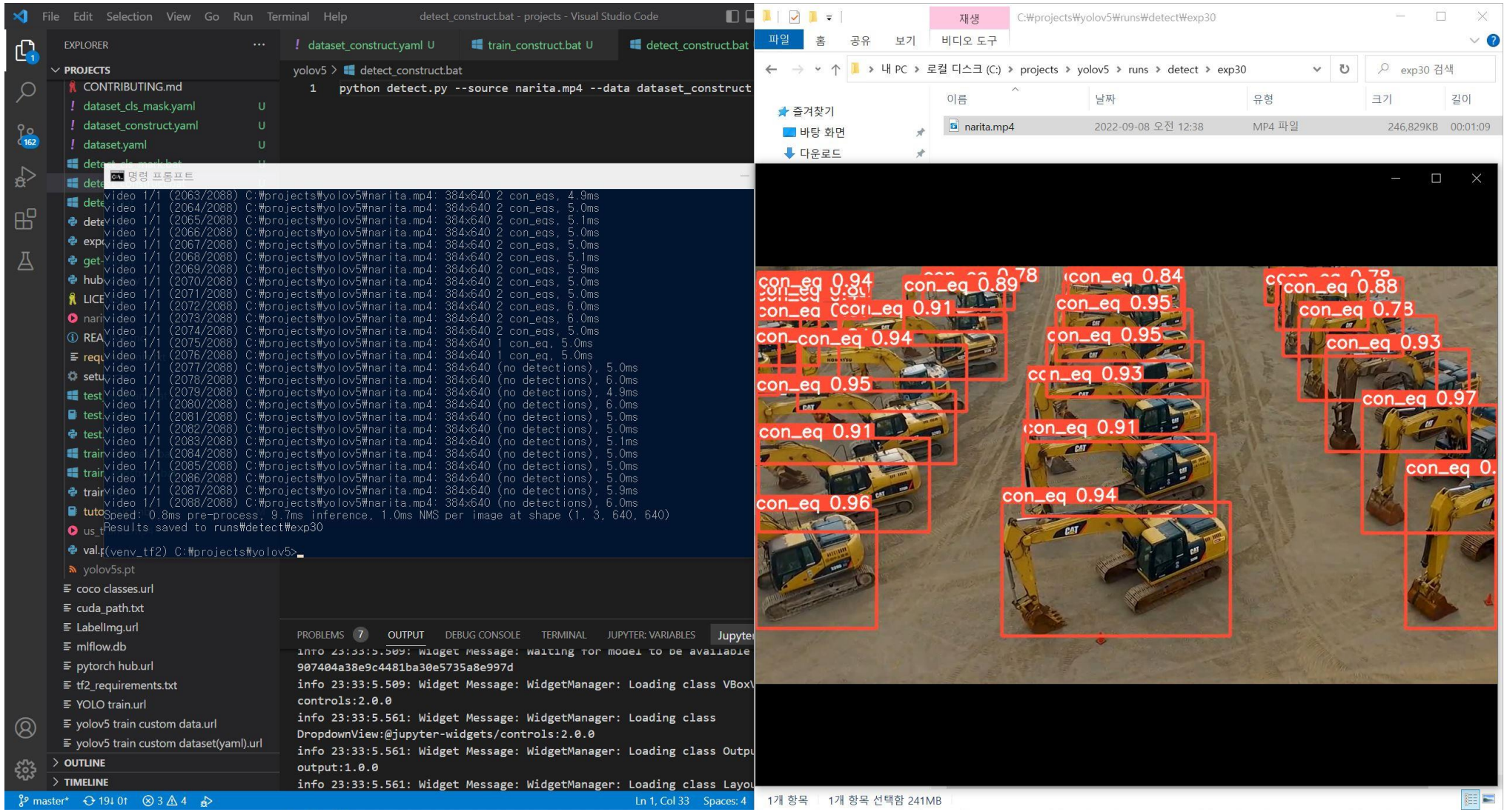
Custom data training

https://docs.ultralytics.com/tutorials/train-custom-datasets/



Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – Train & Prediction



Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – model serving with Flask

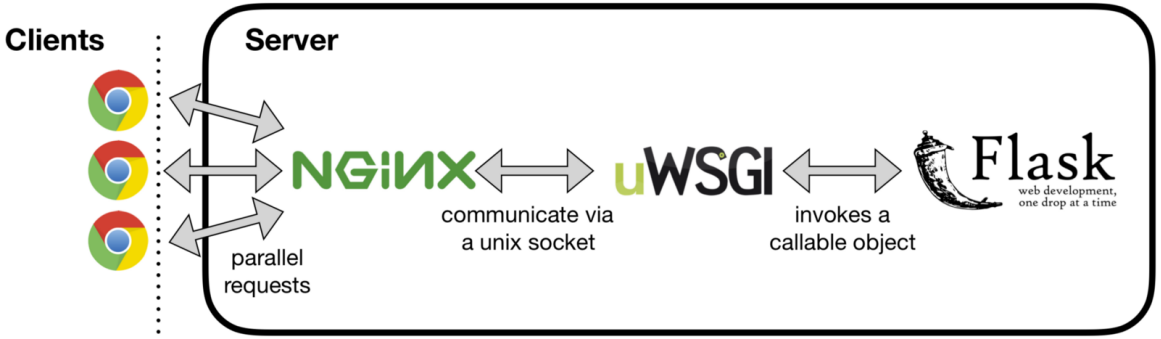
Fast campus

Deploy Object detection model using Flask

https://daddynkidsmakers.blogspot.com/2020/09/django-flask-open-api.html

```
pip install Flask
```

https://pypi.org/project/Flask/



main.py

```python
# main.py

from flask import Flask

app = Flask(__name__)

@app.route("/")
def index():
    return "Congratulations, it's a web app!"

if __name__ == "__main__":
    app.run(host="127.0.0.1", port=8080, debug=True)
```

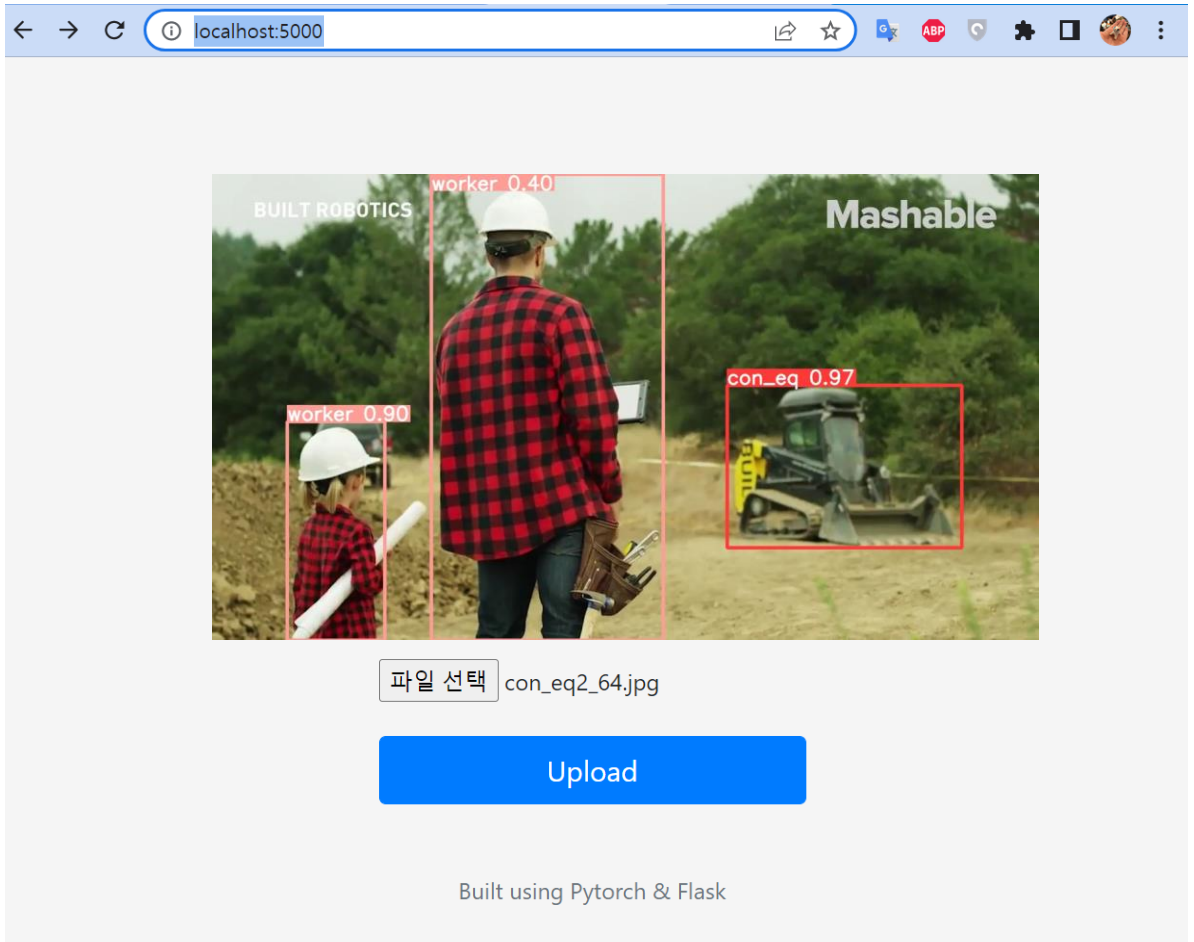
Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – model serving with Flask

Fast campus

Web service app & API development

Deploy Object detection model using Flask



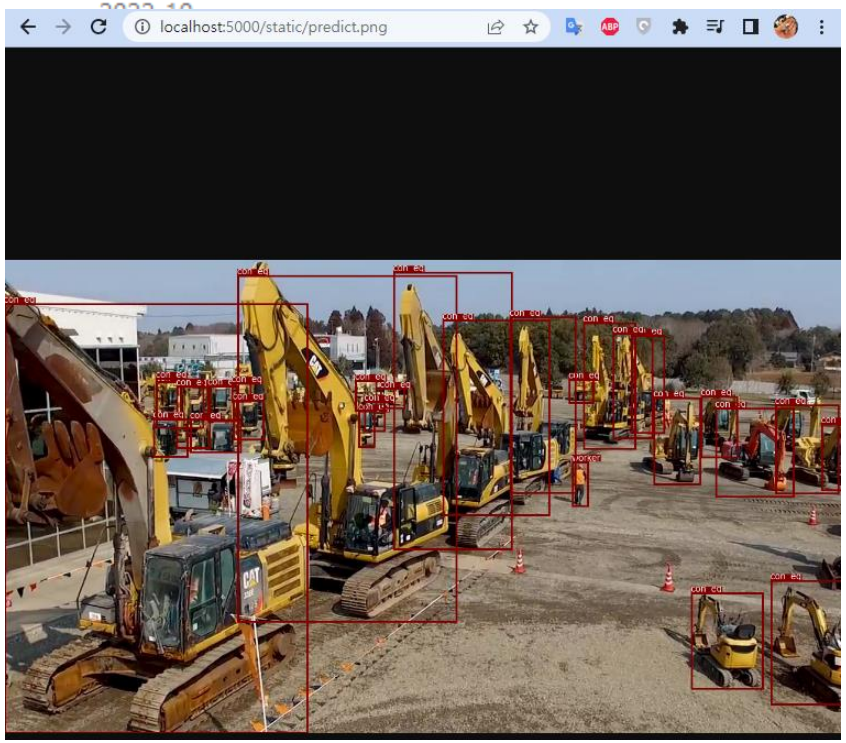
Yolov5 object detection model deployment using flaskhttps://github.com/robmarkcole/yolov5-flask

Web app folder

projects › notebook › 6_yolo_practice › yolov5_flask

이름	수정한 날
.git	
.github	
__pycache__	
docs	
models	
static	
templates	
tests	
utils	
con_eq2_69.jpg	
README.md	
a_file.png	
yolov5_con.pt	
yolov5s.pt	
export.py	
get-pip.py	
hubconf.py	
restapi.py	
webapp.py	
requirements.txt	
openapi_test.bat	
run.bat	
dataset_construct_worker.yaml	
.gitignore	
Dockerfile	

Flask



app = Flask(__name__)

parser = argparse.ArgumentParser(description="Flask App with custom yolov5 models")
model = torch.hub.load(curd, 'custom', path=curd + 'yolov5_con.pt', source='local')

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – model serving with Flask

Fast campus

Deploy Object detection model using Flask

Curl utility installation - https://curl.se/download.html

python restapi.py
curl -X POST -F image=@con_eq2_69.jpg "http://127.0.0.1:5000/v1/object_detection"

```
(venv_tf2) C:₩projects₩yolov5_flask>python restapi.py
YOLOv5  1.3-2-gacl8396 Python-3.9.12 torch-1.12.1+cu113 CUDA:0 (NVIDIA GeForce RTX 3080 Laptop GPU, 8192MiB)

Fusing layers...
Model summary: 213 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
Adding AutoShape...
 * Serving Flask app 'restapi'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.30.1.88:5000
Press CTRL+C to quit
127.0.0.1 - - [16/Oct/2022 08:02:03] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [16/Oct/2022 08:02:05] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [16/Oct/2022 08:02:06] "GET / HTTP/1.1" 404 -
127.0.0.1 - - [16/Oct/2022 08:02:07] "GET / HTTP/1.1" 404 -
```

명령 프롬프트

```
2022-10-16  오전 08:07                 81 openapi.bat
2022-10-09  오후 08:10                 18 run.bat
               2개 파일                 99 바이트
               0개 디렉터리  179,548,721,152 바이트 남음

C:₩projects₩yolov5_flask>openapi.bat

C:₩projects₩yolov5_flask>curl -X POST -F image=@con_eq2_69.jpg "http://127.0.0.1:5000/v1/object_detection"
[{"xmin":24.6365852356,"ymin":561.6345214844,"xmax":129.339263916,"ymax":720.0,"confidence":0.9224505424,"class":
1,"name":"worker"},{"xmin":526.0735473633,"ymin":33.5504608154,"xmax":732.2012329102,"ymax":541.9304199219,"confi
dence":0.9142798185,"class":0,"name":"con_eq"},{"xmin":1021.1021728516,"ymin":352.8305053711,"xmax":1228.39709472
66,"ymax":514.4751586914,"confidence":0.910230875,"class":0,"name":"con_eq"},{"xmin":908.7915039062,"ymin":256.17
62390137,"xmax":964.9847412109,"ymax":316.9440612793,"confidence":0.8944447041,"class":0,"name":"con_eq"},{"xmin"
:850.4646606445,"ymin":253.5209350586,"xmax":895.5513305664,"ymax":320.7725830078,"confidence":0.8598570824,"clas
s":0,"name":"con_eq"},{"xmin":983.1149291992,"ymin":266.1564941406,"xmax":1023.3728637695,"ymax":318.2346801758,"
confidence":0.8123363853,"class":0,"name":"con_eq"},{"xmin":691.6927490234,"ymin":109.5151672363,"xmax":793.22155
76172,"ymax":394.5242614746,"confidence":0.7660834789,"class":0,"name":"con_eq"},{"xmin":978.7466430664,"ymin":31
4.1014709473,"xmax":1081.8812255859,"ymax":403.5487365723,"confidence":0.7509086728,"class":0,"name":"con_eq"},{"
xmin":643.0079345703,"ymin":246.8774414062,"xmax":674.1931152344,"ymax":303.2550048828,"confidence":0.6875187159,
"class":0,"name":"con_eq"},{"xmin":964.1707763672,"ymin":258.2351989746,"xmax":1007.9593505859,"ymax":319.0886535
645,"confidence":0.604162395,"class":0,"name":"con_eq"},{"xmin":755.126159668,"ymin":161.3591308594,"xmax":803.83
53881836,"ymax":321.1946411133,"confidence":0.5691390634,"class":0,"name":"con_eq"},{"xmin":1163.9505615234,"ymin
":320.2859191895,"xmax":1243.7830810547,"ymax":410.7705993652,"confidence":0.428168267,"class":0,"name":"con_eq"}
,{"xmin":867.5435791016,"ymin":246.4231872559,"xmax":906.6706542969,"ymax":315.4702453613,"confidence":0.40350982
55,"class":0,"name":"con_eq"},{"xmin":967.5684204102,"ymin":314.8999938965,"xmax":1001.7764282227,"ymax":360.8254
699707,"confidence":0.3965543807,"class":0,"name":"con_eq"},{"xmin":1033.970703125,"ymin":237.0289916992,"xmax":1
096.1882324219,"ymax":337.1311645508,"confidence":0.3563090563,"class":0,"name":"con_eq"},{"xmin":1242.9893798828
,"ymin":337.2352294922,"xmax":1280.0,"ymax":457.3475952148,"confidence":0.3033725619,"class":0,"name":"con_eq"}]
C:₩projects₩yolov5_flask>
```

```
[{"xmin":24.6365852356,"ymin":561.6345214844,"xmax":129.339263916,"ymax":720.0,"confidence":
0.9224505424,"class":1,"name":"worker"},{"xmin":526.0735473633,"ymin":33.5504608154,"xmax":
732.2012329102,"ymax":541.9304199219,"confidence":0.9142798185,"class":0,"name":"con_eq"},{
"xmin":1021.1021728516,"ymin":352.8305053711,"xmax":1228.3970947266,"ymax":514.4751586914,"
confidence":0.910230875,"class":0,"name":"con_eq"},{"xmin":908.7915039062,"ymin":
256.1762390137,"xmax":964.9847412109,"ymax":316.9440612793,"confidence":0.8944447041,"class
":0,"name":"con_eq"},{"xmin":850.4646606445,"ymin":253.5209350586,"xmax":895.5513305664,"
ymax":320.7725830078,"confidence":0.8598570824,"class":0,"name":"con_eq"},{"xmin":
983.1149291992,"ymin":266.1564941406,"xmax":1023.3728637695,"ymax":318.2346801758,"
confidence":0.8123363853,"class":0,"name":"con_eq"},{"xmin":691.6927490234,"ymin":
109.5151672363,"xmax":793.2215576172,"ymax":394.5242614746,"confidence":0.7660834789,"class
":0,"name":"con_eq"},{"xmin":978.7466430664,"ymin":314.1014709473,"xmax":1081.8812255859,"
ymax":403.5487365723,"confidence":0.7509086728,"class":0,"name":"con_eq"},{"xmin":
643.0079345703,"ymin":246.8774414062,"xmax":674.1931152344,"ymax":303.2550048828,"
confidence":0.6875187159,"class":0,"name":"con_eq"},{"xmin":964.1707763672,"ymin":
258.2351989746,"xmax":1007.9593505859,"ymax":319.0886535645,"confidence":0.604162395,"class
":0,"name":"con_eq"},{"xmin":755.126159668,"ymin":161.3591308594,"xmax":803.8353881836,"
ymax":321.1946411133,"confidence":0.5691390634,"class":0,"name":"con_eq"},{"xmin":
1163.9505615234,"ymin":320.2859191895,"xmax":1243.7830810547,"ymax":410.7705993652,"
confidence":0.428168267,"class":0,"name":"con_eq"},{"xmin":867.5435791016,"ymin":
246.4231872559,"xmax":906.6706542969,"ymax":315.4702453613,"confidence":0.4035098255,"class
":0,"name":"con_eq"},{"xmin":967.5684204102,"ymin":314.8999938965,"xmax":1001.7764282227,"
ymax":360.8254699707,"confidence":0.3965543807,"class":0,"name":"con_eq"},{"xmin":
1033.970703125,"ymin":237.0289916992,"xmax":1096.1882324219,"ymax":337.1311645508,"
confidence":0.3563090563,"class":0,"name":"con_eq"},{"xmin":1242.9893798828,"ymin":
337.2352294922,"xmax":1280.0,"ymax":457.3475952148,"confidence":0.3033725619,"class":0,"
name":"con_eq"}]
```

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – hyperparameter tuning

https://docs.ultralytics.com/tutorials/hyperparameter-evolution/

This guide explains **hyperparameter evolution** for YOLOv5 🚀. Hyperparameter evolution is a method of Hyperparameter Optimization using a Genetic Algorithm (GA) for optimization.

Hyperparameters in ML control various aspects of training, and finding optimal values for them can be a challenge. Traditional methods like grid searches can quickly become intractable due to 1) the high dimensional search space 2) unknown correlations among the dimensions, and 3) expensive nature of evaluating the fitness at each point, making GA a suitable candidate for hyperparameter searches.

```
# Single-GPU
python train.py --epochs 10 --data coco128.yaml --weights yolov5s.pt --cache --evolve

# Multi-GPU
for i in 0 1 2 3; do
  nohup python train.py --epochs 10 --data coco128.yaml --weights yolov5s.pt --cache --evolve --device $i > evolve_gpu_$i.log &
done

# Multi-GPU bash while (not recommended)
for i in 0 1 2 3; do
  nohup "$(while true; do python train.py ... --evolve --device $i; done)" > evolve_gpu_$i.log &
done
```

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – hyperparameter tuning

python train.py --img 320 --batch 16 --epochs 10 --data dataset_construct_worker.yaml --weights yolov5s.pt --workers 2 --evolve

```
hyperparameters: lr0=0.00996, lrf=0.01, momentum=0.93161, weight_decay=0.00053, warmup_epochs=2.92743, warmup_mom
entum=0.78749, warmup_bias_lr=0.09683, box=0.04711, cls=0.5058, cls_pw=0.98156, obj=1.00591, obj_pw=1.04469, iou_
t=0.2, anchor_t=3.9649, fl_gamma=0.0, hsv_h=0.01439, hsv_s=0.72849, hsv_v=0.40369, degrees=0.0, translate=0.09548
, scale=0.46556, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0, ancho
rs=3.37044
```

```
hyperparameters: lr0=0.01192, lrf=0.01, momentum=0.94331, weight_decay=0.00051, warmup_epochs=3.0, warmup_momentu
m=0.87699, warmup_bias_lr=0.10726, box=0.04084, cls=0.45311, cls_pw=1.06439, obj=1.0, obj_pw=0.85243, iou_t=0.2,
anchor_t=4.71122, fl_gamma=0.0, hsv_h=0.01511, hsv_s=0.85195, hsv_v=0.53562, degrees=0.0, translate=0.10038, scal
e=0.54068, shear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0, anchors=3.1
6769
```

```
Logging results to runs#evolve#exp
Starting training for 10 epochs...

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
        0/9      1.01G     0.1279    0.03064    0.03426        130        320: 100%|          | 12/12 [00:01<00:

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
        1/9      1.01G     0.1162    0.03759    0.02496        224        320: 100%|          | 12/12 [00:01<00:
```

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – hyperparameter tuning

Fast campus

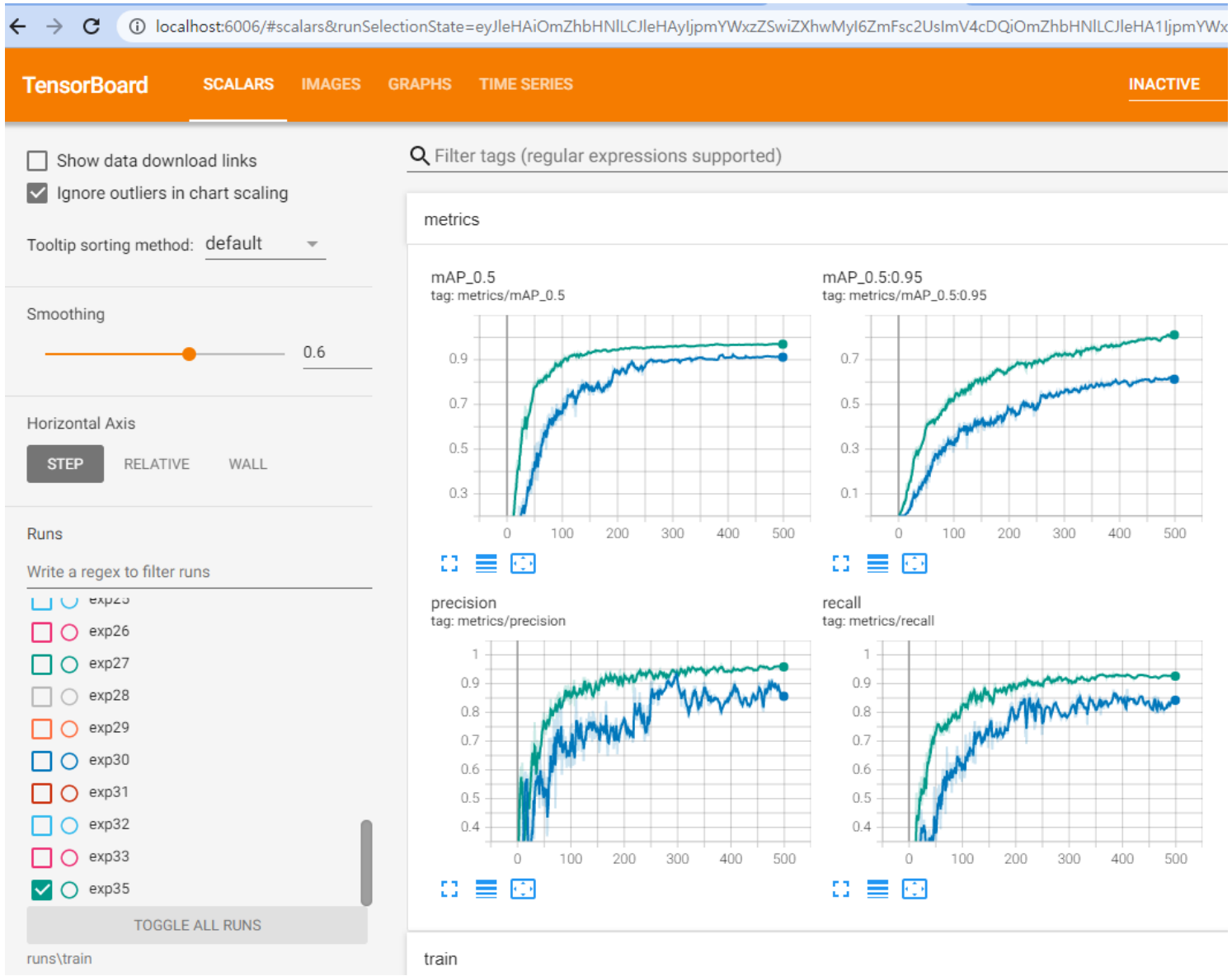
```python
# Train
if not opt.evolve:
    train(opt.hyp, opt, device, callbacks)

# Evolve hyperparameters (optional)
else:
    # Hyperparameter evolution metadata (mutation scale 0-1, lower_limit, upper_limit)
    meta = {
        'lr0': (1, 1e-5, 1e-1),  # initial learning rate (SGD=1E-2, Adam=1E-3)
        'lrf': (1, 0.01, 1.0),  # final OneCycleLR learning rate (lr0 * lrf)
        'momentum': (0.3, 0.6, 0.98),  # SGD momentum/Adam beta1
        'weight_decay': (1, 0.0, 0.001),  # optimizer weight decay
        'warmup_epochs': (1, 0.0, 5.0),  # warmup epochs (fractions ok)
        'warmup_momentum': (1, 0.0, 0.95),  # warmup initial momentum
        'warmup_bias_lr': (1, 0.0, 0.2),  # warmup initial bias lr
        'box': (1, 0.02, 0.2),  # box loss gain
        'cls': (1, 0.2, 4.0),  # cls loss gain
        'cls_pw': (1, 0.5, 2.0),  # cls BCELoss positive_weight
        'obj': (1, 0.2, 4.0),  # obj loss gain (scale with pixels)
        'obj_pw': (1, 0.5, 2.0),  # obj BCELoss positive_weight
        'iou_t': (0, 0.1, 0.7),  # IoU training threshold
        'anchor_t': (1, 2.0, 8.0),  # anchor-multiple threshold
        'anchors': (2, 2.0, 10.0),  # anchors per output grid (0 to ignore)
        'fl_gamma': (0, 0.0, 2.0),  # focal loss gamma (efficientDet default gamma=1.5)
        'hsv_h': (1, 0.0, 0.1),  # image HSV-Hue augmentation (fraction)
        'hsv_s': (1, 0.0, 0.9),  # image HSV-Saturation augmentation (fraction)
        'hsv_v': (1, 0.0, 0.9),  # image HSV-Value augmentation (fraction)
        'degrees': (1, 0.0, 45.0),  # image rotation (+/- deg)
        'translate': (1, 0.0, 0.9),  # image translation (+/- fraction)
```

```
Best results from row 129 of runs\evolve\exp\evolve.csv:
            lr0: 0.0126
            lrf: 0.01
       momentum: 0.974
   weight_decay: 0.00051
  warmup_epochs: 2.44
warmup_momentum: 0.916
 warmup_bias_lr: 0.0875
            box: 0.0585
            cls: 0.458
         cls_pw: 0.666
            obj: 1.01
         obj_pw: 0.73
          iou_t: 0.2
       anchor_t: 3.8
       fl_gamma: 0
          hsv_h: 0.0213
          hsv_s: 0.489
          hsv_v: 0.344
        degrees: 0
      translate: 0.108
          scale: 0.285
          shear: 0
    perspective: 0
         flipud: 0
         fliplr: 0.5
         mosaic: 0.445
          mixup: 0
     copy_paste: 0
        anchors: 3.8
Saved runs\evolve\exp\evolve.png
Hyperparameter evolution finished 300 generations
Results saved to runs\evolve\exp
Usage example: $ python train.py --hyp runs\evolve\exp\hyp_evolve.yaml
```

	A	B	C	D	E	F	G	H
1	metrics/precision	metrics/recall	metrics/mAP_0.5	metrics/mAP_0.5:0.95	val/box_loss	val/obj_loss	val/cls_loss	lr0
2	0.65503	0.16565	0.13569	0.043628	0.068726	0.026783	0.0079198	0.01
3	0.66795	0.16157	0.14221	0.044973	0.067879	0.026229	0.0078405	0.01
4	0.66794	0.15953	0.1296	0.044188	0.066781	0.02401	0.007116	0.00949
5	0.66225	0.16208	0.13056	0.043896	0.067468	0.026804	0.0080328	0.00996
6	0.66396	0.15189	0.12123	0.039392	0.0681	0.024201	0.0071323	0.00949
7	0.65616	0.1631	0.11626	0.034185	0.072554	0.025434	0.0083347	0.01
8	0.6866	0.15392	0.13894	0.045315	0.064363	0.027134	0.0080749	0.00996

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – hyperparameter tuning



Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – hyperparameter tuning

```
# from sklearn.model_selection import GridSearchCV
from ray.tune.sklearn import TuneGridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.datasets import load_iris
import numpy as np

iris = load_iris()
X = iris.data
y = iris.target

x_train, x_test, y_train, y_test = train_test_split(X,y,test_size = 0.3,random_state = 14)

# Example parameters to tune from SGDClassifier
parameter_grid = {"alpha": [1e-4, 1e-1, 1], "epsilon": [0.01, 0.1]}

tune_search = TuneGridSearchCV(
    SGDClassifier(),
    parameter_grid,
    early_stopping=True,
    max_iters=10)

tune_search.fit(x_train, y_train)

#best set of perameter
print(tune_search.best_params_)

#best score with best set of perameters
print(tune_search.best_score)
```

```
!pip install keras-tuner -q
```

```
import keras_tuner

build_model(keras_tuner.HyperParameters())

def build_model(hp):
    model = keras.Sequential()
    model.add(layers.Flatten())
    model.add(
        layers.Dense(
            # Tune number of units.
            units=hp.Int("units", min_value=32, max_value=512, step=32),
            # Tune the activation function to use.
            activation=hp.Choice("activation", ["relu", "tanh"]),
        )
    )
    # Tune whether to use dropout.
    if hp.Boolean("dropout"):
        model.add(layers.Dropout(rate=0.25))
    model.add(layers.Dense(10, activation="softmax"))
    # Define the optimizer learning rate as a hyperparameter.
    learning_rate = hp.Float("lr", min_value=1e-4, max_value=1e-2, sampling="log")
    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=learning_rate),
        loss="categorical_crossentropy",
        metrics=["accuracy"],
    )
    return model


build_model(keras_tuner.HyperParameters())
```

Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – hyperparameter tuning

https://docs.ray.io/en/latest/tune/index.html

Hyperparameter optimization frameworks
https://towardsdatascience.com/10-hyperparameter-optimization-frameworks-8bc87bc8b7e3

Hyperparameter Tuning in Python: a Complete Guide
https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide

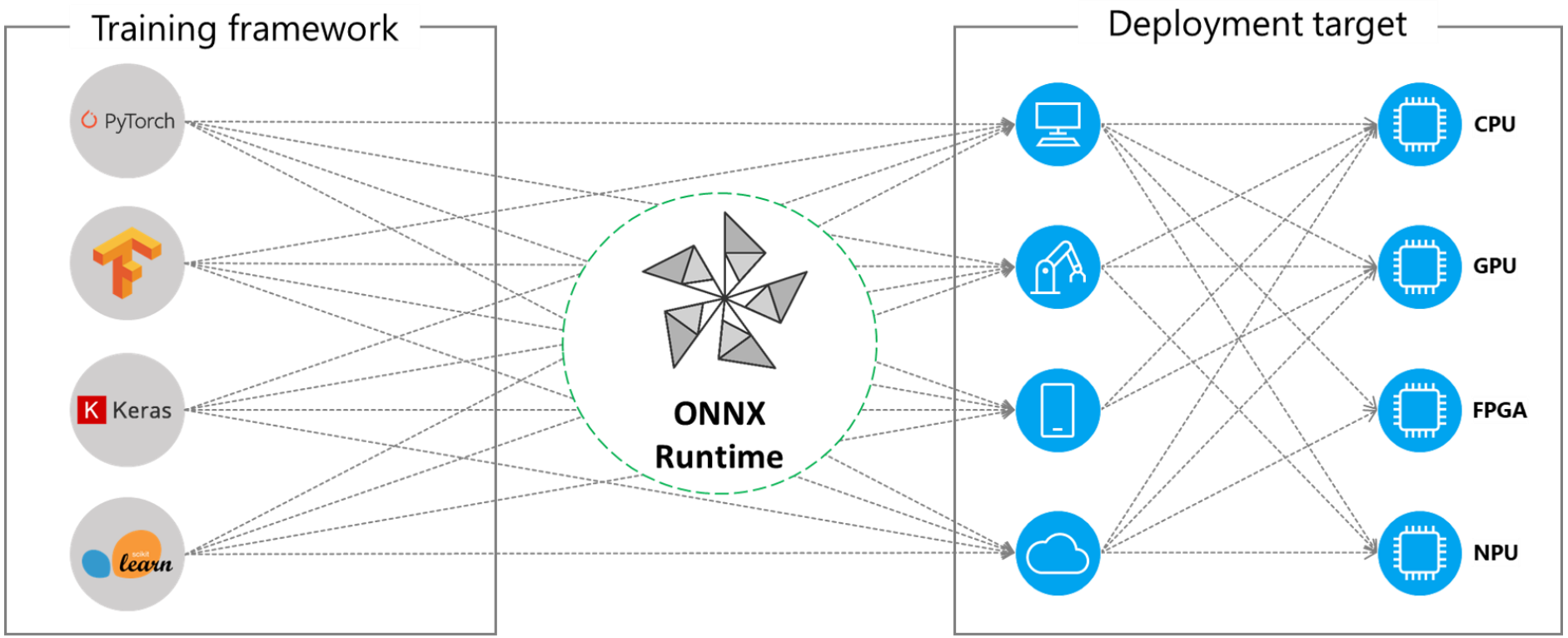
Open-Source Hyperparameter Optimisation Libraries For ML Models
https://analyticsindiamag.com/top-10-open-source-hyperparameter-optimisation-libraries-for-ml-models/

How To Use Keras Tuner for Hyper-parameter Tuning of Deep Learning Models
https://analyticsindiamag.com/how-to-use-keras-tuner-for-hyper-parameter-tuning-of-deep-learning-models/

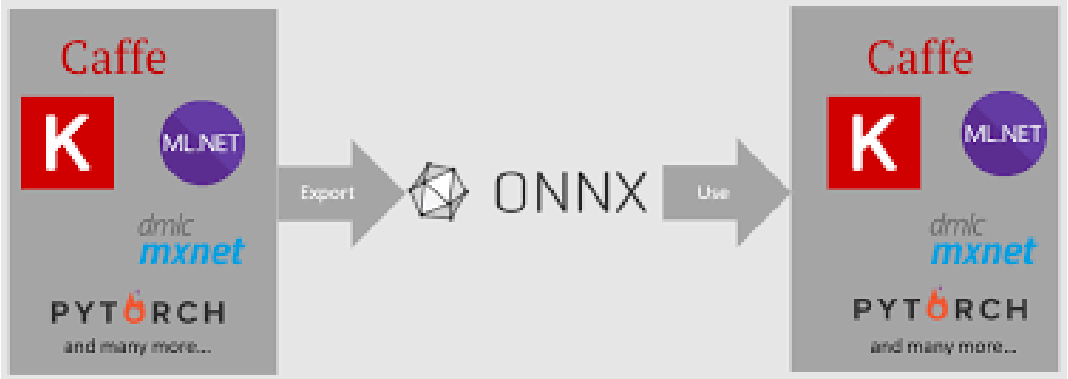
Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – ONNX

Fast campus



https://onnxruntime.ai/docs/tutorials/web/classify-images-nextjs-github-template.html



https://docs.ultralytics.com/tutorials/torchscript-onnx-coreml-export/

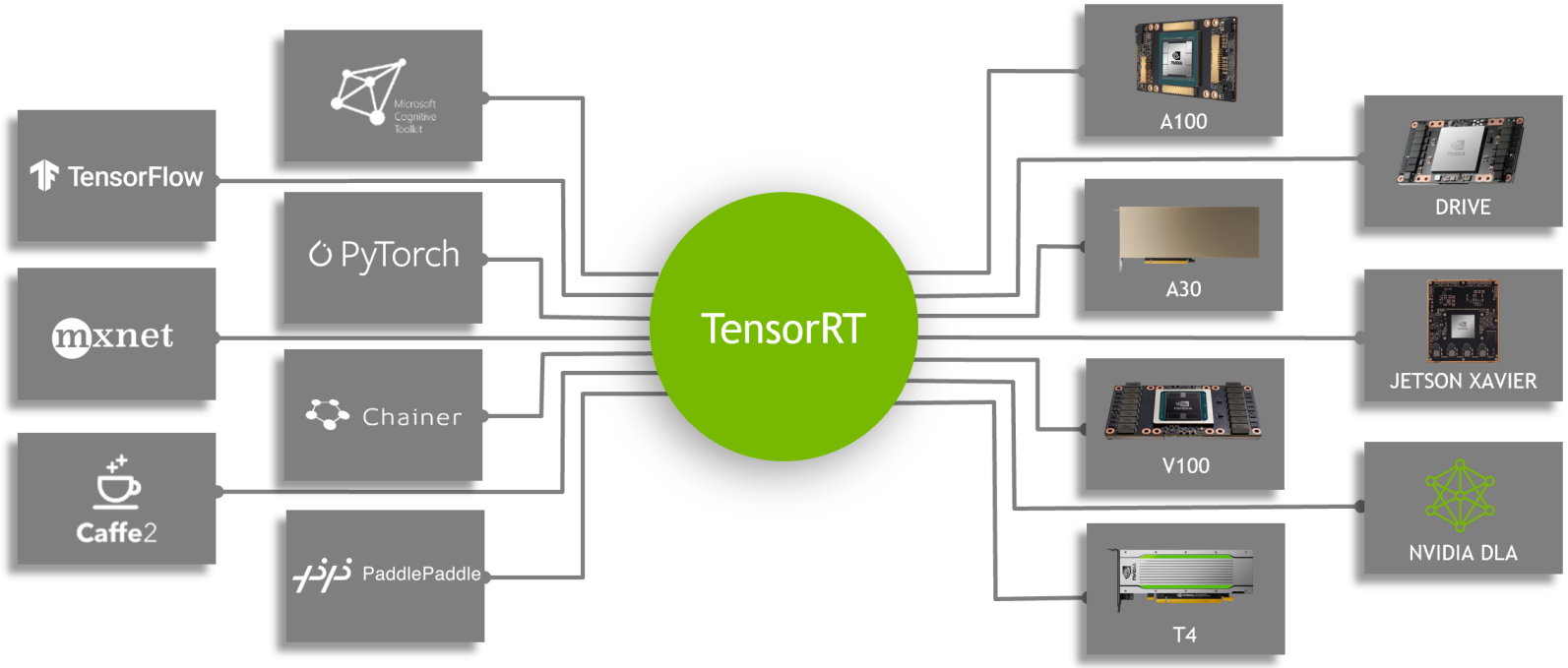
python models/export.py --weights yolov5s.pt --img 640 --batch 1

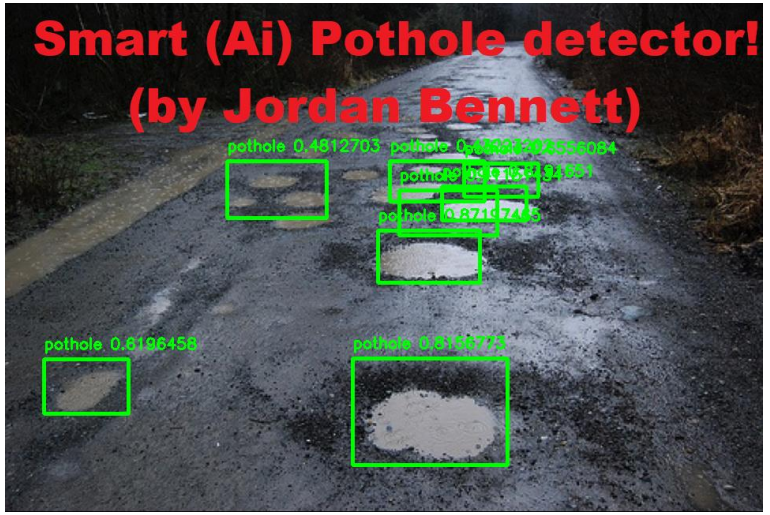
Copyright FAST CAMPUS Corp. All Rights Reserved

YOLO in practice – TensorRT

Fast campus

https://github.com/wang-xinyu/tensorrtx/tree/master/yolov5





https://www.codeproject.com/Articles/5293077/Converting-YOLOv5-PyTorch-Model-Weights-to-TensorF

https://www.seeedstudio.com/blog/2022/08/23/faster-inference-with-tensorrt-on-nvidia-jetson-run-yolov5-at-27-fps-on-jetson-nano/

Copyright FAST CAMPUS Corp. All Rights Reserved

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

AI Vision Limitation and Future

AI – limitations

Copyright FAST CAMPUS Corp. All Rights Reserved

AI Vision Limitation and Future

There are three main challenges that AI research must address today:

1. How can machines learn to represent the world, learn to predict, and learn to act largely by observation?
   Interactions in the real world are expensive and dangerous, intelligent agents should learn as much as they can about the world without interaction (by observation) so as to minimize the number of expensive and dangerous trials necessary to learn a particular task.

2. How can machine reason and plan in ways that are compatible with gradient-based learning?
   Our best approaches to learning rely on estimating and using the gradient of a loss, which can only be performed with differentiable architectures and is difficult to reconcile with logic-based symbolic reasoning.

3. How can machines learn to represent percepts and action plans in a hierarchical manner, at multiple levels of abstraction, and multiple time scales?
   Humans and many animals are able to conceive multilevel abstractions with which long-term predictions and long-term planning can be performed by decomposing complex actions into sequences of lower-level ones.

Yann LeCun, 2022.6, A Path Towards Autonomous Machine Intelligence

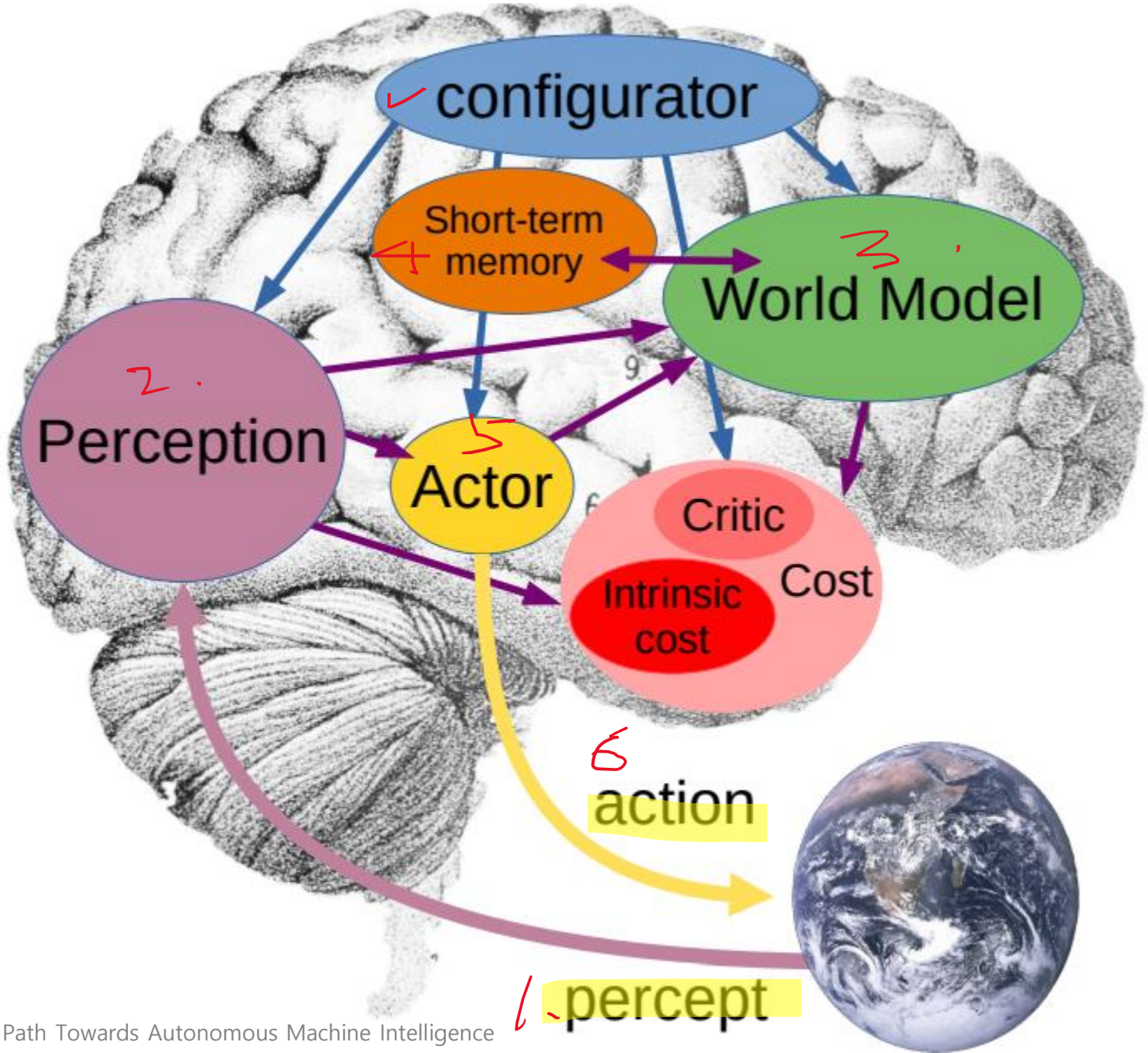
Copyright FAST CAMPUS Corp. All Rights Reserved

AI Vision Limitation and Future

When will my baby recognize me?
Darienne Hosley Stewart, babycener(www.babycenter.com/baby/behavior/when-will-my-baby-recognize-me_1368483)

Copyright **FAST CAMPUS** Corp. All Rights Reserved

AI Vision Limitation and Future



Yann LeCun, 2022.6, A Path Towards Autonomous Machine Intelligence

Copyright FAST CAMPUS Corp. All Rights Reserved

AI Vision Limitation and Future

The proposed architecture for autonomous intelligent agents is depicted in Figure 2.

It is composed of a number of modules whose functions are described below. Some of the modules are configurable on the fly, i.e. their precise function is determined by the configurator module. The role of the configurator is executive control: given a task to be executed, it pre-configures the perception, the world model, the cost and the actor for the task at hand. The configurator modulates the parameters of the modules it feeds into.

The configurator module takes input from all other modules and configures them for the task at hand by modulating their parameters and their attention circuits. In particular, the configurator may prime the perception, world model, and cost modules to fulfill a particular goal.

The perception module receives signals from sensors and estimates the current state of the world. For a given task, only a small subset of the perceived state of the world is relevant and useful. The perception module may represent the state of the world in a hierarchical fashion, with multiple levels of abstraction. The configurator primes the perception system to extract the relevant information from the percept for the task at hand.

The world model module constitutes the most complex piece of the architecture. Its role is twofold: (1) estimate missing information about the state of the world not provided by perception, (2) predict plausible future states of the world. The world model may predict natural evolutions of the world, or may predict future world states resulting from a sequence of actions proposed by the actor module. The world model may predict multiple plausible

Copyright FAST CAMPUS Corp. All Rights Reserved

AI Vision Limitation and Future

Fast campus

The cost module measures the level of "discomfort" of the agent, in the form of a scalar quantity called the *energy*. The energy is the sum of two energy terms computed by two sub-modules: the Intrinsic Cost module and the Trainable Critic module. The overall objective of the agent is to take actions so as to remain in states that minimize the average energy.

The *Intrinsic Cost module* is hard-wired (immutable, non trainable) and computes a single scalar, the *intrinsic energy* that measures the instantaneous "discomfort" of the agent – think pain (high intrinsic energy), pleasure (low or negative intrinsic energy), hunger, etc. The input to the module is the current state of the world, produced by the perception module, or potential future states predicted by the world model. *The ultimate goal of the agent is minimize the intrinsic cost over the long run.* This is where basic behavioral drives

The *Trainable Critic module* predicts an estimate of future intrinsic energies. Like the intrinsic cost, its input is either the current state of the world or possible states predicted by the world model. For training, the critic retrieves past states and subsequent intrinsic costs stored in the associative memory module, and trains itself to predict the latter from the former. The function of the critic module can be dynamically configured by the configurator to direct the system towards a particular sub-goal, as part of a bigger task.

Because both sub-modules of the cost module are differentiable, the gradient of the energy can be back-propagated through the other modules, particularly the world model, the actor and the perception, for planning, reasoning, and learning.

Copyright FAST CAMPUS Corp. All Rights Reserved

AI Vision Limitation and Future

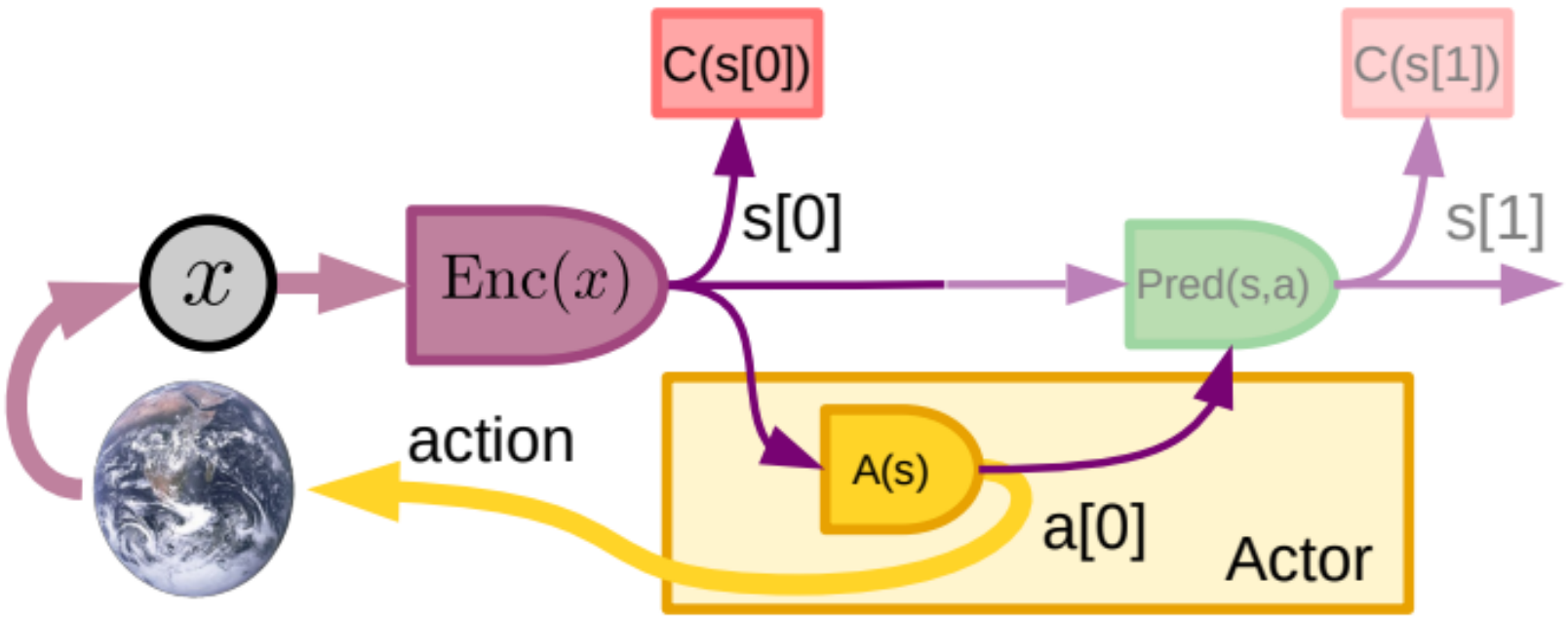
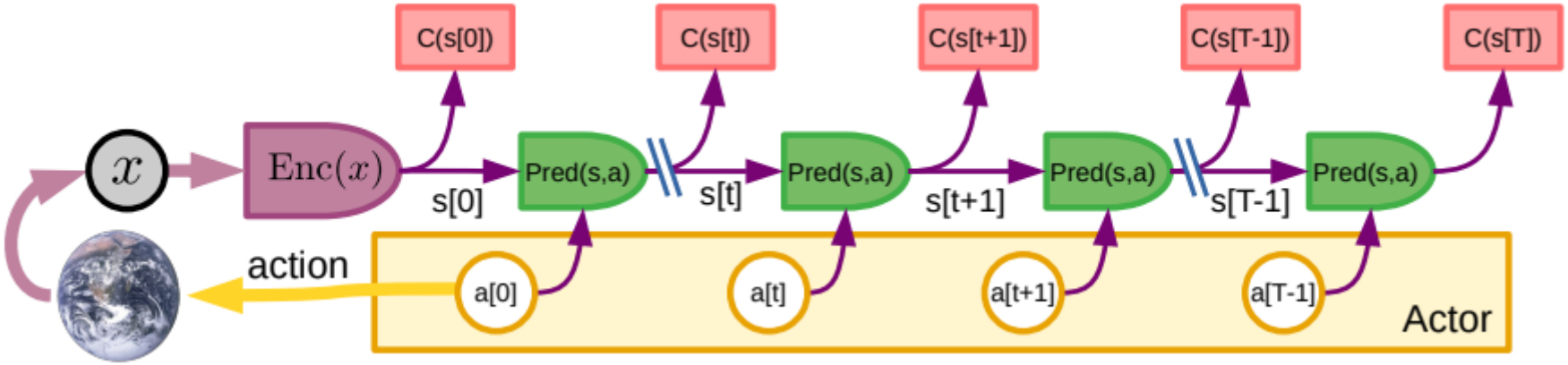


Figure 3: *Mode-1 perception-action episode. The perception module estimates the state of the world* $s[0] = \mathrm{Enc}(x)$. *The actor directly computes an action, or a short sequence of actions, through a policy module* $a[0] = A(s[0])$.
This reactive process does not make use of the world model nor of the cost. The cost module computes the energy of the initial state $f[0] = \mathrm{C}(s[0])$ *and stores the pairs* $(s[0], f[0])$ *in the short-term memory. Optionally, it may also predict the next state using the world model* $s[1] = \mathrm{Pred}(s[0], a[0])$, *and the associated energy* $f[0] = \mathrm{C}(s[0])$ *so that the world model can be adjusted once the next observation resulting from the action taken becomes available.*

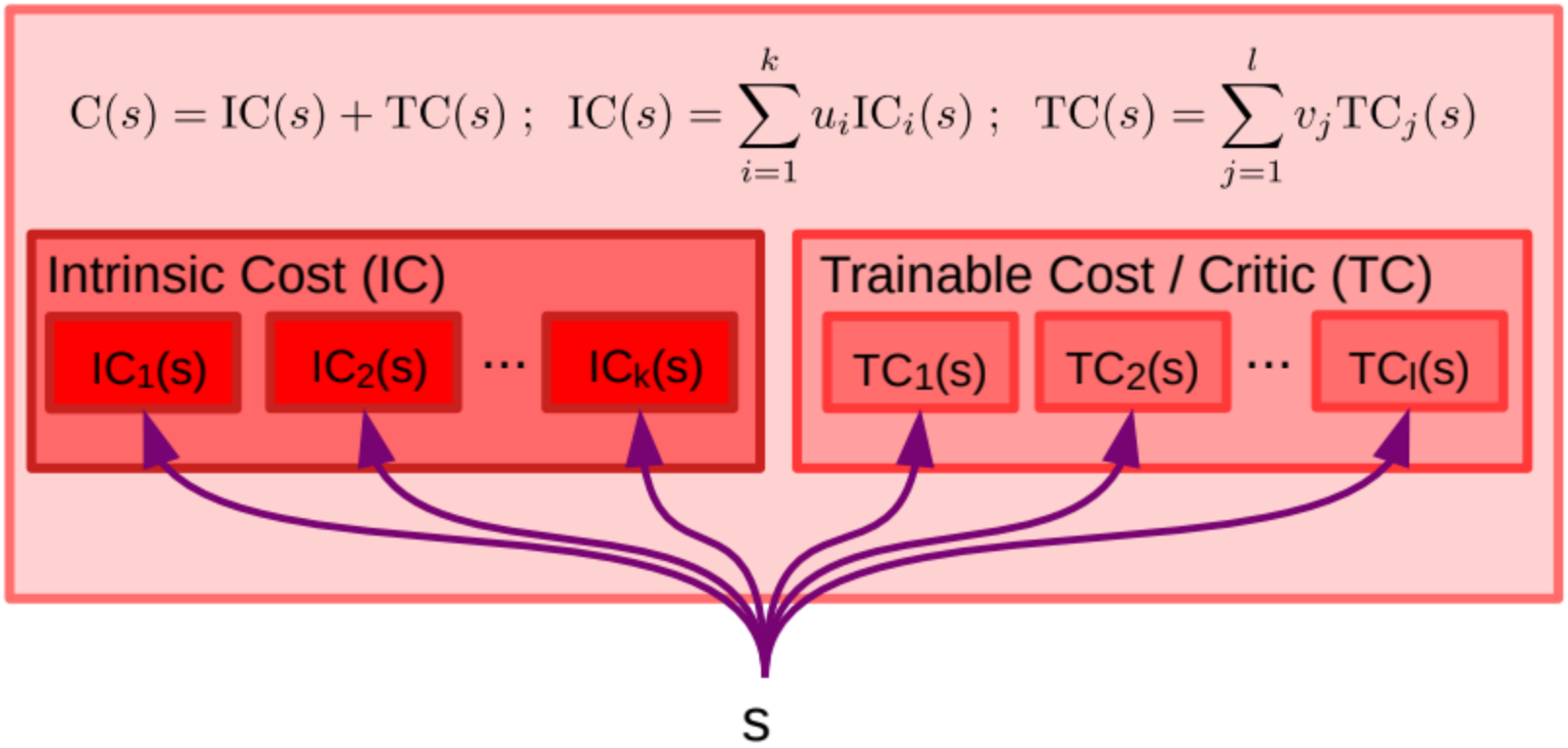
Copyright FAST CAMPUS Corp. All Rights Reserved

AI Vision Limitation and Future



Copyright FAST CAMPUS Corp. All Rights Reserved

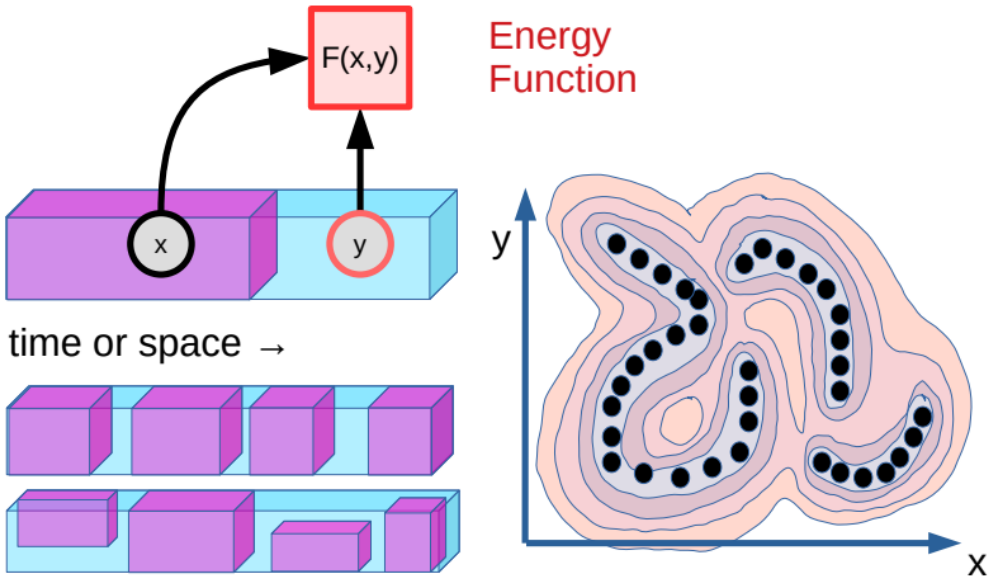
AI Vision Limitation and Future



Copyright FAST CAMPUS Corp. All Rights Reserved

AI Vision Limitation and Future

A general formulation can be done with the framework of *Energy-Based Models* (EBM). The system is a scalar-valued function $F(x, y)$ that produces low energy values when x and y are compatible and higher values when they are not. The concept is depicted in Figure 8. Data points are black dots. The energy function produces low energy values around the data points, and higher energies away from the regions of high data density, as symbolized by the contour lines of the energy landscape. The EBM implicit function formulation enables the system to represent multi-modal dependencies in which multiple values of y are compatible with a given x. The set of y compatible with a given x may be a single point, multiple discrete points, a manifold, or a collection of points and manifolds.

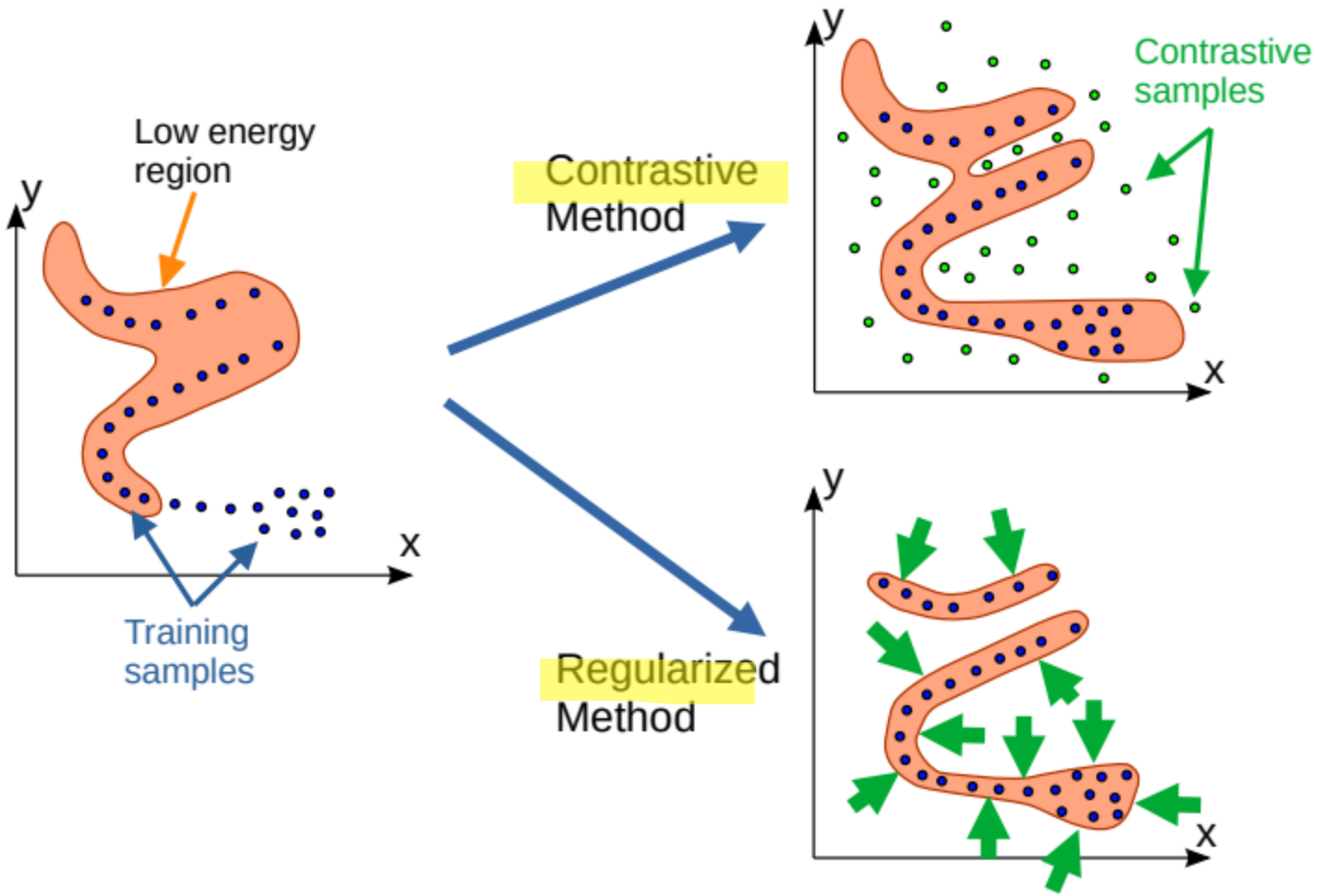


Copyright FAST CAMPUS Corp. All Rights Reserved

AI Vision Limitation and Future

This latent-variable inference by minimization allows us to eliminate z from the energy function:

$$F_w(x, y) = \min_{z \in \mathcal{Z}} E_w(x, y, z) = E_w(x, y, \check{z}) \tag{5}$$



Copyright FAST CAMPUS Corp. All Rights Reserved

AI Vision Limitation and Future

- the world is deterministic but chaotic, hence difficult to predict without infinitely precise perception (aleatoric uncertainty, type 2)
- the world is deterministic but partially observable (aleatoric uncertainty type 3).
- the world is fully observable, but the sensors only give partial information about the world state (epistemic uncertainty, type 1)
- the representation of the world state extracted by the perception module does not contain the full information necessary for accurate prediction (epistemic uncertainty, type 2).
- the world model is inaccurate due to limitations of its representational power (bounded rationality or epistemic uncertainty, type 3).

Acknowledgments

Ideas in this paper are a distillation of years of interactions with many people. It is impossible to list them all here.

I will list some of the more salient ones: Léon Bottou, Yoshua Bengio, Geoffrey Hinton, Rob Fergus, and Jitendra Malick; Emmanuel Dupoux and Stanislas Dehaene; Mikael Henaff,

Copyright FAST CAMPUS Corp. All Rights Reserved

AI Vision Limitation and Future

Next AI

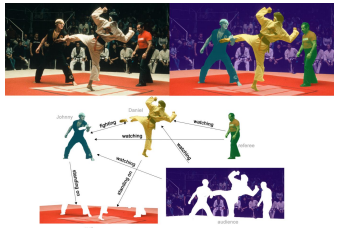
More AI applications.
https://www.aitech.vision/

Human-Centered AI (HCAI)
https://research.ibm.com/blog/what-is-human-centered-ai

Artificial general intelligence (AGI)

abstract thinking
background knowledge
common sense
cause and effect
transfer learning

Vision reasoning



GCN (Graph Convolutional Network)

Vision with robotics

Learning without supervision

Copyright FAST CAMPUS Corp. All Rights Reserved

AI

Copyright **FAST CAMPUS** Corp. All Rights Reserved

fast campus Copyright FASTCAMPUS Corp.All Rights Reserved

Thanks

YOLO5 in practice using Jupyter Notebook, vscode

Fast campus

Utf-8 Bug fix

```
def open(self, filename):
    """Open a new file named 'filename'.  This overrides both the
       'filename' and 'file' arguments to the constructor."""
    self.filename = filename
    self.file = io.open(self.filename, 'r',encoding='utf-8', errors=self.errors)
    self.current_line = 0
```

```
        read_ok = []
        for filename in filenames:
            try:
                with open(filename, encoding='utf-8') as fp:#encoding=encoding
                    self._read(fp, filename)
            except OSError:
```

Copyright FAST CAMPUS Corp. All Rights Reserved