2020

01 SATURDAY | DAY (214 - 152) / WEEK - 31 ST |

AUGUST 2020

Schedule

Log Book.

08:00 NOTE:

09:00 The log book for autumn semester

10:00 has been rewritten again as I left my

11:00 previous one in my dorm.

12:00 Kindly visit ~~via~~ github.com/vicky-mL.

13:00 I have curated notes on how to

14:00 set up the platforms along with

15:00 getting started with CNN & source codes.

16:00

17:00

18:00 02 SUNDAY

19:00

20:00

21:00

Eve.

2020

AUGUST 2020

| DAY (216 - 150) / WEEK - 32 ND | MONDAY 03

Schedule September.

08:00 Objectives of this month.

09:00 * Talk with the superwisor to know how to improve my project which I did during

10:00 Summer Research.

11:00 * Talk with Mr. Wayne San, visiting teacher of UWC, to request FPGA Board.

12:00

13:00 - Arrived campus at 4th week of Sept.

14:00 - Recieved the Xilinx Ultrascale + ZCU102

15:00 FPGA Board.

16:00 - Read the User Manual of the board.

17:00

18:00

19:00

20:00

21:00

Eve.

SEPTEMBER

OCTOBER

NOVEMBER

DECEMBER

2020

04 TUESDAY [DAY (217 - 149) / WEEK - 32 ND]

AUGUST 2020

Week	M	T	W	T	F	S	S
31	27					1	2
32	3	4	5	6	7	8	9
33	10	11	12	13	14	15	16
34	17	18	19	20	21	22	23
35	24	25	26	27	28	29	30

Schedule

October.

08:00 Objectives of this month.

09:00 * Talk with supervisor to finalize my project outcome

10:00 * Get used with FPGA Boards.

11:00 * Demo Yolo object detection.

12:00

13:00 Setting UP FPGA.

14:00

- Need to buy &

15:00 Power Cable ✓

16:00 - USB cables ✓

LAN Cable ✓

17:00

18:00

19:00 Need to setup Linux on personal

20:00 laptop. ✓

21:00

Eve.

Education is the movement from darkness to light. - Allan Bloom

AUGUST 2020

[DAY (218 - 148) / WEEK - 32 ND] WEDNESDAY 05

Schedule

08:00 Vitis-AI package installed from Xilinx official website.

09:00 To be Read.

10:00 * Xilinx ZCU102 User Manual

* Vitis AI Getting Started manual.

11:00

12:00 - checked Yolo source code.

13:00 tried various source codes from

14:00 official site of Darknet pjreddie.com

15:00 and other github repositories.

Installed various required dependencies.

16:00

17:00

18:00 Had many problems with linux

Installations.

19:00

- Anaconda environment failed to

20:00 install at the beginning. Also tensorflow

21:00 Was trying to run source codes.

Eve. directly on linux platform.

Education is the transmission of civilization. - Ariel and Will Durant

SEPTEMBER

OCTOBER

NOVEMBER

DECEMBER

2020

06 THURSDAY [DAY (219 - 147) / WEEK - 32 ND]

AUGUST 2020

Week	M	T	W	T	F	S	S
31	31					1	2
32	3	4	5	6	7	8	9
33	10	11	12	13	14	15	16
34	17	18	19	20	21	22	23
35	24	25	26	27	28	29	30

Schedule

08:00 Rised an issue on stackoverflow developer's forums and got replies.

09:00 Remedy

10:00 - Update the linux kernel by changing the location to US -general.

11:00

12:00 - Delete Anaconda if previously installed
- Install Anaconda 3.7 version alone. As newer version didnt have CUDA support.

13:00 - Install tf by creating a seperate env in anaconda itself. → pip install tensorflow

14:00

15:00

16:00

17:00

18:00

19:00

20:00

21:00 Tasks Achieved in October.

Eve. - Read User Manuals. ✓

- Set up Linux on personal laptop ✓

- Set up FPGA x (Didnt recieve ~~comp~~)

Even if you're on the right track, you get run over if you just sit there. - Will Rogers

AUGUST 2020

[DAY (220 - 146) / WEEK - 32 ND] FRIDAY 07

Schedule

08:00 November:-

09:00 Objectives of this month:

10:00 * Set up FPGA.

11:00 * Demo YOLO.
* Meet the Morderator.

12:00 The power cable has to be imported from Europe. So the seller in Taobao told it would take 21 days to deliver. So the task for setting up the FPGA is ~~passed~~ paused until next month.

13:00

14:00

15:00

16:00

17:00

18:00

19:00 - Followed the tutorial to run

20:00 Yolo on personal laptop.

21:00 A new environment was

Eve. created on Anaconda.

~~sudo~~ sudo activate tf1.

Every improvement in communication makes the bore more terrible. - Frank Moore Colby

SEPTEMBER
OCTOBER
NOVEMBER
DECEMBER

2020

08 SATURDAY

(DAY 221 - 145 / WEEK - 32 NO)

AUGUST 2020

Schedule

This env contains all dependencies
09:00 to run YOLO.

- CUDA 7.0 +
- Tensorflow.
- Caffe.
- OpenCV.
- Numpy.

- A demonstration of the realtime simulation of YOLO has been performed.
- It works at < 20 FPS and lags a lot.
- We used pretrained weight file called ~~weights.cfg~~. yolov3.cfg
yolov3.weights.
- Overall, we ~~have~~ achieved the task for this month.

09 SUNDAY

Everybody thinks of changing the world, but no one thinks of changing himself. - Leo Tolstoy

AUGUST 2020

(DAY 223 - 143 / WEEK - 33 RO) MONDAY 10

Schedule

I was trying to train my own
08:00 weight file with new classes.
To train our own class, the following
09:00 has to be done.
* Gather the data
* Label the images using labelling software.
* Store the annotated data in JSON format
/other normal .txt also works.

To use this ~~soft~~ software.
Download by : pip install labelImg.
Open the software. - Browse to the path
where the gathered Img data is -
Open each image and annotate by simply
click and Drag. - Define the classes.
- Ctrl + s to save the label.

This is very time consuming.
So I switched to an online
portal called Supervisely.

SEPTEMBER
OCTOBER
NOVEMBER
DECEMBER

Everyday living is life lessons. - Allen Zimama

Schedule

Supervisely - An online portal to annotate and train weights using State-of-the-Art CNN models. They have attractive platform.

I initially wanted to train for detecting traffic lights colour. They had gathered dataset for traffic light detection.

To do that.

Create an Account.

Create new project.

Download their dataset or can import the custom dataset.

A new windows opens with Annotation tools.

Create all the classes.

Choose the image and annotate by click & drag.

then choose its class and press enter.

Eve

Experience is what you get when you don't get what you want. - Don Stanford

Schedule

Make sure to keep some samples for validation. The software also has options to define which sample to be treated as validation & training.

Once done we can view the statistics to get user friendly info on our dataset.

Now this dataset can be fed into any existing models. No coding needed. The only thing that has to be changed is the Number of classes and another hyperparameter in the source code.

We can also define the threshold, epochs, batch size.

Eve

Flattery is like cologne water, to be smelt of, not swallowed. - Josh Billings

2020

13 THURSDAY | DAY (226 - 140) / WEEK - 33 RD |

AUGUST 2020

Schedule

08:00 This model can be trained using paid cloud platform.

09:00 But I tried to use my laptop

10:00 hardware for training. This is where

11:00 I got stuck.

12:00 The platform has to be connected

13:00 to a linux based personal computer

14:00 using Docker with Nvidia Docker

15:00 enabled. I had difficulties in installing

16:00 those docker environments in my

17:00 personal laptop.

18:00 As time is very short and

19:00 annotation of dataset might have to

20:00 be added more for training accuracy.

21:00 I gave up with this idea of

Eve training custom data. Moreover,

the PC cant handle training. ~~for~~

AI.

For every obstacle there is a solution over under, around or through. - Dan Zadra

AUGUST 2020

FRIDAY 14 | DAY (227 - 139) / WEEK - 33 RD |

Schedule

08:00 As the use of pretrained weights

09:00 can meet the primary goal of the

10:00 project, the plan for training custom

11:00 dataset has bee withdrawn.

12:00 later in November.

13:00 - Bought FPGA cables

- Power supply yet to be delivered.

14:00 - Made myself familier with

15:00 linux platform & Vitis AI (for FPGA)

16:00 working directory.

17:00

18:00 Tasks achieved on November.

19:00 * Demo of YOLO on Laptop ✓

20:00 * Set up FPGA ×

21:00 * Meet the Moderator ✓.

Eve Met the moderator to talk about project outline. Gathered suggestions and advise on how to improve the outcome of the project.

Friendship makes prosperity more shining and lessens adversity by dividing and sharing it. - Cicero

SEPTEMBER OCTOBER NOVEMBER DECEMBER

2020

15 SATURDAY [DAY (228 - 138) / WEEK - 33 RD]

AUGUST 2020

Week	M	T	W	T	F	S	S
31	31					1	2
32	3	4	5	6	7	8	9
33	10	11	12	13	14	15	16
34	17	18	19	20	21	22	23
35	24	25	26	27	28	29	30

Schedule

Suggestion given by Moderator:

08:00 * Have a clear target . . .

09:00 * Improvement on Project Outline

10:00 Report.

11:00 * Have a concise ghantt chart

12:00 to better reaching the goals.

13:00 * Make sure that the project outcome

14:00 is valuable and notable.

15:00

16:00

17:00

18:00 16 SUNDAY

19:00

20:00

21:00

Eve.

Get not your friends by bare compliments, but by giving them sensible tokens of your love. - Socrates

2020

AUGUST 2020

[DAY (230 - 136) / WEEK - 34 TH] MONDAY 17

Schedule

December

08:00 Objectives of this month.

09:00 * Set up the FPGA

10:00 * Read AI Algorithms.

11:00 * Talk to Supervisor about Progress.

12:00

13:00 Recived the power supply cable.

14:00

15:00 Downloaded the boot image and burned

16:00 it into SD card for booting up the

17:00 FPGA.

18:00 Next:-Get Access to FPGA Directory.
-Extract Vitis AI into the FPGA Directo

19:00

20:00 The User manual did not cover

21:00 in detail on how to connect FPGA with the laptop via ssh port.

Eve. I am totally unaware of ssh via LAN & USB cable.

SEPTEMBER OCTOBER NOVEMBER DECEMBER

Good fences make good neighbors. - Robert Frost

2020

18 TUESDAY [DAY (231 - 135) / WEEK - 34 TH]

AUGUST 2020

Schedule

I raised a threat in Xilinx Forum on how to connect via LAN cable.

The threat was answered after 1 week.

Things to be done to connect:

* Download Putty, a SSH, UART, SCP connection terminal.

* Select the USB port where the FPGA is connected with laptop.

* The terminal can now show the FPGA directory.

I got stuck again, I didn't know how to copy files from computer via SSH LAN cable.

Eve.

Great things are not done by impulse, but by a series of small things brought together. - Vincent Van Gogh

2020

AUGUST 2020

[DAY (232 - 134) / WEEK - 34 TH] WEDNESDAY 19

Schedule

I raised another threat in Xilinx forums. - got answered in a week.

Things to be done.

On FPGA PUTTY terminal.

sudo ipconfig eth0 192.168.1.100.

On PC terminal.

sudo ipconfig eth0 192.168.1.101

We need to make sure that the Ip address is in this format. and Subnet IP has to be same.

Verify by typing "ping 192.168.1.100" on linux PC terminal.

"sudo scp -r /opt/vitis_ai/xilinx_zcu_board_package root@192.168.0.100:~/"

After transferring go to that directory in Putty terminal and run sh install.sh.

Happiness is not a matter of intensity, but of balance and order and rhythm and harmony. - Thomas Merton

SEPTEMBER OCTOBER NOVEMBER DECEMBER

2020

20 THURSDAY | DAY (233 - 132) / WEEK - 34 TH |

AUGUST 2020

Schedule

Here we met a problem while installing the package. It showed an error. I raised a threat regain in xilinx Forums.

Week ~~2nd~~ 3rd week - Jan 1st week. concentrated on Exams.

Raised multiple threats in forums in the meanwhile and get help from developers.

Happiness lies in the job of achievement and the thrill of creative effort. - Franklin D. Roosevelt

AUGUST 2020

| DAY (234 - 1321 / WEEK - 34 TH | FRIDAY 21

Schedule

January

Objectives of this month:

* set up FPGA
* talk with supervisor if the issue persist.

Trying Reistalled the FPGA boot image with different versions.

But none of it booted up the board. Only one particular Boot image worked. - But did not install properly the package.

Planned for a trip to HK and Macau late January and left china on 22nd.

Initial plan was to come back and continue the project. But due to corona outbreak, returned back to my home country.

He who praises you for what you lack, wishes to take from you what you have. - Don Juan Manuel

SEPTEMBER
OCTOBER
NOVEMBER
DECEMBER

2020

22 SATURDAY [DAY (295 - 131) / WEEK - 34 TH]

AUGUST 2020

Schedule

* Tasks achieved in January

08:00 * Set up board X

09:00

The plan for using the FPGA board has been given up.

10:00

11:00

12:00

13:00

14:00

15:00

16:00

17:00

18:00

23 SUNDAY

19:00

20:00

21:00

Eve.

2020

AUGUST 2020

[DAY (237 - 129) / WEEK - 35 TH] MONDAY 24

Schedule

Febuary

Tasks to be done this month

08:00

09:00 * Talk with superv to alter the objectives.

10:00 * Read AI Vision Related Paper.

11:00

12:00

A detailed discussion has been made with the superwisor on what is achieved for now.

14:00

15:00 The superwisor suggested to look for alternatives.

16:00

17:00

18:00 I have read a series of

19:00 research paper and have curated

20:00 the list of paper that I read on

21:00 my github repo : github.com/vicky-ml

Eve.

SEPTEMBER

OCTOBER

NOVEMBER

DECEMBER

2020

25 TUESDAY | DAY (238 - 128) / WEEK - 35 TH |

AUGUST 2020

Schedule

Articles Refered:

- Annupaanuswamy.com/yolo-object-detection-tutorial
- medium.com/yolo-object-detection-algorithm
- medium.com/real-time-object-detection-using-state-of-the-art
- towardsdatascience.com/yolov3-various-state-of-the-art-object-detection-
- cv-tricks.com/object-detection-2019
- medium.com/seiforce/ai-hardware-and-the-battle-for-more-compact-power
- towardsdatascience.com/ai-accelerator-products

...

"I am prepared to go anywhere, provided it be forward." - David Livingstone

AUGUST 2020

2020

[DAY (239 - 127) / WEEK - 35 TH] WEDNESDAY 26

Schedule

Yolov3 source code was analysed. Could'nt ~~stole~~ run it on current laptop as I left my laptop in China. The backup laptop in my house does not have GPU in it.

Tasks Achieved this month.

- Made case study on Object detection Algorithm. ✓
- & Hardware.

SEPTEMBER
OCTOBER
NOVEMBER
DECEMBER

"I went into the business for the money, and the art grew out of it." - Charlie Chaplin

2020

27 THURSDAY [DAY (240 - 126) / WEEK - 35 TH]

AUGUST 2020

Schedule

March

08:00

Things to be done:

09:00 * Find cloud Platform that provide AI Accelerated hardware.

10:00 * Do more case study.

11:00

12:00

13:00 In order to meet the requirements to complete all the given Objectives of the Proj

14:00 I checked out for cloud platforms that provided FPGA-As-A-Service (FAAS) platforms

15:00 - Available Platform that provides Xilinx FAAS.

16:00 * AWS * Plurify.

17:00 * Nimbix

18:00

19:00

20:00

21:00 I tried to perform the task with AWS first. Again I had difficulty in understanding the cloud architecture to set up instances. Got stuck when

Eve.

I will not say I've failed a 1000 times; I'll say I found a 1000 ways that can cause failure. - Thomas Edison

2020

AUGUST 2020

[DAY (241 - 125) / WEEK - 35 TH] FRIDAY 28

Schedule

08:00 trying to connect with the AWS instance from home PC to access the Xilinx FPGA working directory.

09:00 Plurify was not free at that time.

10:00

11:00

12:00 So I chose Nimbix for following reasons -

13:00 - Free for First 200 hours.

14:00 - Easy to create new instance

15:00 - Easy to Access via their VM.

16:00

17:00 Was getting used with Nimbix portal throughout April.

18:00

19:00 Meanwhile, I came across learning to create our own CNN architecture for training the Neural Networks using popular frameworks

20:00 • Tensorflow • Caffe • MXNet • pytorch.

21:00

Eve.

I will pay more for the ability to deal with people than for any other ability under the sun. - John D. Rockfeller

SEPTEMBER OCTOBER NOVEMBER DECEMBER

2020

29 SATURDAY [DAY (242 - 124) / WEEK - 35 TH]

AUGUST 2020

Schedule

Tasks Achieved in [redacted] March.

08:00

* Found a promisable alternative for metting 100% of given objective.

09:00

* Successfully set up a working platform on Nimbix

10:00

* Found ways to provide extended targets as outcome of this project.

11:00

12:00

13:00

14:00

15:00

16:00

17:00

18:00 30 SUNDAY

19:00

20:00

21:00

Eve.

If we do not find anything very pleasant, at least we shall find something new. - Voltaire

AUGUST 2020

2020

[DAY (244 - 122) / WEEK - 36 TH] MONDAY 31

Schedule

April

08:00 Objectives of this Month:

09:00

10:00 • Demo the AI vision algorithm using

11:00 pre-designed source codes in Vitis-AI.

12:00 • Consult with the supervisor about

13:00 tasks accomplished and future goals.

14:00 • Gather Reasearch papers and do more

15:00 case studies to start with thesis writing.

16:00

Using the nimbix HPC I was able to

18:00 run simulations for ResNet-50 &

19:00 GoogLeNet in different quatized data

20:00 types and throughput.

21:00 The portal gave some version

Eve. error when tried to [redacted] install

and Run yolo model from the

Predesigned model.

If you trust before you try, you may repent before you die. - Nathan Bailey

SEPTEMBER

OCTOBER

NOVEMBER

DECEMBER

2020

01 TUESDAY [DAY (245 - 121) / WEEK - 36 TH]

Schedule

Provided a Detailed outcome so far on what was achieved to the Supervisor.

- Upon Supervisor's advice, I started writing the thesis.

Meanwhile, the architecture of various CNN models were learnt and was trying to demonstrate it by training on Google Collab platform.

Framework used: MxNet.

↓ Easy Implementation of CNN layers in just one line of code for each batch of filters.

A code sample was used to make our own CNN models.

In the land of the blind, the one-eyed man is stoned to death. - Joan D. Vinge

SEPTEMBER 2020

2020

[DAY (246 - 120) / WEEK - 36 TH] WEDNESDAY 02

Schedule

The following CNN models were simulated and trained using Fashion-MNIST Dataset (Just 2 classes)

A detailed link to the resources and dependencies used in this project has been given & maintained in my github

Repo: github.com/vicky-ml

trained at Epochs: 10; Batches: 64; l_r: 0.01

1) AlexNet
   - 18 layers in total.
   - Loss: 0.28
   - train Acc: 0.899.

2) NiN:
   - 15 layers in total.
   - Loss: 0.584
   - train Acc: 0.781

OCTOBER NOVEMBER DECEMBER

It is dangerous to be sincere unless you are also stupid. - George Bernard Shaw

Schedule

08:00

VGG11:

- 11 layers in total.
- Loss : 0.172.
- train Acc : 0.937.

09:00

10:00

11:00

12:00

GoogLeNet:

- 8 layers + 3 parallel batches in total.
- Loss : 0.330
- train Acc : 0.876.

13:00

14:00

15:00

16:00

17:00

18:00

ResNet-18.

- 18 layers in total.
- Loss : 0.026
- train Acc : 0.992.

19:00

20:00

21:00

Eve

It is impossible to walk rapidly and be unhappy. - Howard Murphy

Schedule

08:00

Yolo :

Was able to run only inference. Was not able to train because of not having CUDA enabled hardware.

- Provides Image in 19.907 ms on google Collab.

09:00

10:00

11:00

12:00

13:00

14:00

15:00

16:00

17:00

18:00

19:00

20:00

21:00

Eve

It is possible to fail in many ways while to succeed is possible only in one way - Aristotle

05 SATURDAY [DAY (249 - 117) / WEEK - 36 TH]

Schedule

FPGA Demo.

08:00

09:00 To be done before Simulation.

10:00 * Set working directory.

11:00 * Git clone the xilinx official repo

12:00 * Create an exclusive environment.
- can use existing env: conda activate ml-tut

13:00

14:00 * Install the packages. - Use the

15:00 tutorial from Desktop in the nimbix VM

16:00 * Now the AI Algorithms can be

17:00 run by using typical linux commands.

18:00 * Need to maintain the Hardware

19:00 and other hardware configuration.

20:00

21:00 We have used ALVEO U-200

Eve. FPGA Boards for demo. Other

latest model from Xilinx called

ALVEO-U50 can also be used.

Justice being violated destroys justice being preserved preserves. - Manusmriti

SEPTEMBER 2020

2020

[DAY (251 - 115) / WEEK - 37 TH] MONDAY 07

Schedule

08:00 We have provided Yolo V3 on

09:00 ALVEO U50.

10:00 The previous version of, Xilinx Alveo

11:00 U200 showed version and dependencies

12:00 error when trying to install.

13:00 All the outputs were gathered.

14:00 and curated in a user friendly manner

15:00 on my github repo.

16:00

17:00 Tasks Completed this month:

18:00 * FPGA Simulations ✓

* other CNN models simulation ✓

19:00 * Case Study on AI Hardware Acc ✓

20:00 * Thesis Writing - Lit Review ✓

21:00 - Background ✓

Eve.

Laughing is the sensation of feeling good all over and showing it principally in one spot. - Josh Billings

OCTOBER NOVEMBER DECEMBER

COMPUTER VISION

Basics

COMPUTER VISION

- At the end of the class, you will be familier with basic CNN architecture
- Understand how computer persieve and interpret images
- Able to deploy YOLO on your PC

Pre-requisite:

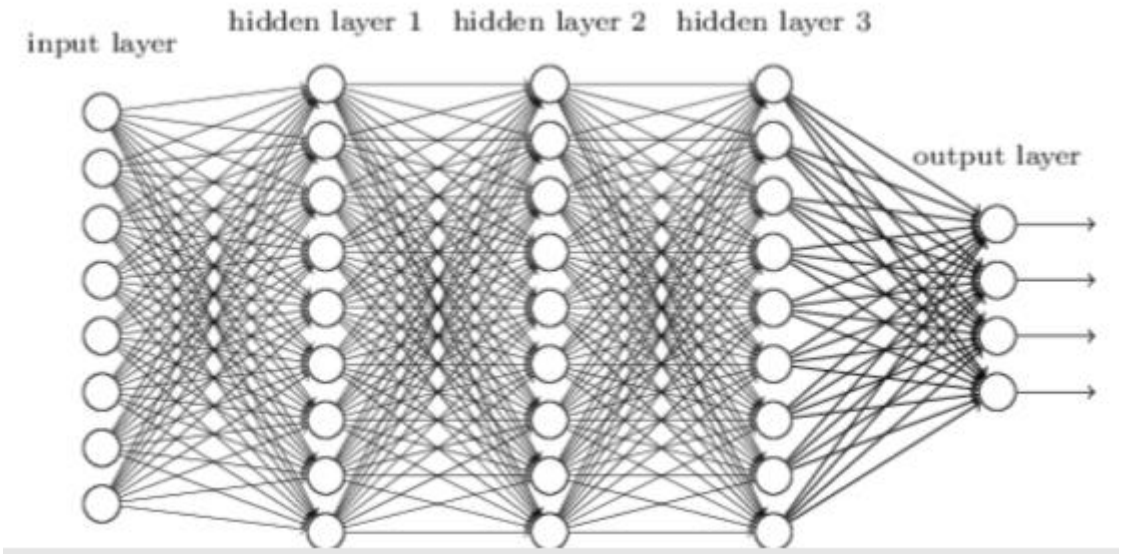
- You are expected to know
- Basic coding algorithm flow
- Linear Algebra
- Comfortable with compiling and executing codes from github

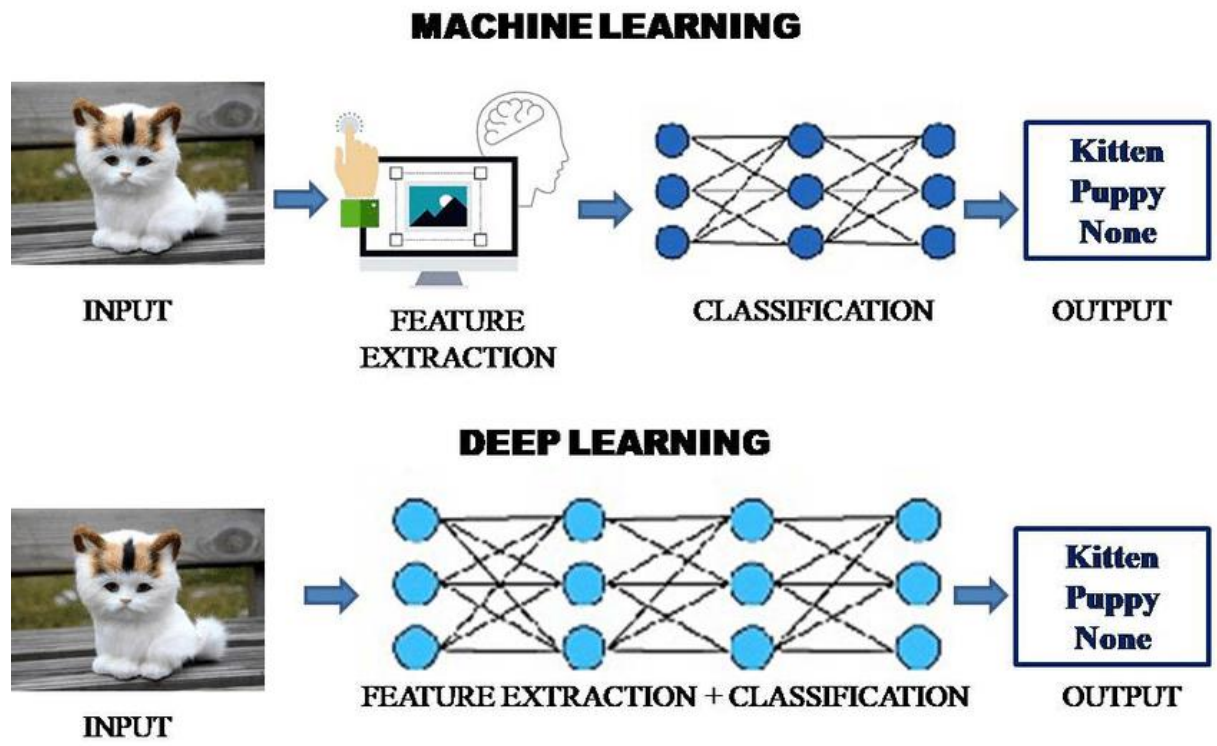
WHAT IS DEEP LEARNING?

Most machine learning methods work well because of human-designed input features or representations – Our job is to find thebestfeatures to send to learning techniques such as SVM.

- Deep learning algorithms learn multiple levels of representations (here: hiddenlayer1,2,3) and an output layer
- From “raw” inputs x (e.g. sound, pixels, characters, or words)
- Neural networksare the currently successful method for deep learning

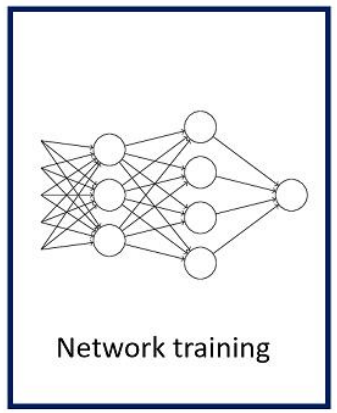
More deeper models results in more accurate representation of features and better classification




MACHINE LEARNING
Kitten
Puppy
None
INPUT
FEATURE EXTRACTION
CLASSIFICATION
OUTPUT
DEEP LEARNING
Kitten
Puppy
None
INPUT
FEATURE EXTRACTION + CLASSIFICATION
OUTPUT

MNIST – Popular CNN model a.k.a Hello world of ML

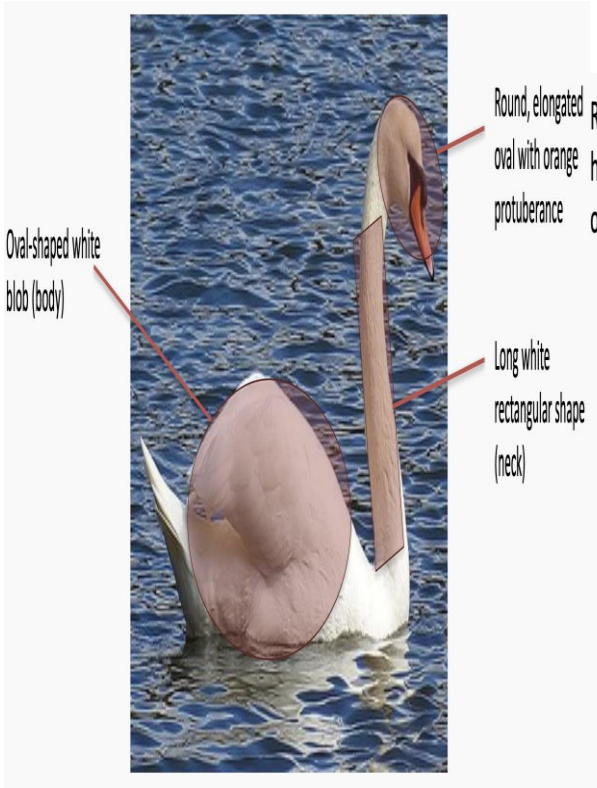
Data & Labels

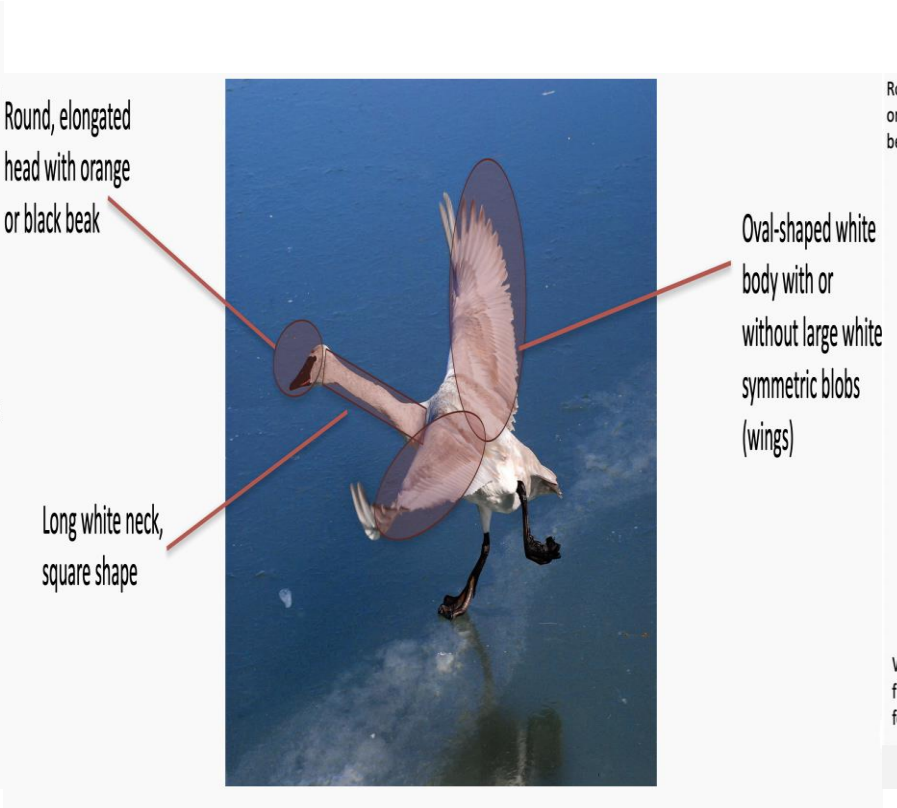


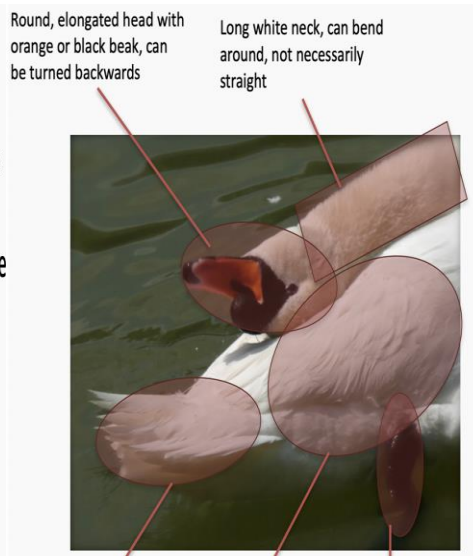
0
1
2
3
4
5
6
7
8
9

Lets try to learn the flow involved in building a CNN model capable of finding SWANS

A swan has some unique features. All its features has to be extracted and is used for recognition purposes.







Man in swan tent photographing swans

Worst-case scenarios.

CNN

- The general idea of CNN's is to intelligently adapt to the properties of images:
- Pixel position and neighborhood have semantic meanings
- Elements of interest can appear anywhere in the image

A CNN has

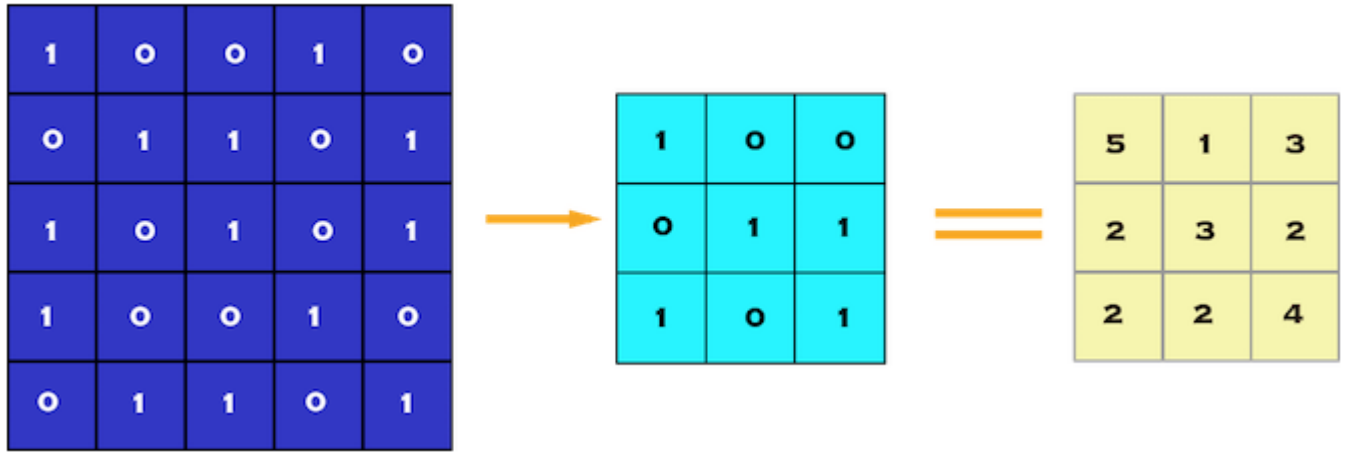
- Convolutional layers
- ReLU layers
- Pooling layers
- a Fully connected layer

Enter the Convolutional Neural Network

- We analyze the influence of nearby pixels by using something called a filter. A filter is exactly what you think it is, in our situation, we take a filter of a size specified by the user (a rule of thumb is 3x3 or 5x5) and we move this across the image from top left to bottom right. For each point on the image, a value is calculated based on the filter using a convolution operation
- A filter could be related to anything, for pictures of humans, one filter could be associated with seeing noses, and our nose filter would give us an indication of how strongly a nose seems to appear in our image, and how many times and in what locations they occur.

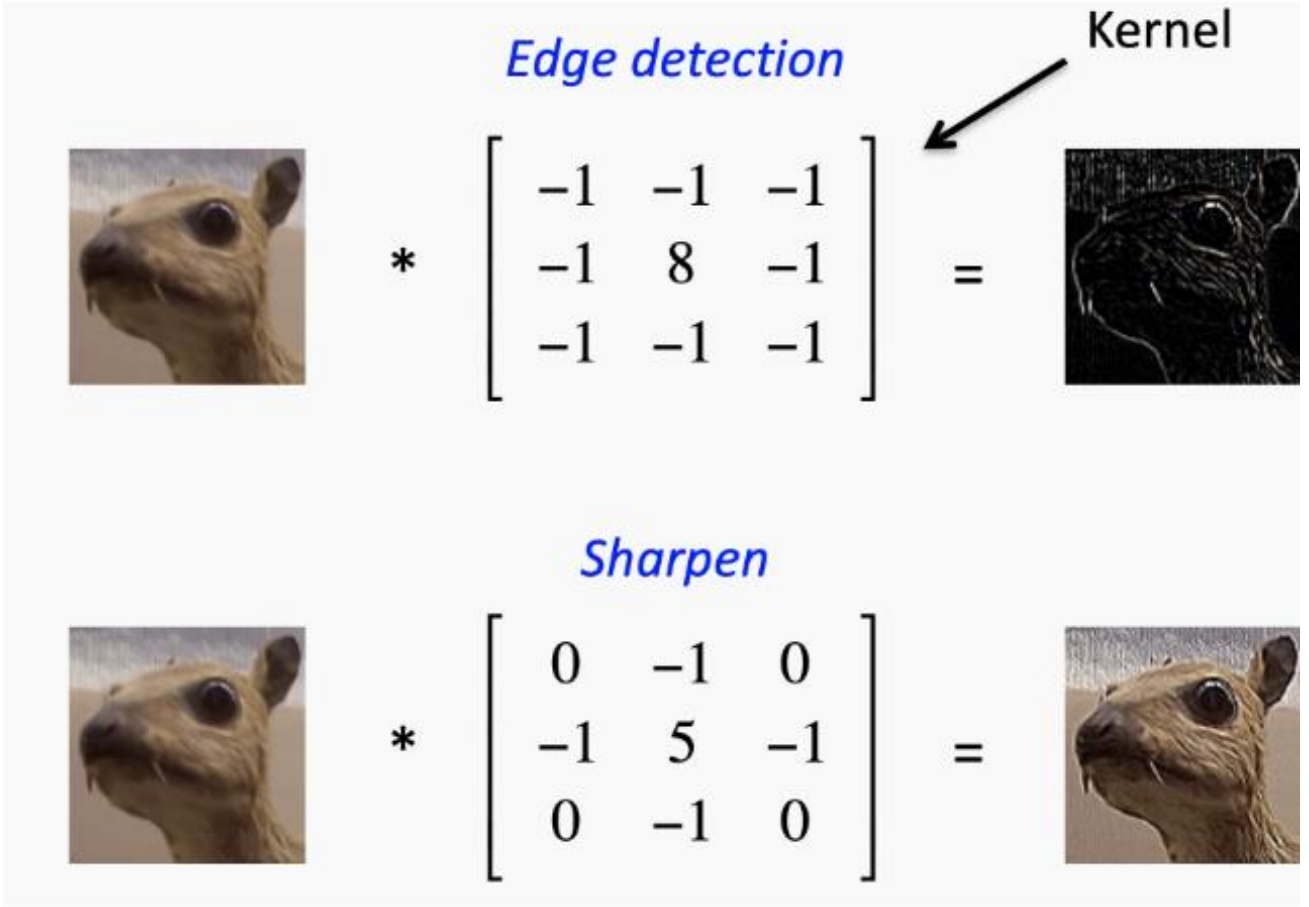
CNN LAYERS

- Convolutional layer is the very first layer where we extract features from the images in our datasets
- It decreases the size of the image without losing the relationship between pixels



Convolutional layers(Filters)

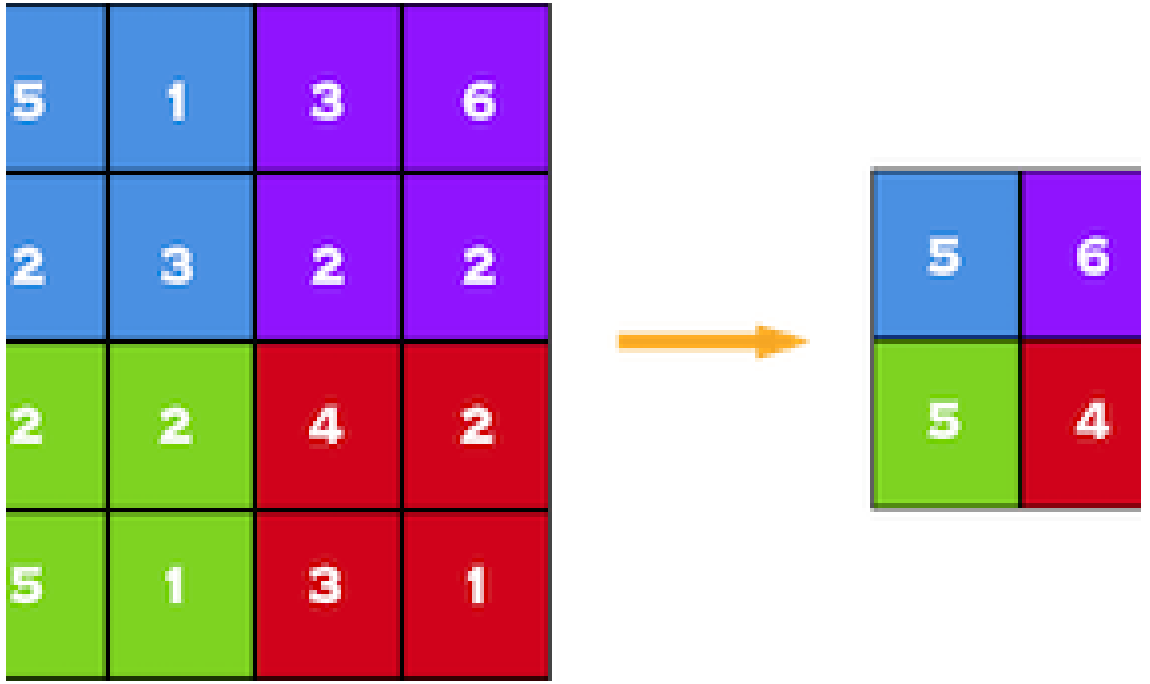
After the filters have passed over the image, a feature map is generated for each filter. These are then taken through an activation function, which decides whether a certain feature is present at a given location in the image. We can then do a lot of things, such as adding more filtering layers and creating more feature maps, which become more and more abstract as we create a deeper CNN.



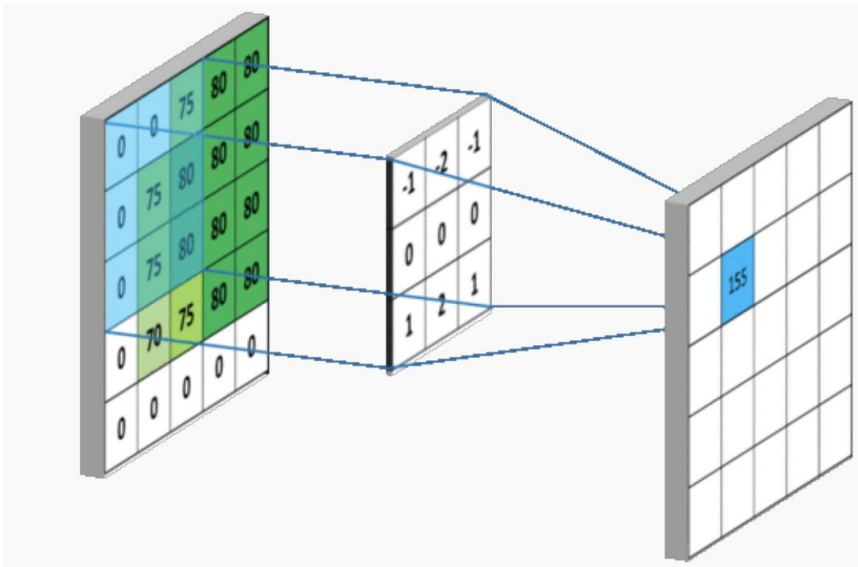
Examples of kernel filters for CNN's.

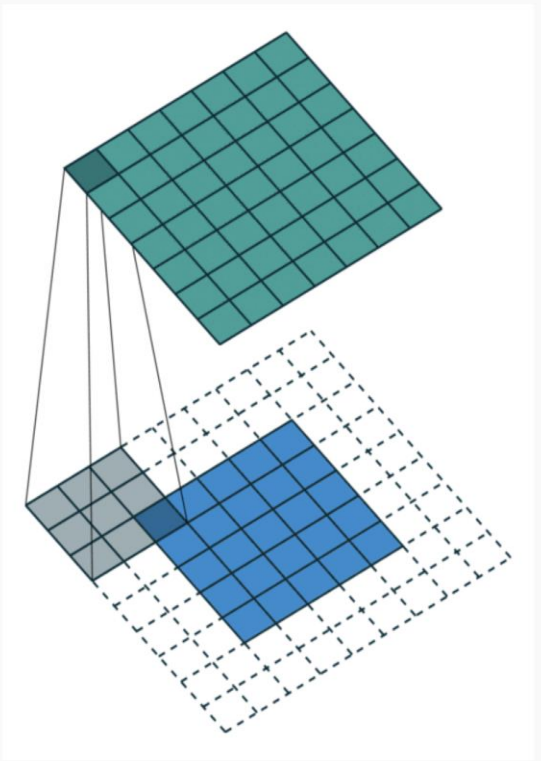
Pooling layers

When constructing CNNs, it is common to insert pooling layers after each convolution layer to reduce the spatial size of the representation to reduce the parameter counts which reduces the computational complexity. In addition, pooling layers also helps with the overfitting problem. Basically we select a pooling size to reduce the amount of the parameters by selecting the maximum, average, or sum values inside these pixels. Max Pooling, one of the most common pooling techniques, may be demonstrated as follows:

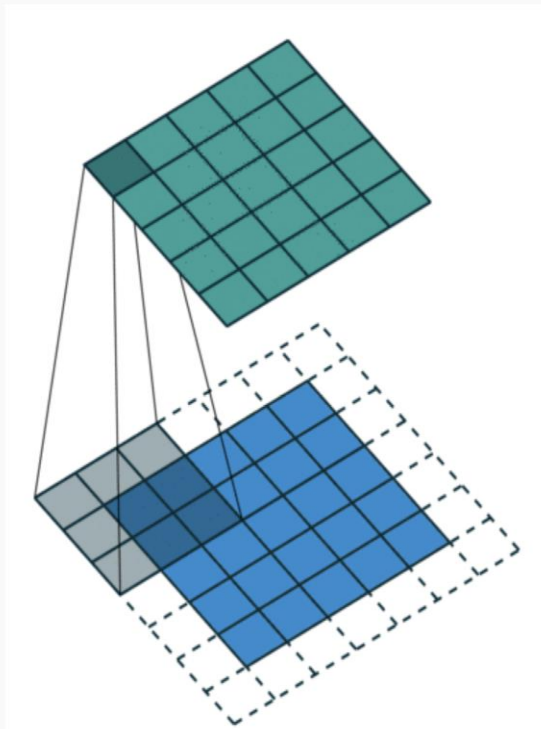


Example of how convolutions are applied to images using kernel filters.



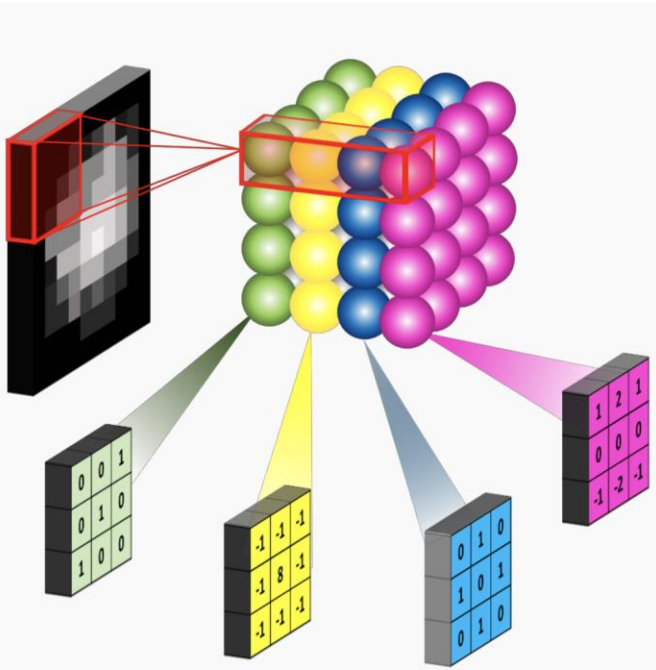


Full padding. Introduces zeros such that all pixels are visited the same amount of times by the filter. Increases size of output.

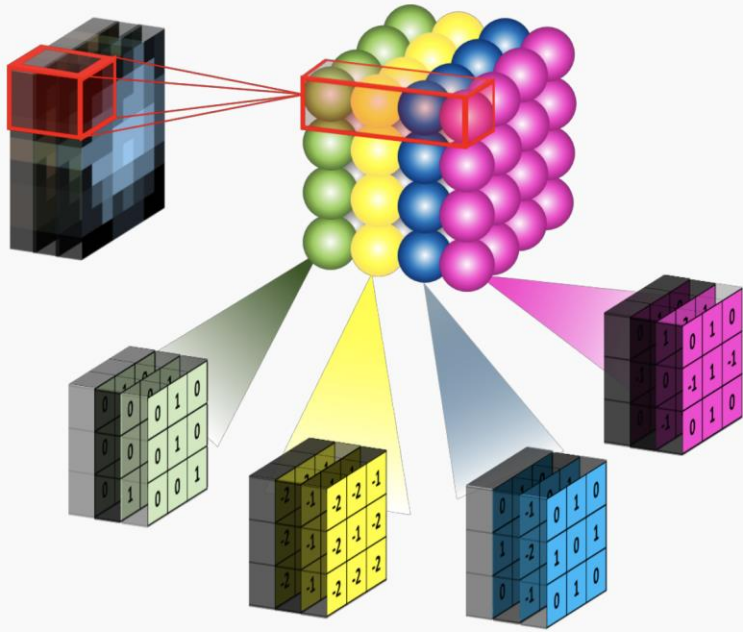


Same padding. Ensures that the output has the same size as the input.

How do we connect our filters together?



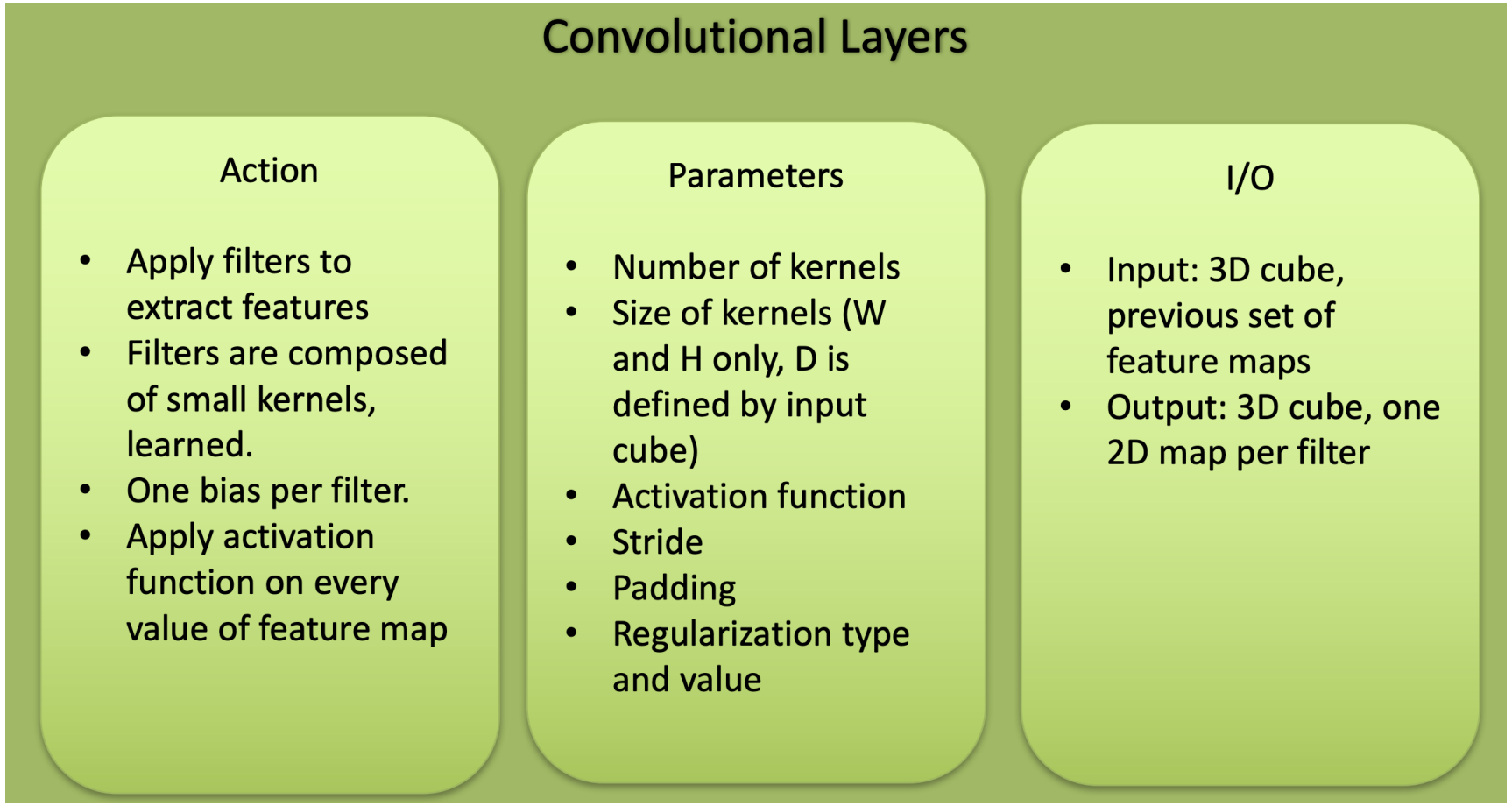
Convolutional layer with four 3x3 filters on a black and white image (just one channel)



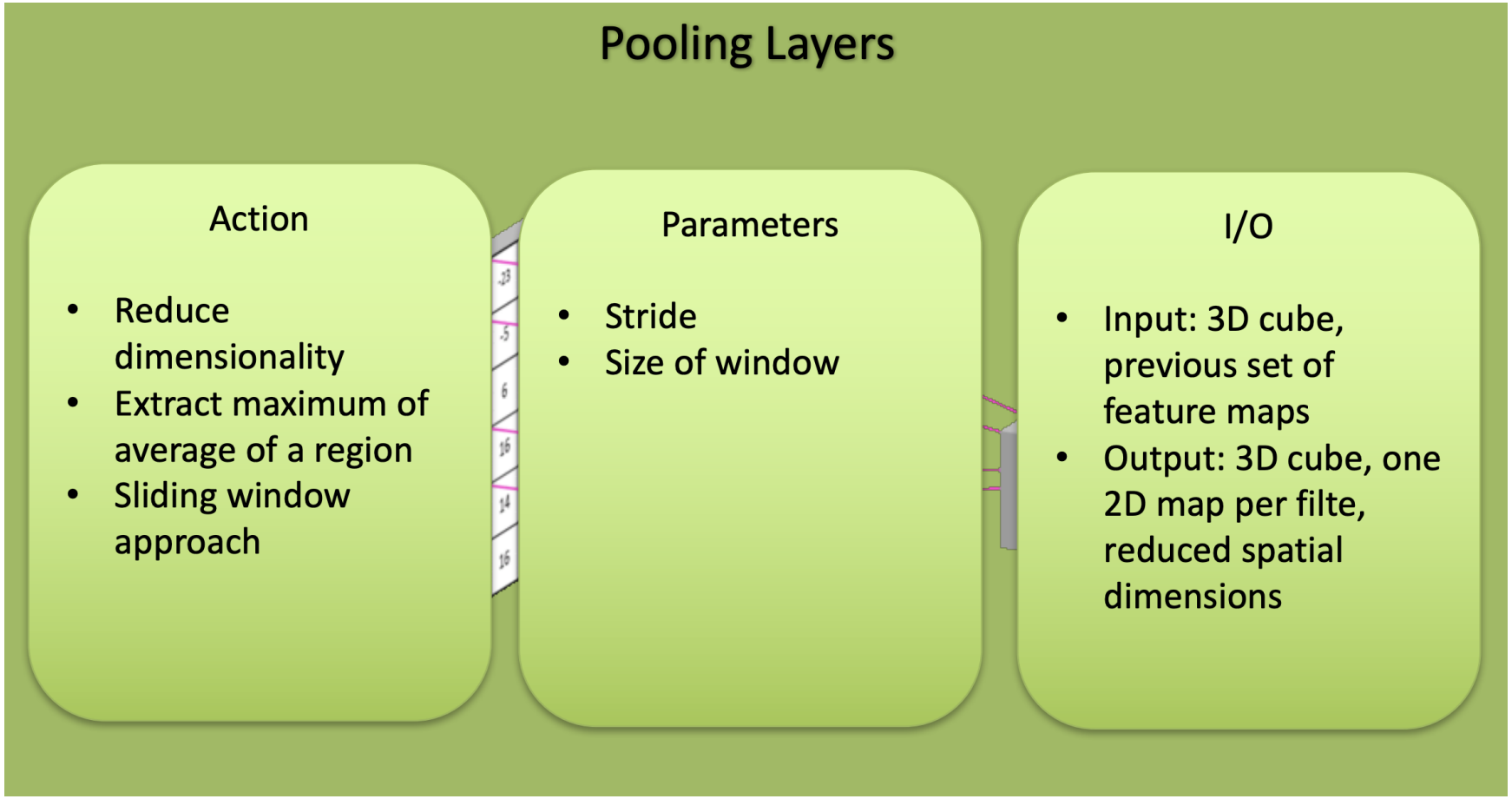
Convolutional layer with four 3x3 filters on an RGB image. As you can see, the filters are now cubes, and they are applied on the full depth of the image..

- To be clear, each filter is convolved with the entirety of the 3D input cube but generates a 2D feature map.
- Because we have multiple filters, we end up with a 3D output: one 2D feature map per filter
- The feature map dimension can change drastically from one convolutional layer to the next: we can enter a layer with a 32x32x16 input and exit with a 32x32x128 output if that layer has 128 filters.
- Convolving the image with a filter produces a feature map that highlights the presence of a given feature in the image.

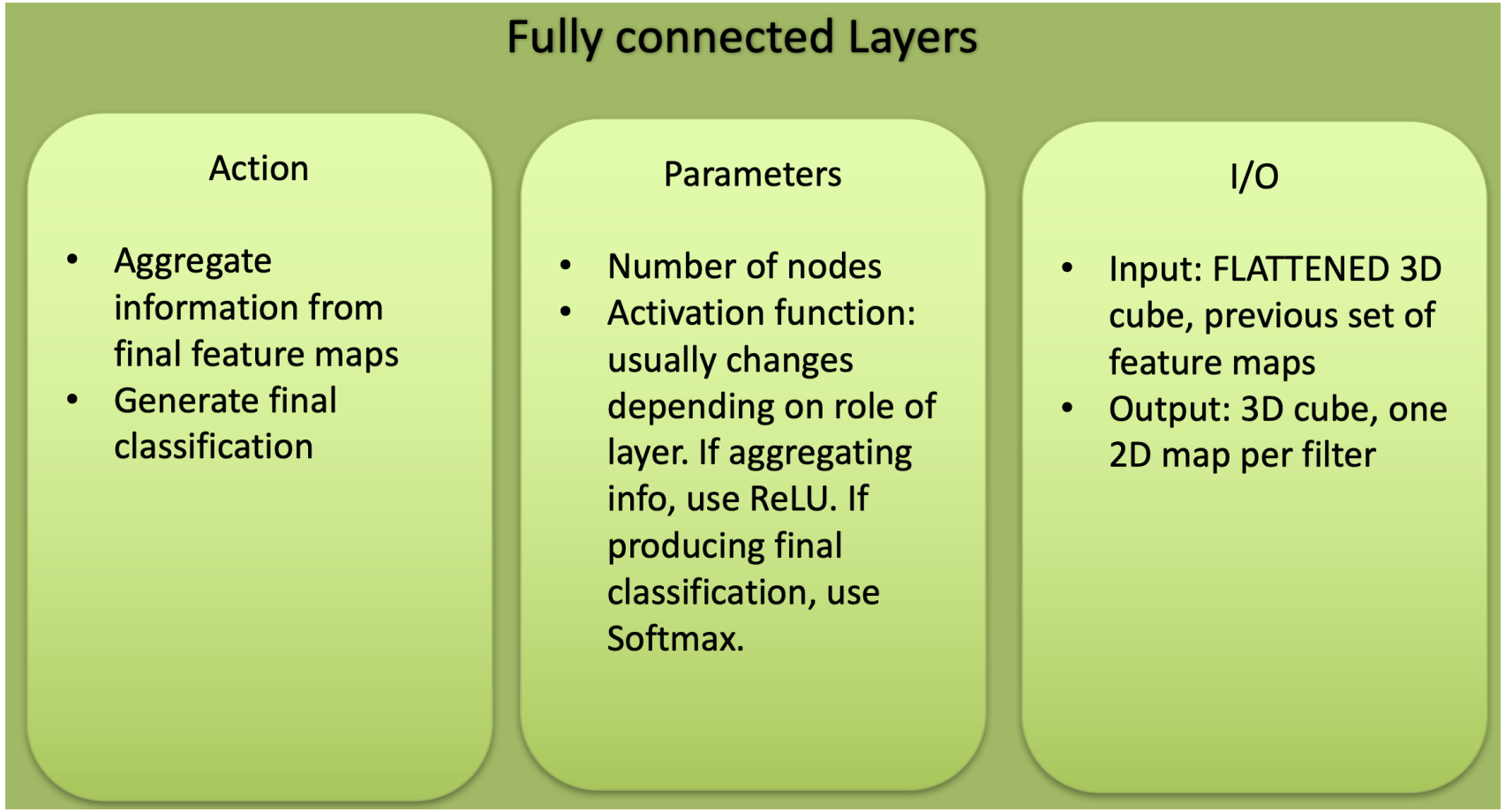
SUMMARY


Convolutional Layers
Action
Apply filters to extract features
Filters are composed of small kernels, learned.
One bias per filter.
Apply activation function on every value of feature map
Parameters
Number of kernels
Size of kernels (W and H only, D is defined by input cube)
Activation function
Stride
Padding
Regularization type and value
I/O
Input: 3D cube, previous set of feature maps
Output: 3D cube, one 2D map per filter

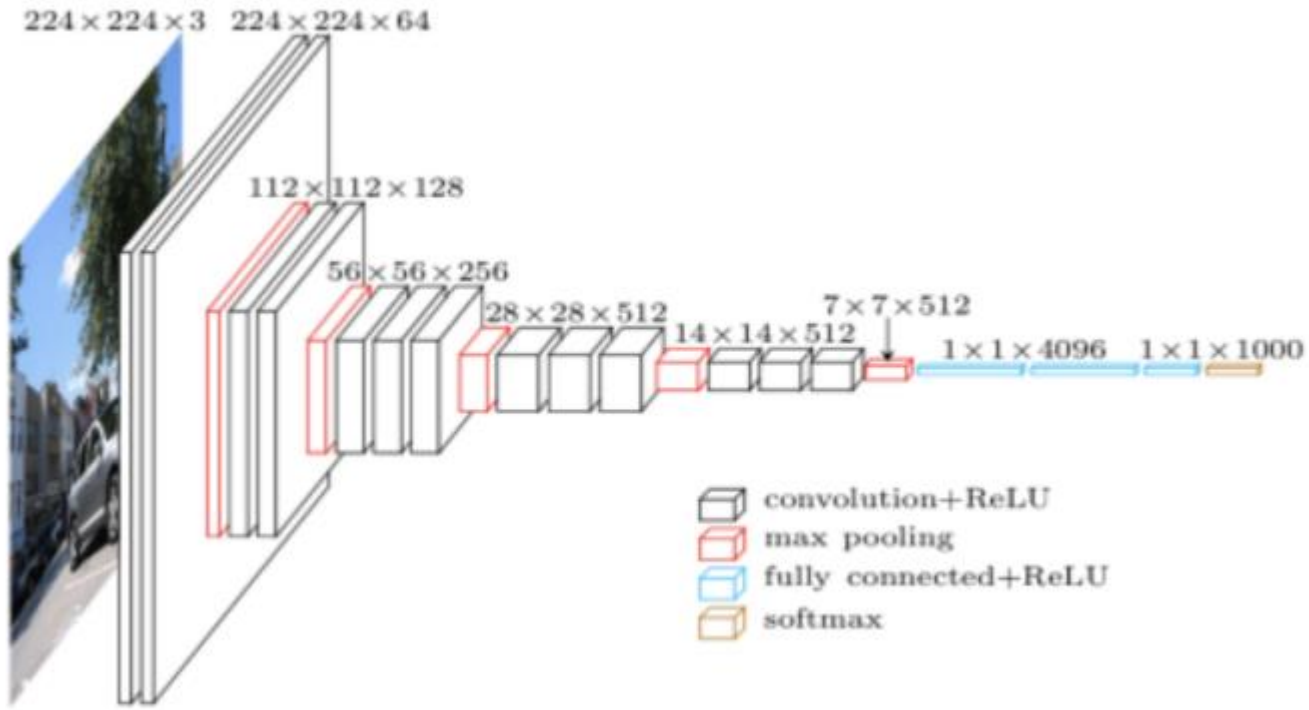
SUMMARY


Pooling Layers
Action
Reduce dimensionality
Extract maximum of average of a region
Sliding window approach
Parameters
Stride
Size of window
I/O
Input: 3D cube, previous set of feature maps
Output: 3D cube, one 2D map per filte, reduced spatial dimensions

SUMMARY


Fully connected Layers
Action
Aggregate information from final feature maps
Generate final classification
Parameters
Number of nodes
Activation function: usually changes depending on role of layer. If aggregating info, use ReLU. If producing final classification, use Softmax.
I/O
Input: FLATTENED 3D cube, previous set of feature maps
Output: 3D cube, one 2D map per filter

Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification.



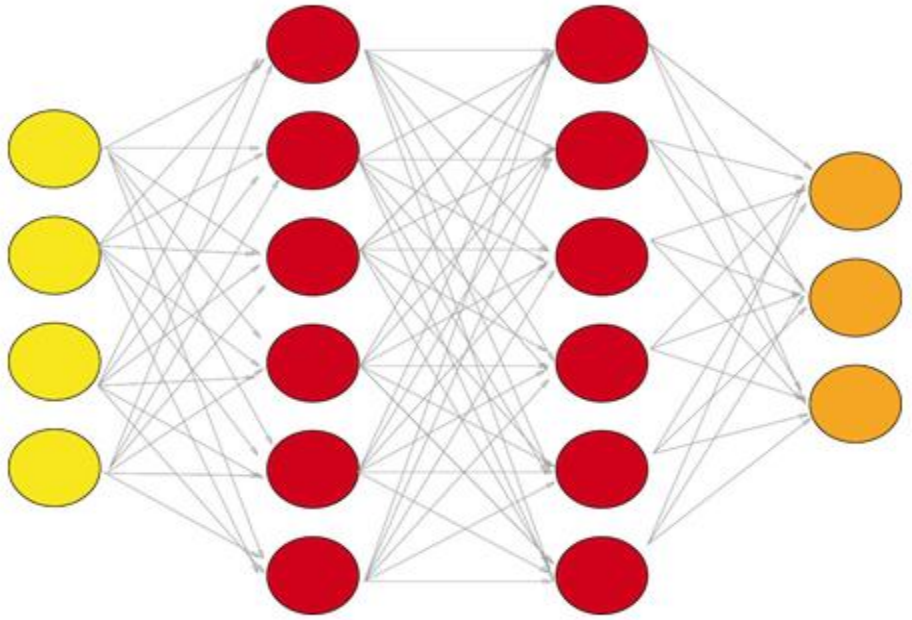
The architecture of a standard CNN.

What do CNN layers learn?

- Each CNN layer learns filters of increasing complexity.
- The first layers learn basic feature detection filters: edges, corners, etc
- The middle layers learn filters that detect parts of objects. For faces, they might learn to respond to eyes, noses, etc
- The last layers have higher representations: they learn to recognize full objects, in different shapes and positions.

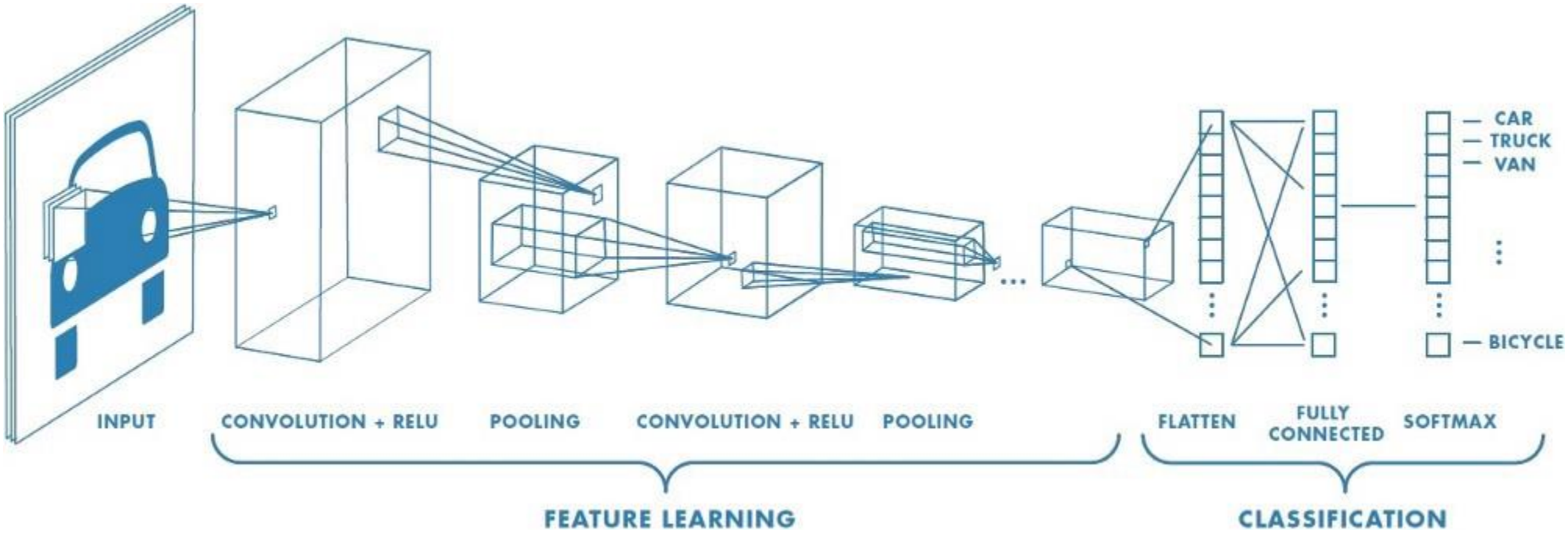
Fully connected layer

- The main purpose of the artificial neural network is to combine our features into more attributes. These will predict the classes with greater accuracy. This combines features and attributes that can predict classes better.



A fully connected layer with two hidden layers

Summary of the CNN Flow:



A CNN

- starts with an input image
- applies many different filters to it to create a feature map
- applies a ReLU function to increase non-linearity
- applies a pooling layer to each feature map
- flattens the pooled images into one long vector.
- inputs the vector into a fully connected artificial neural network.
- processes the features through the network. The final fully connected layer provides the “voting” of the classes that we’re after.
- trains through forward propagation and backpropagation for many, many epochs. This repeats until we have a well-defined neural network with trained weights and feature detectors.

YOLO – You Only Look Once

- LeNet-5
- AlexNet
- VGG-16
- Inception-v1
- Inception-v3
- ResNet-50
- Xception
- YOLO-Darket
- Inception-ResNets

Among these, YOLO is found to be the fastest and most efficient Architechture.
YOLO v3 uses a variant of Darknet, which originally has 53 layer network trained on Imagenet. For the task of detection, 53 more layers are stacked onto it, giving us a **106 layer fully convolutional underlying architecture for YOLO v3**.
The Deeper the better!!!

- Darknet is a framework written using C and CUDA which enables easy implementation on various CPU and GPU.
- Yolo is the state-of-the-art object detection algorithm:
- Bounding box prediction • Class prediction • Prediction across scales
- Feature extraction • Output Result

How it works?

Yolo initially divides the image into a grid of 13 by 13 cells. And each cells can now predict upto 5 bounding boxes with confident scores. Once the bounding boxes are framed with certain threshold, the image is then applied to the pretrained images at multiple locations and the highest scoring regions are considered for the output.

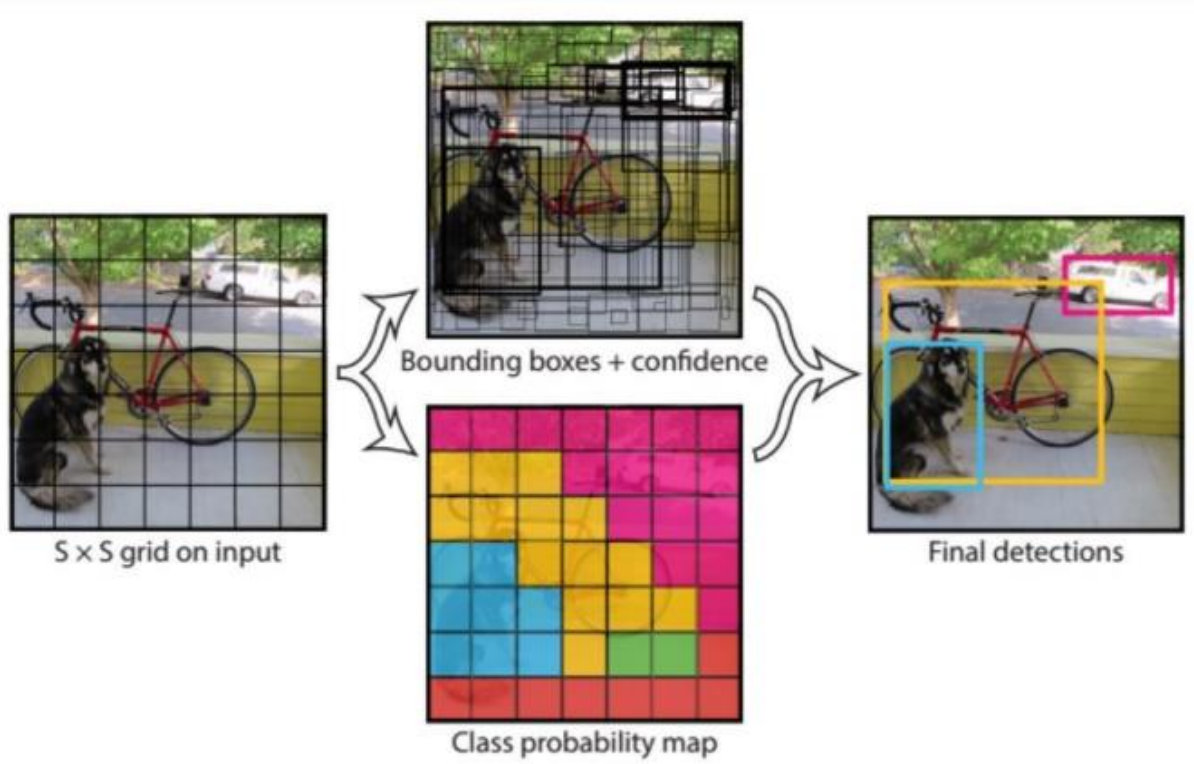


FIG 2: OUTLINE ARCHITECTURE OF YOLO [11]

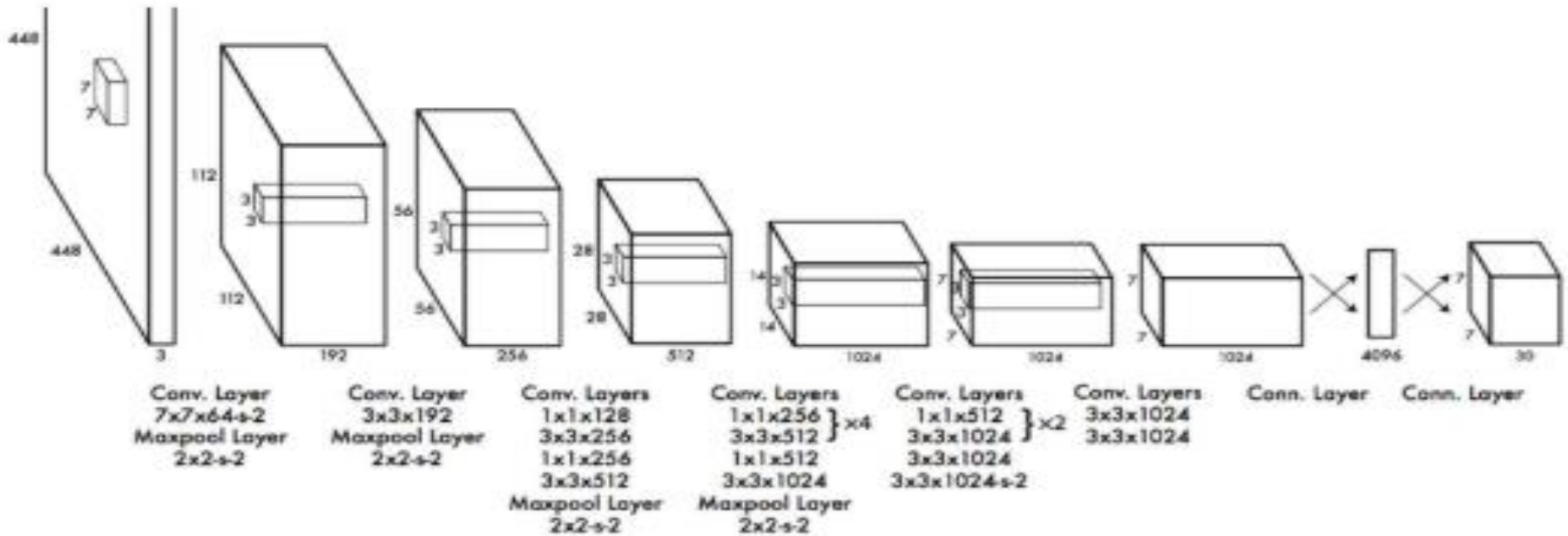
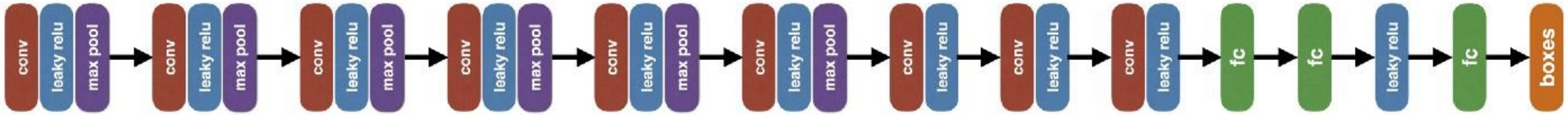


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

Architecture of Tiny Yolo

Tiny yolo consists of 9 convolutional layers and 3 fully connected layers.
Each convolution layer consists of convolution, leaky relu and max pooling operations. The first 9 convolution layers can be understood as the feature extractor, whereas the last three full connected layers can be understood as the "regression head" that predicts the bounding boxes.



USEFUL LINKS:

- Yolo tutorial by Siraj : https://www.youtube.com/watch?v=4eIBisqx9_g
- Github - https://github.com/llSourcell/YOLO_Object_Detection
- How CNN works : https://www.youtube.com/watch?v=FmpDIaiMIeA
- Detailed Explanation: https://www.youtube.com/watch?v=JB8T_zN7ZC0
- CNN with python basics : https://www.youtube.com/watch?v=04G3kRFI7pc
- Mnist TUTORIAL: https://missinglink.ai/guides/convolutional-neural-networks/python-convolutional-neural-network-creating-cnn-keras-tensorflow-plain-python/

You need to have Python 3.5 or 3.6, Tensorflow, numPY, openCV on your (laptop) environment to get started. Below is some guidance that has worked for me on Windows.

Step 1 — Install the dependencies for Windows

For beginners you can install the following to have a clean slate of ready-to-go personal computing environment for your future ML experiments.

1. Download & install Anaconda package 64-bit version and choose the **Python 3.6 version**. (**link** to video tutorial) This automatically installs Python and many popular data scientist/ML libraries (*NumPy, Scikit-Learn, Pandas, R, Matplotlib…*), tools (*Jupyter Notebook, RStudio*) and hundreds of other open source packages for your future projects. When you start out, it feels like the closest thing to the holy grail of ML packages… For example I still use the Anaconda Jupyter Notebook for almost all my ML experiments mostly out of convenience. **openCV** library is not included though and we will install it separately as it is needed for real-time computer vision tasks. (*Hint for Anaconda folks!*)

2. Install **Tensorflow** and **Keras** (optional). TensorFlow is the most popular AI software library and is created/maintained by Google. **Keras** is another highly popular & high-level neural networks API, written in Python and capable of running on top of TensorFlow. It was developed with a focus on enabling fast experimentation. When you feel like "quitting" after going thru *Andrew Ng 's like really low level material, Keras feels like a piece of cake! Because it is a high level language based on Python. **Someone else** has done the hard work for you!*

It is common to have issues while trying to install all these open source packages especially if you are on a Windows machine. It takes a while to make everything work and resolve all the version or conflict issues. My best practice is to basically Google such issues and find solutions online. Websites like stackoverflow are super helpful and save your time & sanity!

In general, I also find helpful to create a separate new **conda virtual environment** to mitigate the Windows installation issues. More on that here.

Step 2 — Install the DarkNet/YOLA, Darkflow stuff

DarkNet: Originally, YOLO algorithm is implemented in DarkNet framework by Joseph Redmon. Darknet is an open source custom neural network framework written in C and CUDA. It is fast, easy to install, and supports both CPU and GPU computations. You can find the open source on GitHub.

Darkflow: It is a nickname of an implementation of YOLO on TensorFlow. Thanks to Trinh Hoang Trieu, Darknet models are converted to Tensorflow and can be installed on both Linux and Windows environments. Lets do it!

Open your anaconda prompt and clone the darkflow github repository. (*You may need to install **Git Bash** Windows for git command to work)*

git clone https://github.com/thtrieu/darkflow

 # Alternative is to basically go to the DarkFlow GitHub page and download the master repository to your local (i.e. C:\users\user_name\darkflow)

If you have not already created a new virtual environment in Step 1, then create a conda environment for darkflow installation.

```
conda create -n your_env_name python=3.6
```

Activate the new environment using anaconda prompt.

```
activate your_env_name
```

You can install the needed OpenCV with a conda-forge repository. conda-forge is a github organization containing repositories of conda libraries.

```
conda config --add channels conda-forge
```

```
conda install opencv
```

Build the Cython extensions in place. This is a widely used Python to C compiler and wrapper that helps us to call the DarkNet C-code from Python.

```
python setup.py build_ext --inplace
```

or try the following as alternative

```
pip install -e .
```

If you get an error, try changing the working directory to darkflow (cd darkflow) first and re-run one of the above commands.

Cool. The above steps will hopefully setup a local environment to run darkflow and perform object detection task on images or videos.

Lastly we need to download the **CFG** and **WEIGHTS** files. The pre-trained model name is YOLOv2 that is trained on a COCO image data set containing 80 classes (image types like car, dog, person, aeroplane etc).

WEIGHTS file: Please download the yolov2.weights file from here. Pls **create a darkflow/bin** directory for keeping these weights file.

CFG file: Create a yolo.cfg text file of corresponding model in the existing **darkflow/cfg** directory under your local darkflow folder. Check here for the source file. You can copy paste the raw GitHub content with a notepad if you want. Also do not forget to have a look at Darkflow's command line help options for future reference.

```
python flow --h
```

PS: I found this blog (Abhijeet Kumar) very helpful while i was figuring out the needed installations.

We are all set.

Lets run Darkflow YOLO command line to render some video!

I fancy using the Anaconda command prompt to execute the following commands. You can find it from the Windows Start menu via searching for “Anaconda Prompt”. In the prompt window, activate your new

Tensorflow virtual environment via “activate your_environ_name” command. Then execute the “cd darkflow” command to change the current working directory to your local Darkflow repository. Then you can try the following commands to start running DarkFlow to process images & videos.

1. For processing existing images, you can run the following command:

   python flow --model cfg/yolo.cfg --load bin/yolov2.weights --imgdir sample_img

Pls note that darkflow/sample_img is a directory with sample photos.

2. For processing a video file, you can move the to-be-rendered video file under master darkflow folder and then use the following command:

python flow --model cfg/yolo.cfg --load bin/yolov2.weights --demo samplename.mp4

Hint: If you append “ — saveVideo” at the end, you can save the processed video under master folder as well.

3. For rendering a real time streaming video via your laptop camera:

python flow --model cfg/yolo.cfg --load bin/yolov2.weights --demo camera

(if you come across any ‘file/library not exist’ error in anaconda prompt, simply type ‘pip install xxxx’ where, xxxx is the library name that is missing or you can try copying the error message and pasting it on google search engine)

Alternatively, you can run the .ipynb file from the link below once you download all the necessary dependancies in order to ease the compilation and code structure

https://github.com/Garima13a/YOLO-Object-Detection

You can implement YOLO OBJECT DETECTION to detect custom trained datasets. Refer to the following links to know more

1. https://github.com/jacksonxliu/YOLOv3-tiny-custom-object-detection
2. https://blog.francium.tech/custom-object-training-and-detection-with-yolov3-darknet-and-opencv-41542f2ff44e
3. https://github.com/ultralytics/yolov3/wiki/Train-Custom-Data
4. http://emaraic.com/blog/yolov3-custom-object-detector

Setting Up the ZCU102/104 Evaluation Board

The Vitis™ AI software is made available via docker hub
https://hub.docker.com/r/xilinx/vitis-ai/tags

Vitis AI consists of the following two docker images:

- xilinx/vitis-ai:tools-1.0.0-cpu
- xilinx/vitis-ai:runtime-1.0.0-cpu

Setting Up the Host

- Clone the Vitis AI repository:

On linux terminal : “git clone https://github.com/xilinx/vitis-ai”

- Set up Vitis AI to target Alveo cards (Only for systems with Alveo cards)
- Run the following commands:

cd Vitis-AI/alveo/packages

sudo su ./install.sh

Start the Docker Container.

a. Change directories to Vitis AI:

cd Vitis-AI/

b. Run one of the following command sets.

- For a CPU tools container:

./docker_run.sh xilinx/vitis-ai:1.0.0-cpu

- For a GPU-enabled tools container:

cd Vitis-AI/docker

./docker_build.sh

cd Vitis-AI

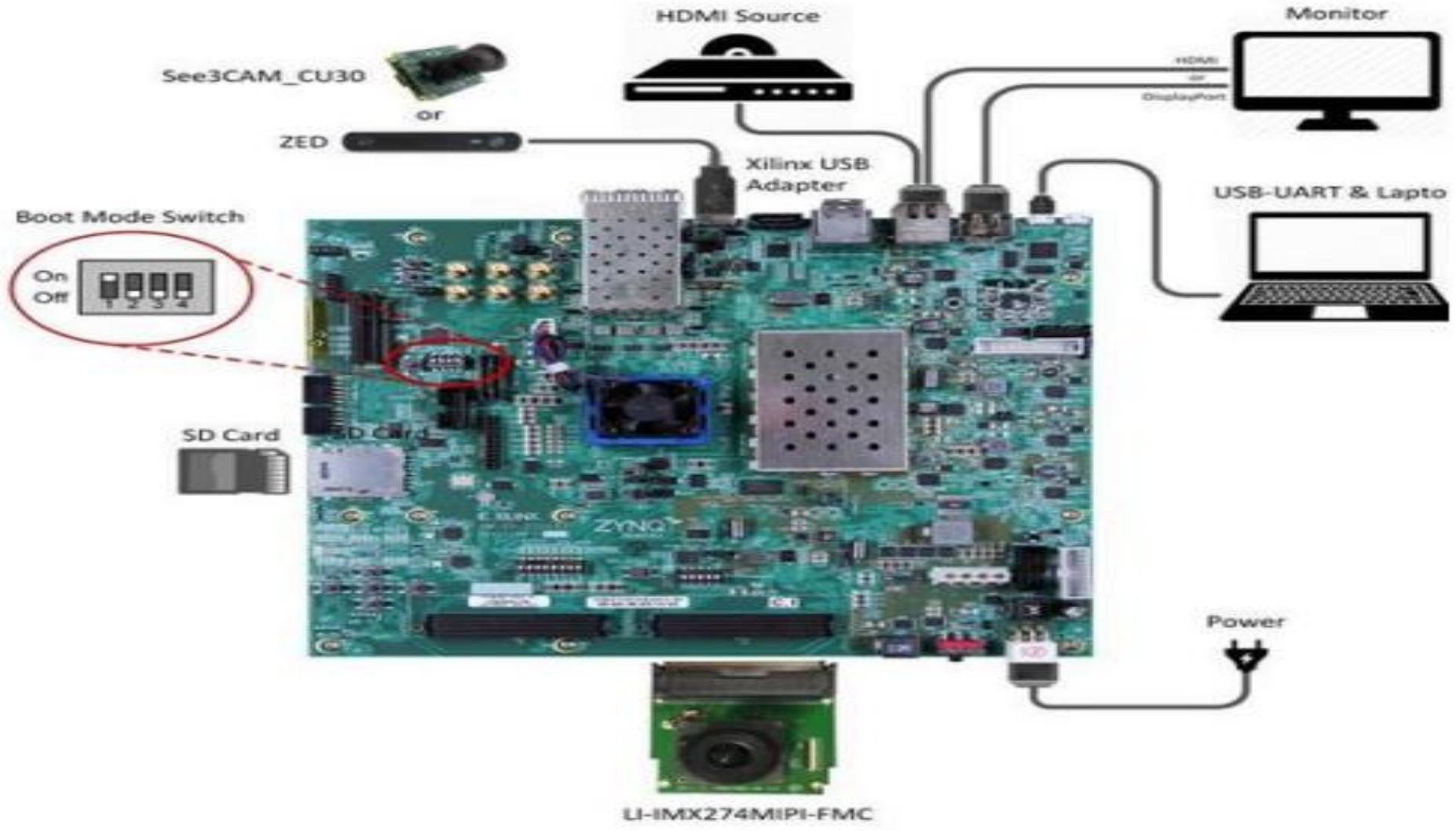
./docker_run.sh xilinx/vitis-ai:runtime-1.0.0-gpu

- For MPSoC Runtime tools container:

./docker_run.sh xilinx/vitis-ai:runtime:1.0.0-cpu

c. Upon starting the container your current working directory will be mounted to: /workspace

Xilinx ZCU102 Evaluation Board and Peripheral Connections



Flashing the OS Image to the SD Card

- For ZCU102, the system images can be downloaded from here:

https://www.xilinx.com/bin/public/openDownload?filename=xilinx-zcu102-dpu-v2019.2.img.gz

The image file has to be flashed into the sd card

i. Download and install Etcher from: https://etcher.io/

ii. Eject any external storage devices such as USB flash drives and backup hard disks. Then, insert the SD card into the slot on your computer, or into the reader.

iii. Run Etcher. Choose the image file -> Choose the SD card ->Click Flash

Booting the Evaluation Board

i. Connect the power supply (12V ~ 5A).
ii. Connect the UART debug interface to the host and other peripherals as required.
iii. Turn on the power and wait for the system to boot.

Accessing the Evaluation Board

There are three ways to access the ZCU102 board: • UART port • Ethernet connection • Standalone

You generally need to connect the board via both UART port and Ethernet connection with the host PC

Download and install Putty or Teraterm in linux using linux Apps search bar

UART Port

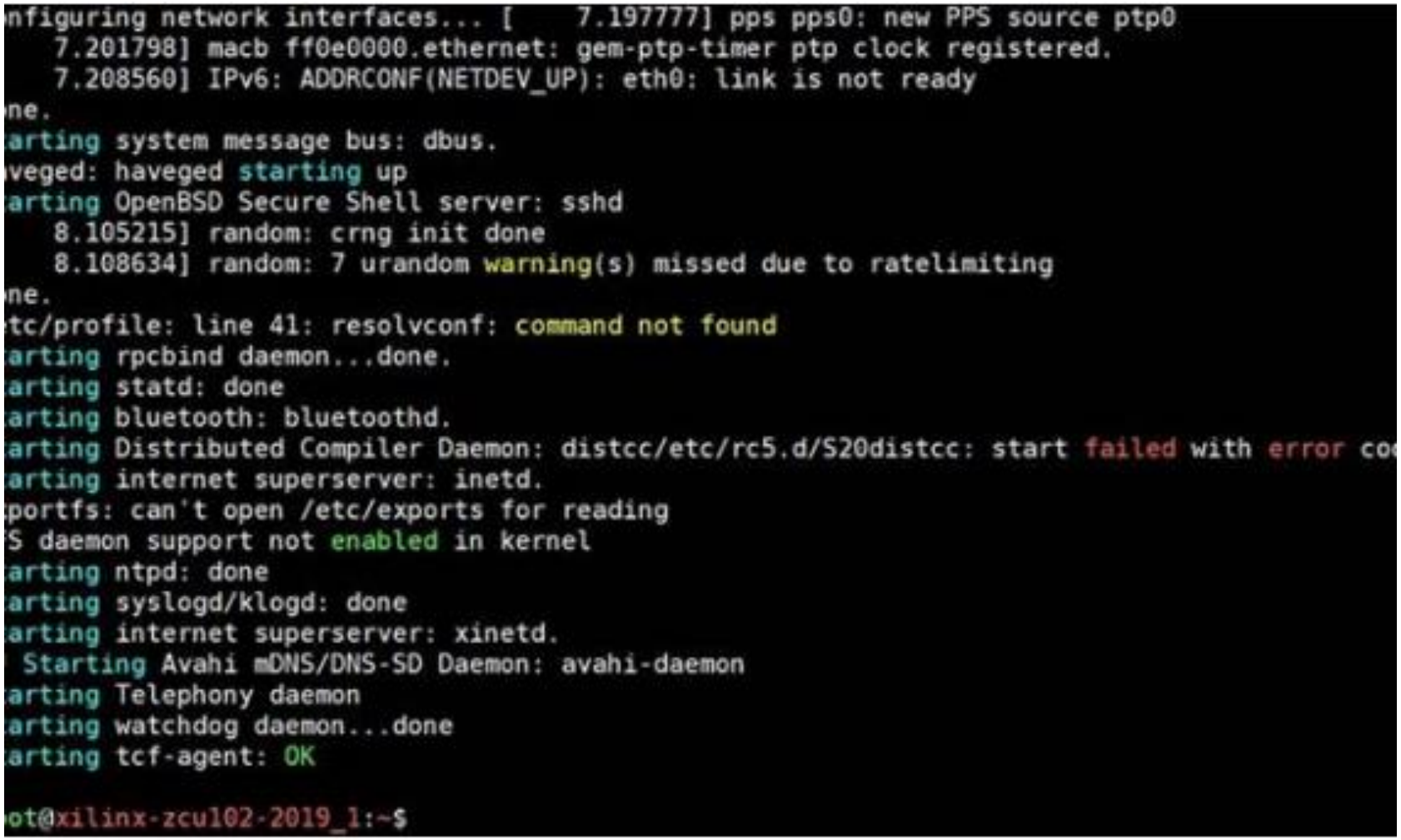
Open Putty by typing “sudo putty” -> select serial port -> Choose the USB port

(The ttyUSB number varies. I suggest you to try each ones. If you cant find ttyUSB in dropdown list box in putty, try google the issue. The linux connection port drivers might need an update. Try typing “lsusb” on linux terminal)

Select • baud rate: 115200 bps • data bit: 8 • stop bit: 1 • no parity

(Make sure you type the right baudrate. Once, the terminal is launched, press enter to get the cursor)

Example of UART Boot on Putty Terminal



Ethernet Port

- You need to assign appropriate ip address in both host pc and the target to establish Ethernet Connection
- On putty terminal type “ipconfig eth0 192.168.0.101”
- On linux terminal type “ipconfig –a” to know the name of the Ethernet port on host pc we are trying to connect. Typically it is eth0 or something like enx00…,
- On linux terminal type “sudo ipconfig eth0*(or enx00… or other name)* 192.168.0.100”
- Verify the Ethernet connection by typing “ping 192.168.0.101” on linux terminal. This will exchange packets between host and target. Press Ctrl+C to stop verifying

Installing Vitis AI Package on the Evaluation Board

- With an Ethernet connection established, you can copy the Vitis AI installation package from docker image vitis-ai-docker-runtime to the evaluation board and set up Vitis AI running environment for the ZCU102 board.
- On linux terminal “sudo scp -r /opt/vitis_ai/xilinx_vai_board_package root@192.168.0.101:~/”

(syntax: scp-r filetobecopied root@IPofboard:~/”

On the ZCU102 board, change to the ~/xilinx_vai_board_package/ directory and run install.sh. The Vitis AI runtime and utility tools will be installed into system automatically. You can now copy Vitis AI samples from docker image vitis-ai-docker-runtime to the evaluation board for evaluation.

Running Examples

- samples can be found at https://github.com/xilinx/vitis-ai
- The /alveo folder contains the sample for DPU-v1 on Alveo platform, and the folder mpsoc contains the samples for edge DPU on ZCU102 and ZCU104 boards
- If you are using Xilinx ZCU102 and ZCU104 boards to run samples, make sure to enable X11 forwarding with the command export DISPLAY=192.168.0.10:0.0 (assuming the IP address of host machine is 192.168.0.10) when logging in to the board using an SSH terminal since all the examples require Linux windows system to work properly.

- Vitis AI samples can be found in the following locations:
- ZCU102 board samples: https://github.com/Xilinx/Vitis-AI/tree/master/mpsoc/vitis_ai_samples_zcu102
- After downloading the samples, copy them into your /workspace/sample/ folder
- Test images can be found in /workspace/sample/vitis_ai_samples_zcu102/images/
- copying the whole directory /workspace/sample/vitis_ai_samples_zcu102/ to ZCU102 board directory /home/root/ is recommend.

- The launching command for each sample is listed in the following table. For Python samples, note that the absolute path for dpuv2_rundir should be specified.

ID	Example Name	Command
1	resnet50	./resnet50 dpuv2_rundir
2	resnet50_mt_py	python3 resnet50.py 3 /home/root/vitis_ai_samples_zcu102/ resnet50_mt_py/dpuv2_rundir/
3	inception_v1_mt_py	python3 inception_v1.py 3 /home/root/vitis_ai_samples_zcu102/inception_v1_mt_py/dpuv2_rundir/
4	pose_detection	./pose_detection video/pose.mp4 dpuv2_rundir
5	video_analysis	./video_analysis video/structure.mp4 dpuv2_rundir
6	adas_detection	./adas_detection video/adas.avi dpuv2_rundir
7	segmentation	./segmentation video/traffic.mp4 dpuv2_rundir

Legacy DNNDK Examples

- The legacy DNNDK C++/Python examples can be found at the following locations:
- ZCU102 examples: https://github.com/Xilinx/Vitis-AI/tree/master/mpsoc/dnndk_samples_zcu102

After downloading the samples, copy them into the /workspace/sample/ folder within the runtime container. These examples can be built with Arm GCC cross-compilation toolchains.

- The samples stay under the directory /workspaces/sample/dnndk_samples_zcu102/. After all the samples are built by Arm GCC cross-compilation toolchains within runtime container, it is recommended to copy the whole directory /workspaces/sample/dnndk_samples_zcu102/ to ZCU102 board directory /home/root/.

Running Examples

ResNet-50

$dnndk_sample_base/resnet50 contains an example of image classification using Caffe ResNet-50 model. It reads the images under the $dnndk_sample_base/dataset/image500_640_480 directory and outputs the classification result for each input image. You can then launch it with the./resnet50 command.

Video Analytics

An object detection example is located under the $dnndk_sample_base/video_analysis directory. It reads image frames from a video file and annotates detected vehicles and pedestrians in real-time. Launch it with the command ./video_analysis video/structure.mp4 (where video/structure.mp4 is the input video file).

ADAS Detection

An example of object detection for ADAS (Advanced Driver Assistance Systems) application using YOLO-v3 network model is located under the directory $dnndk_sample_base/adas_detection. It reads image frames from a video file and annotates in real-time. Launch it
with the ./adas_detection video/adas.avi command (where video/adas.mp4 is the input video file).

Semantic Segmentation

An example of semantic segmentation in the $dnndk_sample_base/segmentation directory. It reads image frames from a video file and annotates in real-time. Launch it with the ./segmentation video/traffic.mp4 command (where video/traffic.mp4 is the input video file).

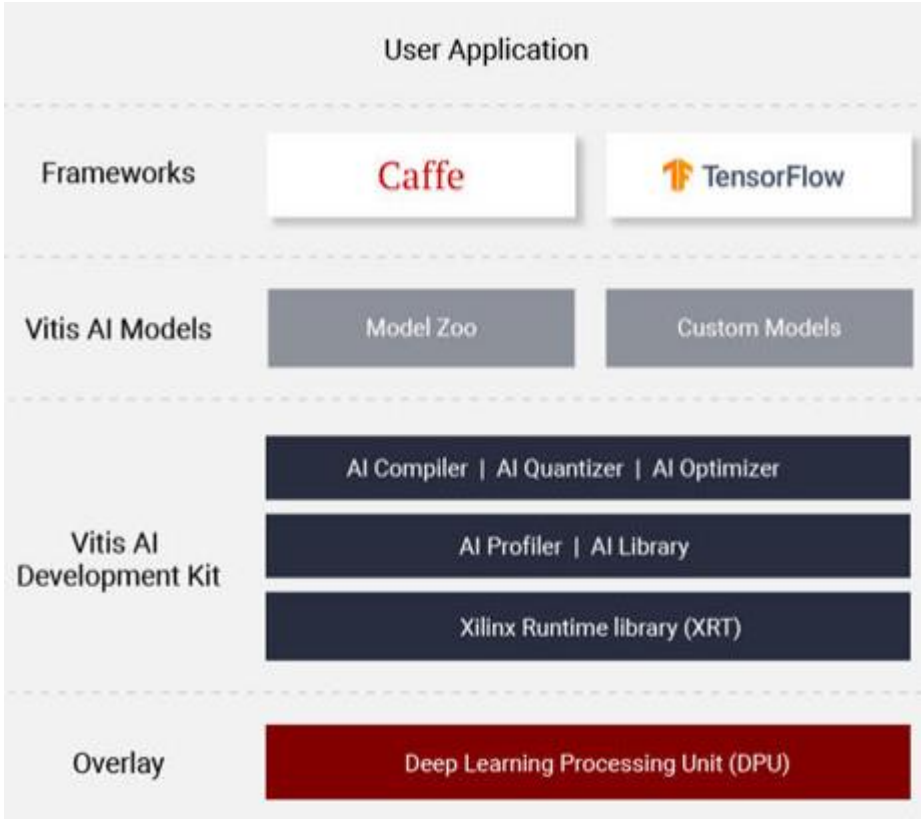
Running Examples On Cloud

- Working with Vitis AI via cloud is fairly straightforward
- Refer to this youtube tutorial https://www.youtube.com/watch?v=FCP0sN_HInw

Accelerating AI Vision on FPGA using Xilinx Vitis AI SDK

Vitis AI Development Kit

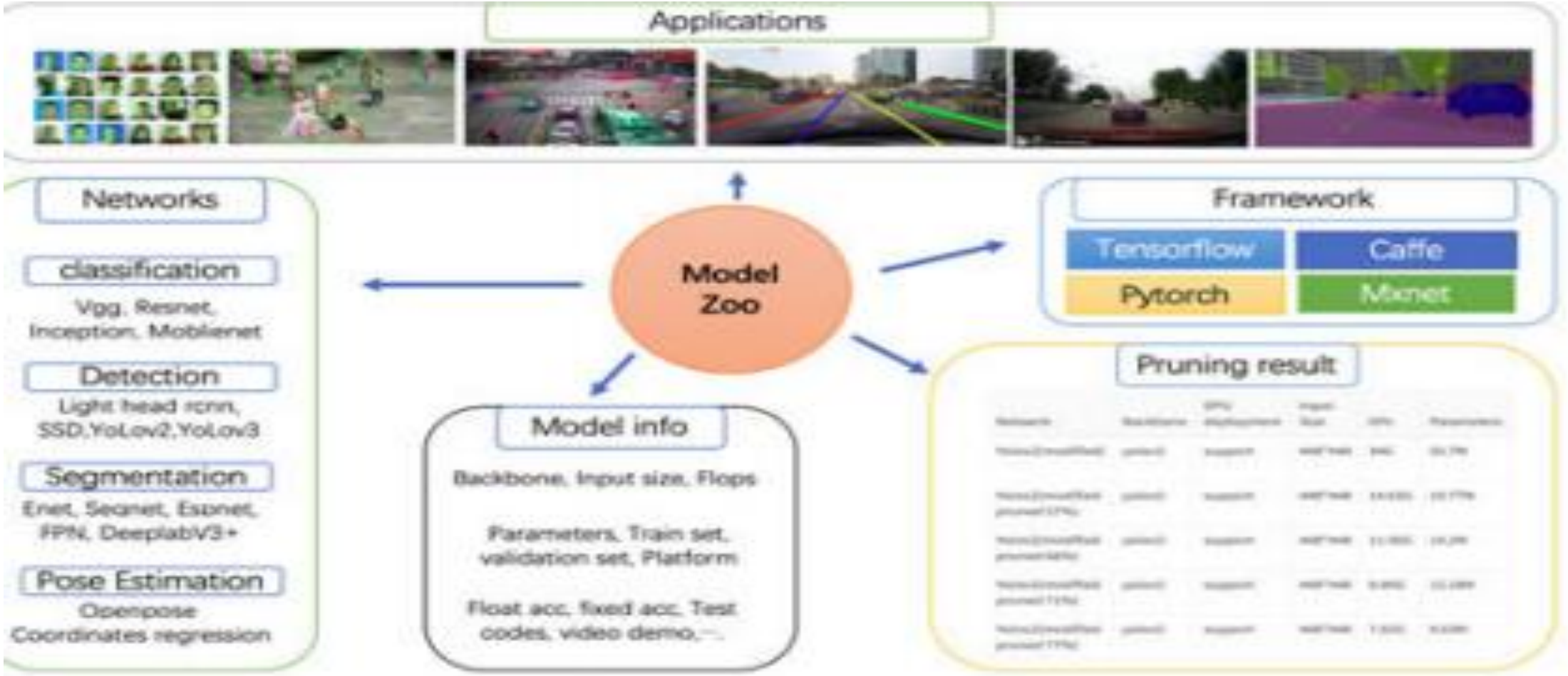
The Vitis™ AI development environment consists of the Vitis AI development kit, for the AI inference on Xilinx® hardware platforms, including both edge devices and Alveo™ accelerator cards



Vitis AI Development Kit

- It consists of optimized IP cores, tools, libraries, models, and example designs.
- Provides High efficiency and easy implementation of AI acceleration on Xilinx FPGA
- Enables users to develop DL applications without FPGA knowledge
- Provides a comprehensive set of pre-optimized models that are ready to deploy on Xilinx devices.
- The AI library offers unified high-level C++ and Python APIs for maximum portability from edge to cloud.

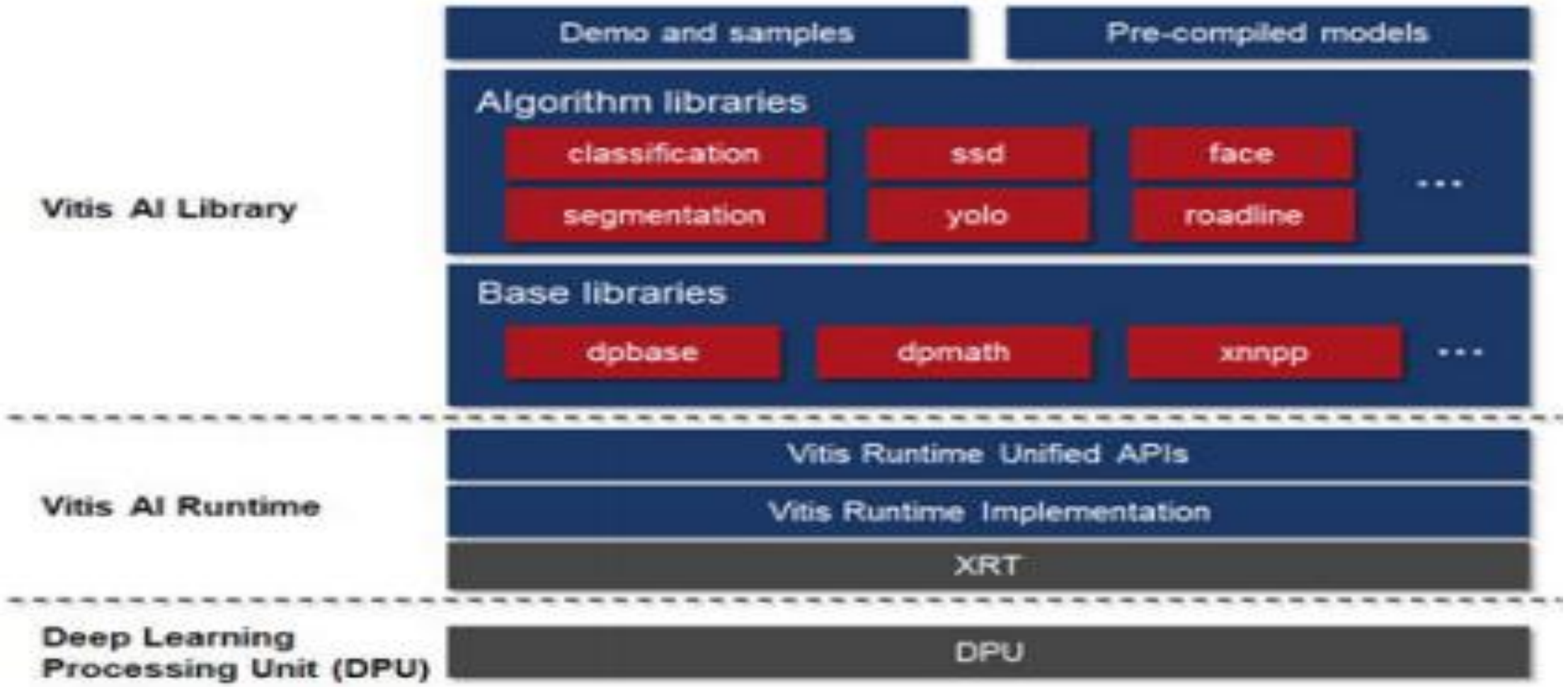
AI Model Zoo


Applications
Networks
classification
Vgg, Resnet,
Inception, Mobilenet
Detection
Light head rcnn,
SSD,YoLov2,YoLov3
Segmentation
Enet, Seanet, Espnet,
FPN, DeeplabV3+
Pose Estimation
Openpose
Coordinates regression
Model
Zoo
Model info
Backbone, Input size, Flops
Parameters, Train set,
validation set, Platform
Float acc, fixed acc, Test
codes, video demo,...
Framework
Tensorflow
Caffe
Pytorch
Mxnet
Pruning result

AI Model Zoo

- AI Model Zoo includes optimized deep learning models to speed up the deployment of deep learning inference on Xilinx platforms. These models cover different applications, including ADAS/AD, video surveillance, robotics, data center, etc. You can get started with these pretrained models to enjoy the benefits of deep learning acceleration. For more information, see https://github.com/Xilinx/Vitis-AI/tree/master/AI-Model-Zoo

AI Library


Demo and samples
Pre-compiled models
Algorithm libraries
classification
ssd
face
...
segmentation
yolo
roadline
Vitis AI Library
Base libraries
dpbase
dpmath
xnnpp
...
Vitis Runtime Unified APIs
Vitis AI Runtime
Vitis Runtime Implementation
XRT
Deep Learning Processing Unit (DPU)
DPU

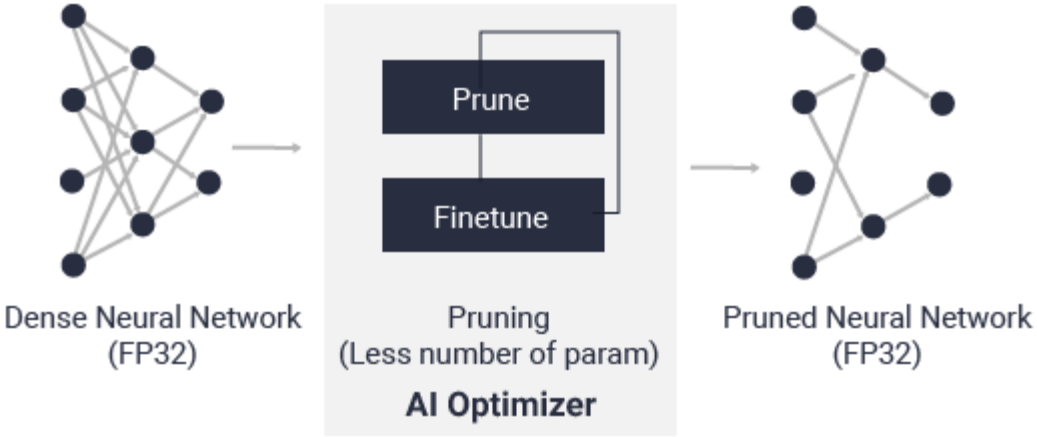
AI Library

- The Vitis AI Library is a set of high-level libraries and APIs built for efficient AI inference with Deep-Learning Processor Unit (DPU).
- The Vitis AI Library provides an easy-to-use and unified interface by encapsulating many efficient and high-quality neural networks.
- The Vitis AI Library allows you to focus more on the development of their applications, rather than the underlying hardware.

AI Optimizer

It can reduce model complexity by 5x to 50x with minimal accuracy impact.

Deep Compression takes the performance of your AI inference to the next level.

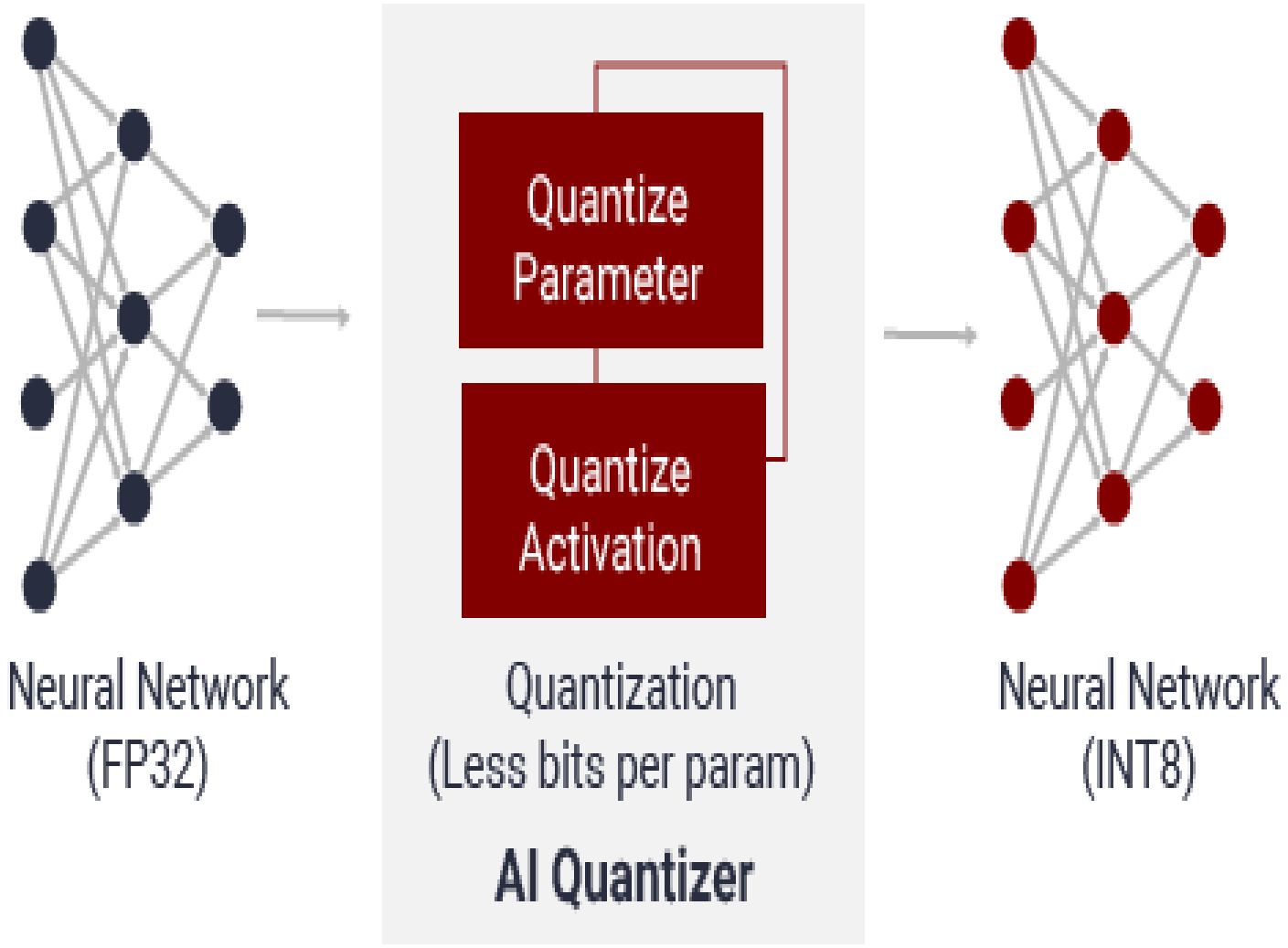


AI Quantizer

By converting the 32-bit floating-point weights and activations to fixed-point like INT8, the AI Quantizer can reduce the computing complexity without losing prediction accuracy.

The fixed-point network model requires less memory bandwidth, thus providing faster speed and higher power efficiency than the floating-point model.

By converting the data type, we can reduce the amount of memory required for execution



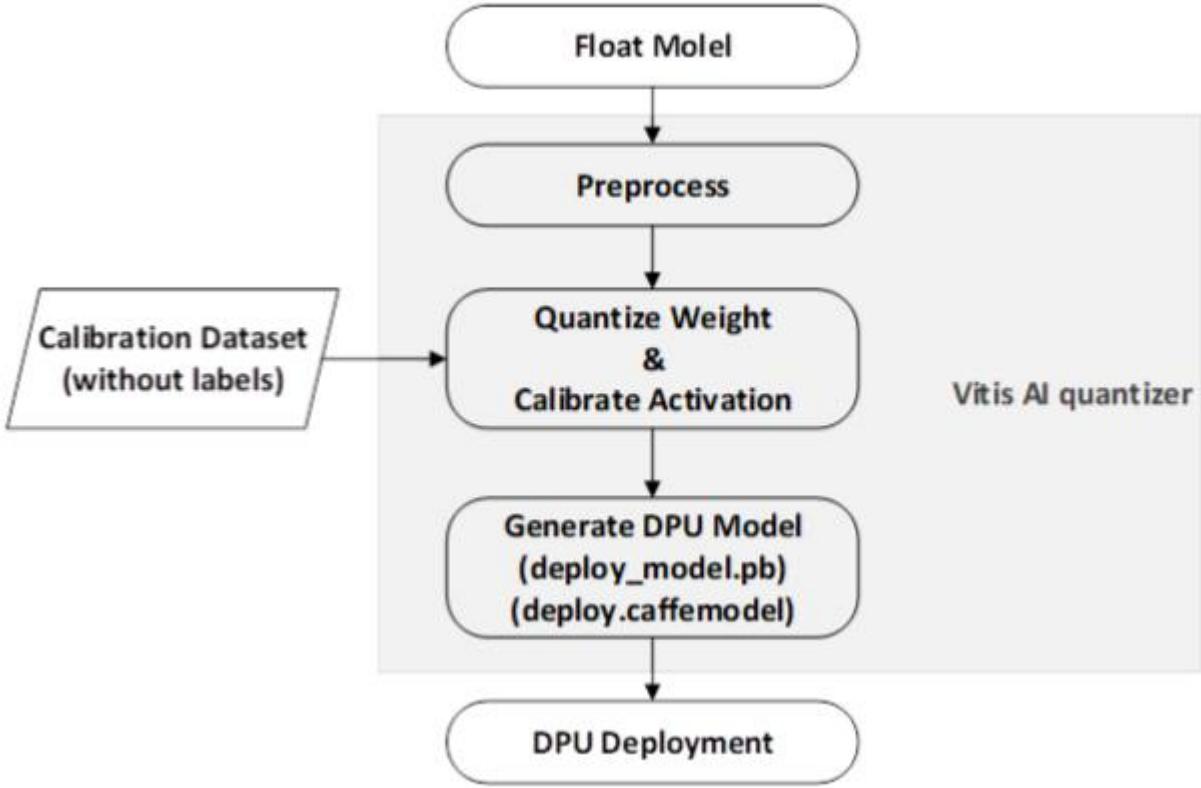
The overall model quantization flow is detailed in the following figure.

The Vitis AI quantizer takes a floating-point model as input (prototxt and caffemodel for the Caffe version, and frozen GraphDef file for the TensorFlow version), performs pre-processing (folds batchnorms and removes useless nodes), and then quantizes the weights/biases and activations to the given bit width.

To capture activation statistics and improve the accuracy of quantized models, the Vitis AI quantizer needs to run several iterations of inference to calibrate the activationsGenerally, the quantizer works well with 100–1000 calibration images. This is because there is no need for back propagation, the un-labeled dataset is sufficient.

After calibration, the quantized model is transformed into a DPU deployable model

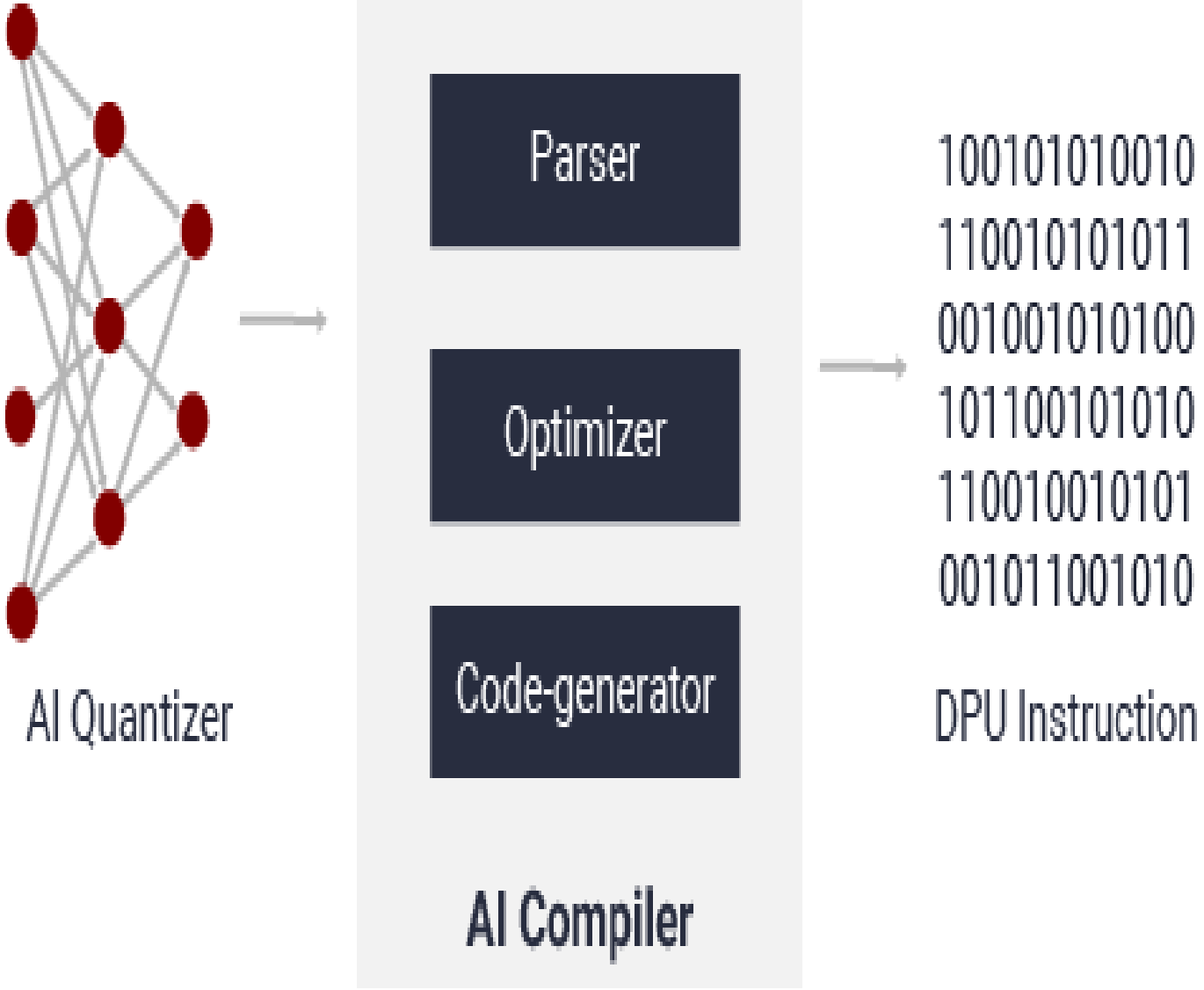
This model can then be compiled by the Vitis AI compiler and deployed to the DPU. The quantized model cannot be taken in by the standard vision Caffe or TensorFlow framework.



AI Compiler

Maps the AI model to a high-efficient instruction set and data flow. Also performs sophisticated optimizations such as layer fusion, instruction scheduling, and reuses on-chip memory as much as possible

It then performs multiple kinds of compilation optimizations and transforming techniques, including computation nodes fusion, efficient instruction scheduling, full reuse of DPU on-chip data, etc.

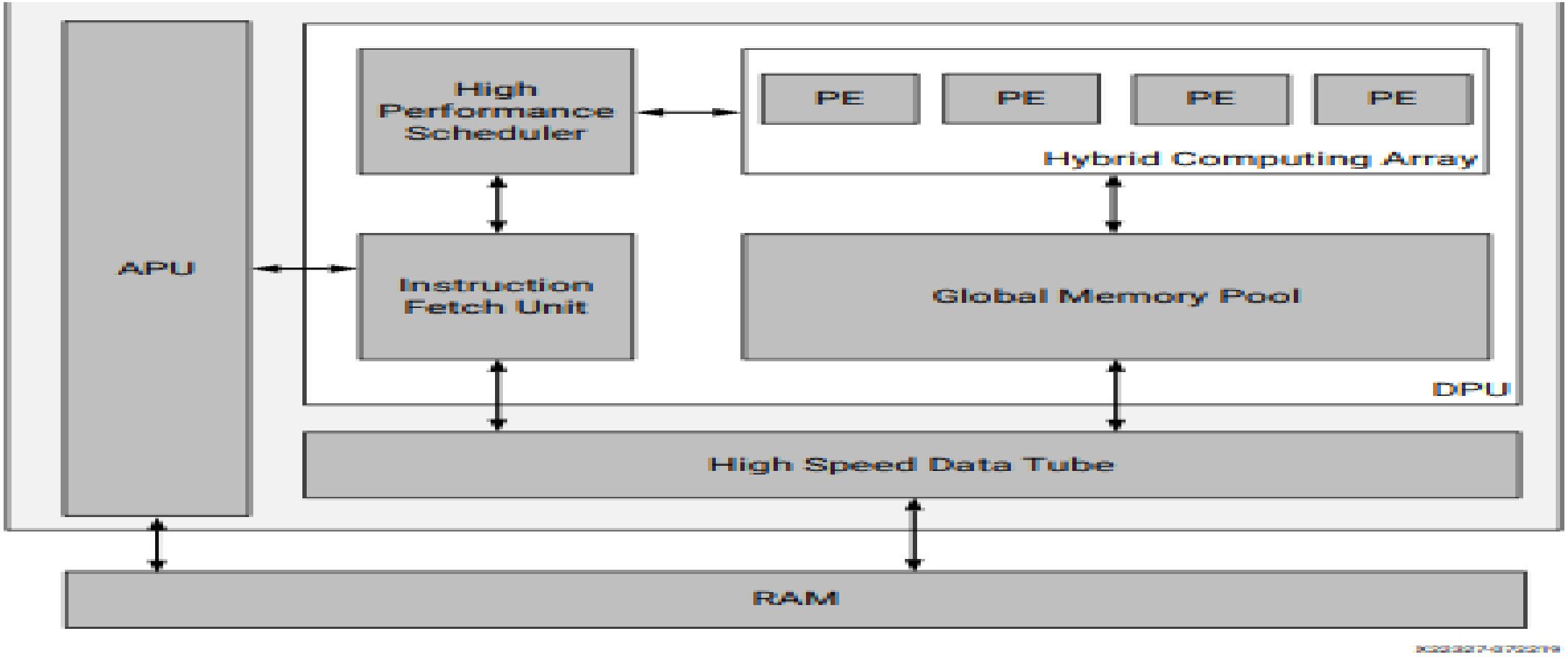


Deep Learning Processor Unit (DPU)

- Vitis AI offers a series of different DPUs for both embedded devices such as Xilinx Zynq-7000, Zynq UltraScale+ MPSoC, and Alveo enabling unique differentiation and flexibility in terms of throughput, latency, scalability, and power.
- Allows efficient implementation of many deep learning networks.
- The Xilinx® Deep Learning Processing Unit (DPU) is a configurable computation engine dedicated to convolutional neural networks. The degree of parallelism utilized in the engine is a design parameter and can be selected according to the target device and application. It includes a set of highly optimized instructions, and supports most convolutional neural networks, such as VGG, ResNet, GoogLeNet, YOLO, SSD, MobileNet, FPN, and others

- https://www.xilinx.com/support/documentation/ip_documentation/dpu/v3_1/pg338-dpu.pdf

DPU TOP LEVEL DIAGRAM



where,

- APU – Application Processing Unit
- PE – Processing Engine
- DPU – Deep Learning Processing Unit
- RAM – Random Access Memory

All these Vitis AI Devkit Components provides an efficient compilation of the Deep learning networks to integrate into the FPGA memory

The Xilinx FPGA devices and evaluation boards supported by Vitis AI development kit v1.0 release are:

- Cloud: Xilinx Alveo™ cards U200, U250
- Edge: Xilinx MPSoC evaluation boards ZCU102, ZCU104

Model Deployment Overview

The Vitis™ AI toolchain provides an innovative workflow to deploy deep learning inference applications on the DPU with the following four steps

1. Quantize the neural network model.
2. Compile the neural network model.
3. Program with Vitis AI programming interface.
4. Run and evaluate the deployed DPU application

Accelerating Subgraph with ML Frameworks

- Partitioning is the process of splitting the inference execution of a model between the FPGA and the host. Partitioning is necessary to execute models that contain layers unsupported by the FPGA. Partitioning can also be useful for debugging and exploring different computation graph partitioning and execution to meet a target objective

Partitioning Steps

1. Loading the original graph
2. Partitioning : In this step the subgraph specified by startnode and finalnode sets is analyzed for FPGA acceleration. This is done in multiple phases. Each supported partition subgraph is stored for visualization and debug purposes
3. Freezing the modified graph
4. Run natively in Tensorflow

- If model is not properly frozen, the compiler might fail optimizing some operations such as batchnorm.

DEBUGGING AND PROFILING

With the combined use of these tools, users can conduct DPU debugging and performance profiling independently.

- DSight: DPU performance profiling.
- DDump: Parsing and dumping over DPU ELF file, shared library, and hybrid executable
- DExplorer: Runtime mode management and DPU signature checking.
- DLet: Parsing DPU Hardware Handoff file and generated DPU configuration file.

AI Profiler

The performance profiler allows programmers to perform an in-depth analysis of the efficiency and utilization of your AI inference implementation.

- After the model is deployed on edge DPU, perhaps the running results are not as desired, running into a lower accuracy issue. Under this situation, the users can check the model’s accuracy after quantized by Vitis AI quantizer

- Xilinx Vitis AI User Manual: https://www.xilinx.com/support/documentation/sw_manuals/vitis_ai/1_0/ug1414-vitis-ai.pdf
- Xilinx ML SUITE (Similar Version) :

➢ https://www.xilinx.com/products/acceleration-solutions/xilinx-machine-learning-suite.html

➢ https://github.com/Xilinx/ml-suite