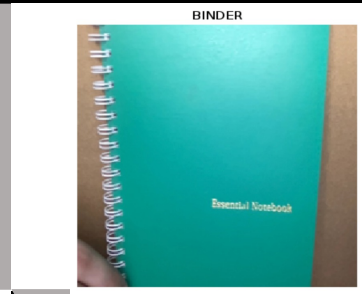
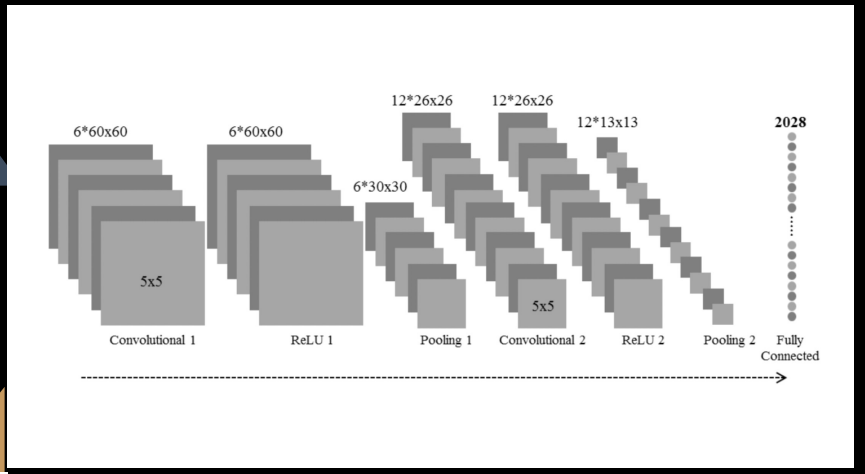
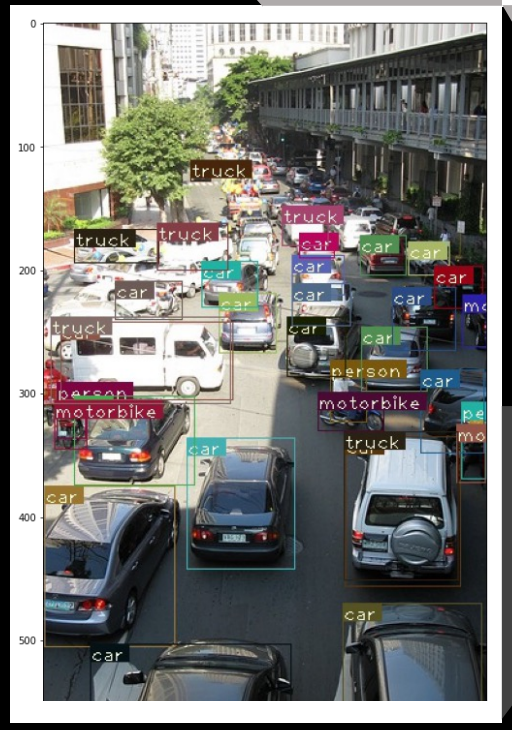






Activity 6: Finding Objects and Scenes

Submitted by: Mark Jeremy G. Narag
Student Number: 2014-64423
Program: PhD Physics

Physics 301
2nd Sem AY 2021-22

Objectives

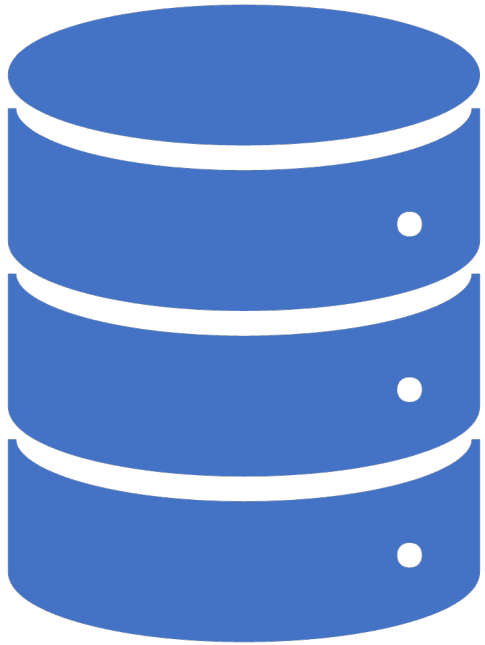
- Open and write videos using Matlab
- Use pretrained models to identify objects in a streaming video.

Procedure

1. Modify the code above to allow object identification in a video file.
2. Explore other contributed deep learning neural networks.
3. Bonus worthy : Figure out equivalent video processing tools in Python
4. Bonus worthy : Investigate contributed deep learning NN for action detection

I divided this activity into four sections:

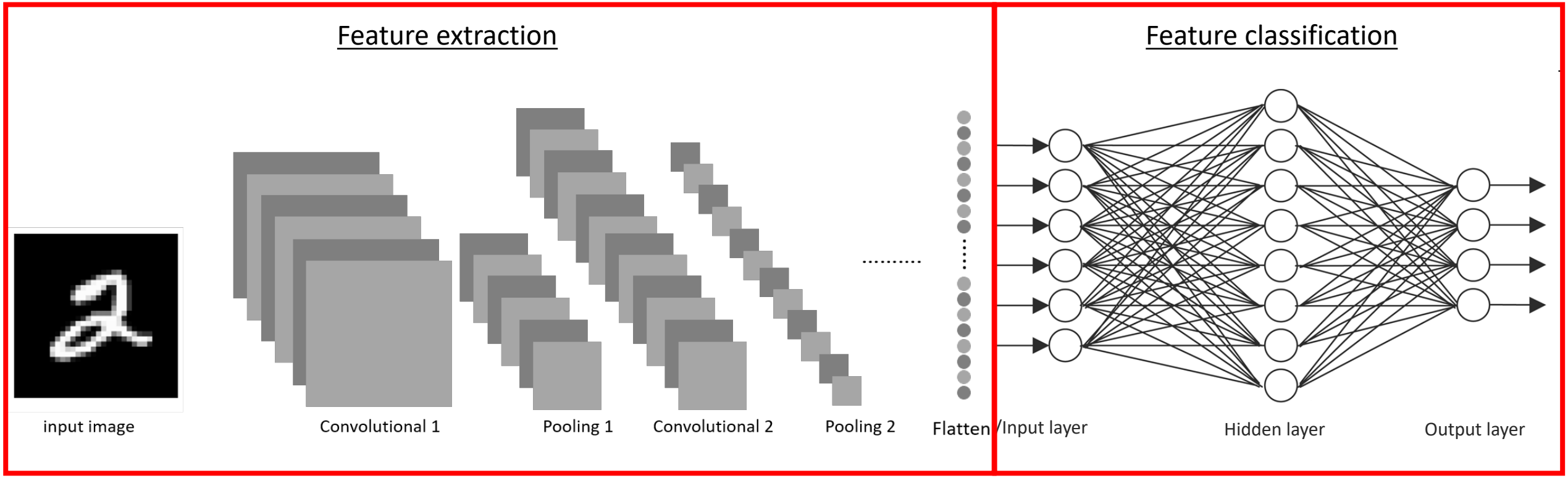
1. Convolutional Neural Network
2. Matlab live object recognition
3. Matlab object recognition from video
4. Python object recognition



1. Convolutional neural networks. *How does it work?*

Before we start, let's first learn how does a deep neural network works. How can it identify an object?

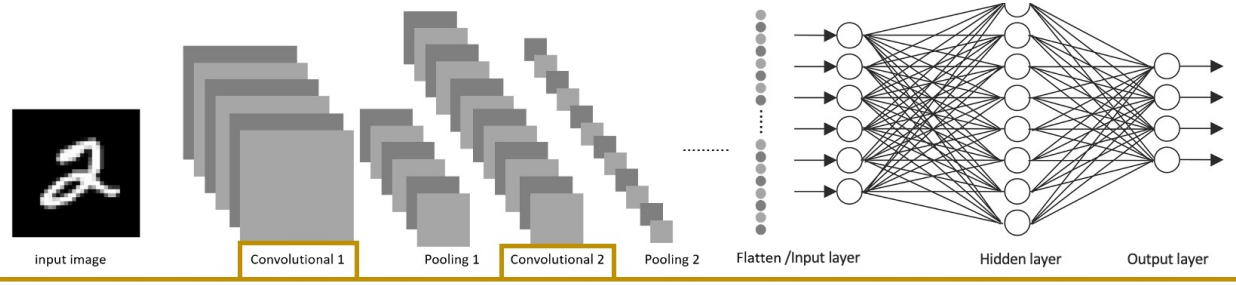
In brief explanation, a deep neural network consists of series of convolution and pooling layers that extract features from an input image. These layers are then followed by artificial neural network in which the classification takes place. The output of the last layer gives the final class of the input image. The diagram below illustrates a simple deep neural network [1]. In particular, working on images, we call this convolutional neural networks (CNN).



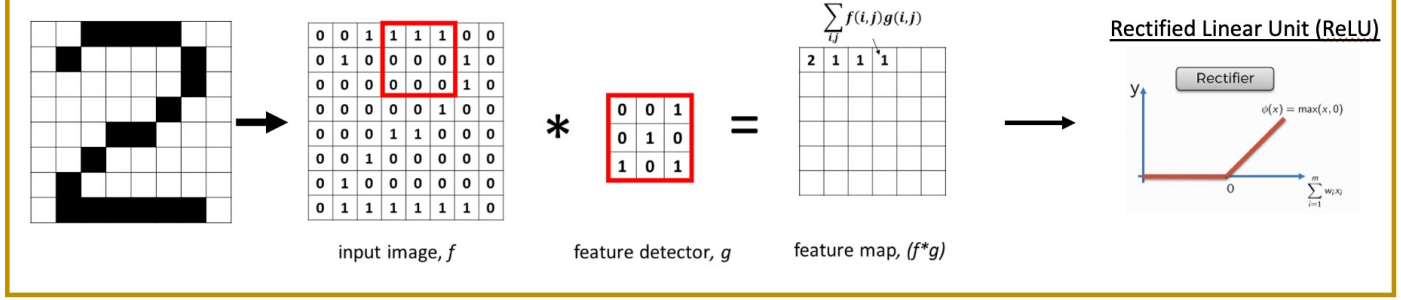
[1] Narag, M.J., Unraveling the relationship: Discovering artistic influences of painters using Convolutional Neural Network. MS Thesis. National Institute of Physics, University of the Philippines Diliman, 2021.

1. Convolutional neural networks. *How does it work?*

Mathematically, a convolution operator (Equation 1) is applied to the input image. As can be seen from the figure below, input image *f* is convolved with **feature detector *g***. The feature detector, also called filter, is a combination of 1s and 0s. Several of these filters are convolved with the input image. The output is called **feature map *(f*g)***. An activation function, rectified linear unit, is applied to the solve the nonlinearity of the feature map since images are non-linear in nature.



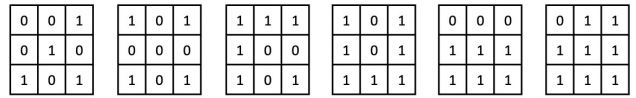
operation: $$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2] \quad (1)$$




input image

*

feature detectors:



convolved images:

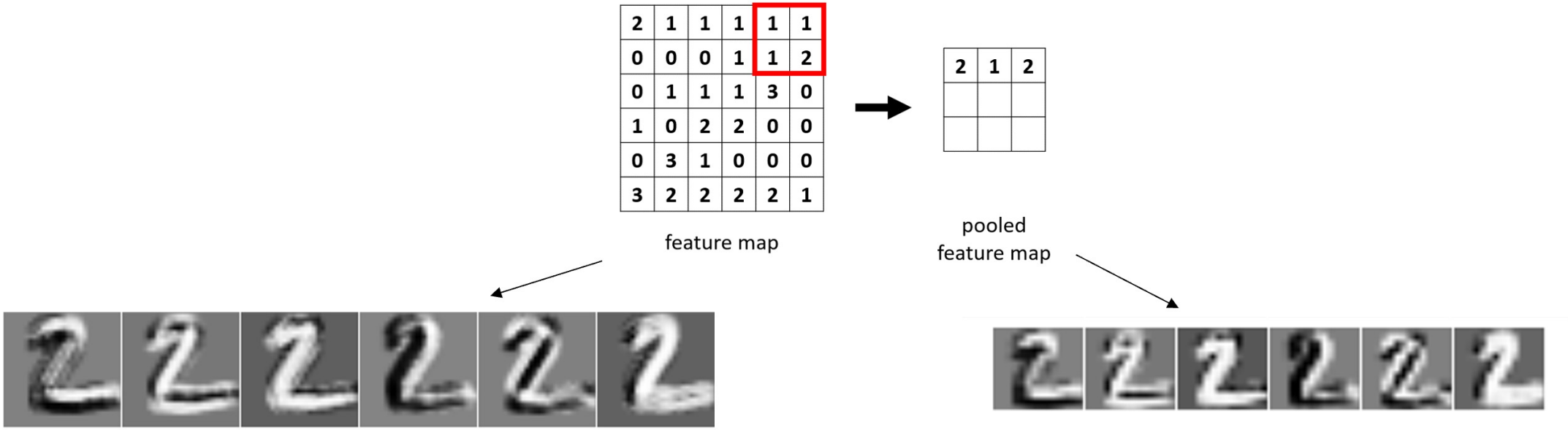
convolved image + ReLU:

[1] Narag, M.J., Unraveling the relationship: Discovering artistic influences of painters using Convolutional Neural Network. MS Thesis. National Institute of Physics, University of the Philippines Diliman, 2021.

1, Convolutional neural networks. *How does it work?*

Typically, the convolution layers are followed by **pooling layers**. *Why do we need pooling?* Pooling is applied to ***achieve spatial invariance by reducing the resolution*** of the feature maps. Spatial invariance is the shifting or distortions between similar images. For example we have different handwritten images of number ”2”, although the images are different, it is still recognizable as number ”2” because the general form of the image is still there. Even if we down sampled the feature map using max pooling, the general form will still be recognizable. We can see this in the figure below [1].

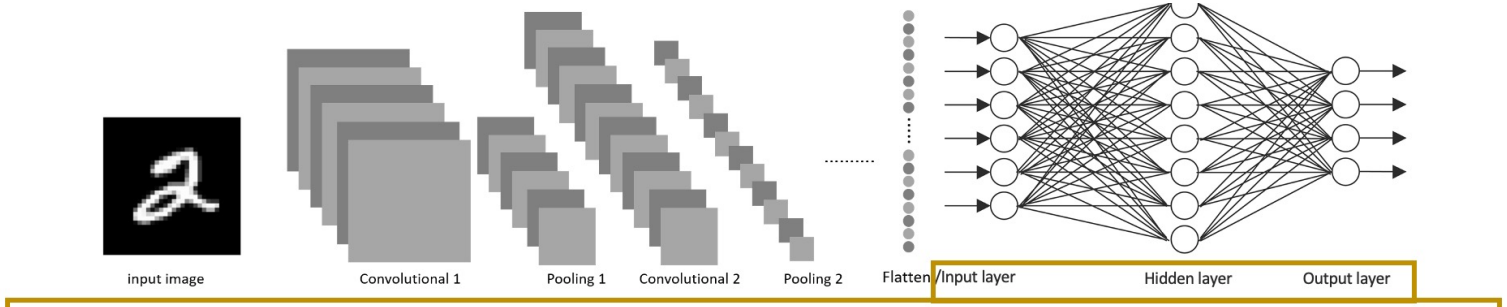
There are several types of pooling – max pooling, mean pooling, etc. The example below is max pooling where we take the maximum value per slide.



[1] Narag, M.J., Unraveling the relationship: Discovering artistic influences of painters using Convolutional Neural Network. MS Thesis. National Institute of Physics, University of the Philippines Diliman, 2021.

1. Convolutional neural networks. *How does it work?*

The two layers – convolution and pooling, extracts meaningful features from the input image. The classification happens on the last layer of our CNN – the **artificial neural network layer (ANN)**. ANN consists of several interconnected neurons in the hidden layer. Its input is the concatenated feature map from the feature extraction stage and the final layer is the classification label. Mathematically, weights are multiplied to each neuron and it gets updated until we reach our desired accuracy during the training stage. The weights are updated through **back propagation.**



BACK PROPAGATION

Error function: $$E = \frac{1}{2N}\sum_{i=1}^{N}(\vec{y_i} - y_i)^2 \quad (2)$$

y_i target value

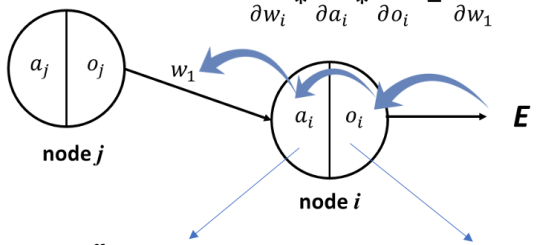
$\hat{y}_i$ output value

N Set of input-output pairs

Updated weight: $$w_{i,j}^{k+} = w_{i,j}^{k} + \eta\left(\frac{\partial E}{\partial w_{i,j}^{k}}\right) \quad (3)$$

η learning parameter

$$\frac{\partial a_i}{\partial w_i} * \frac{\partial o_i}{\partial a_i} * \frac{\partial E}{\partial o_i} = \frac{\partial E}{\partial w_1}$$



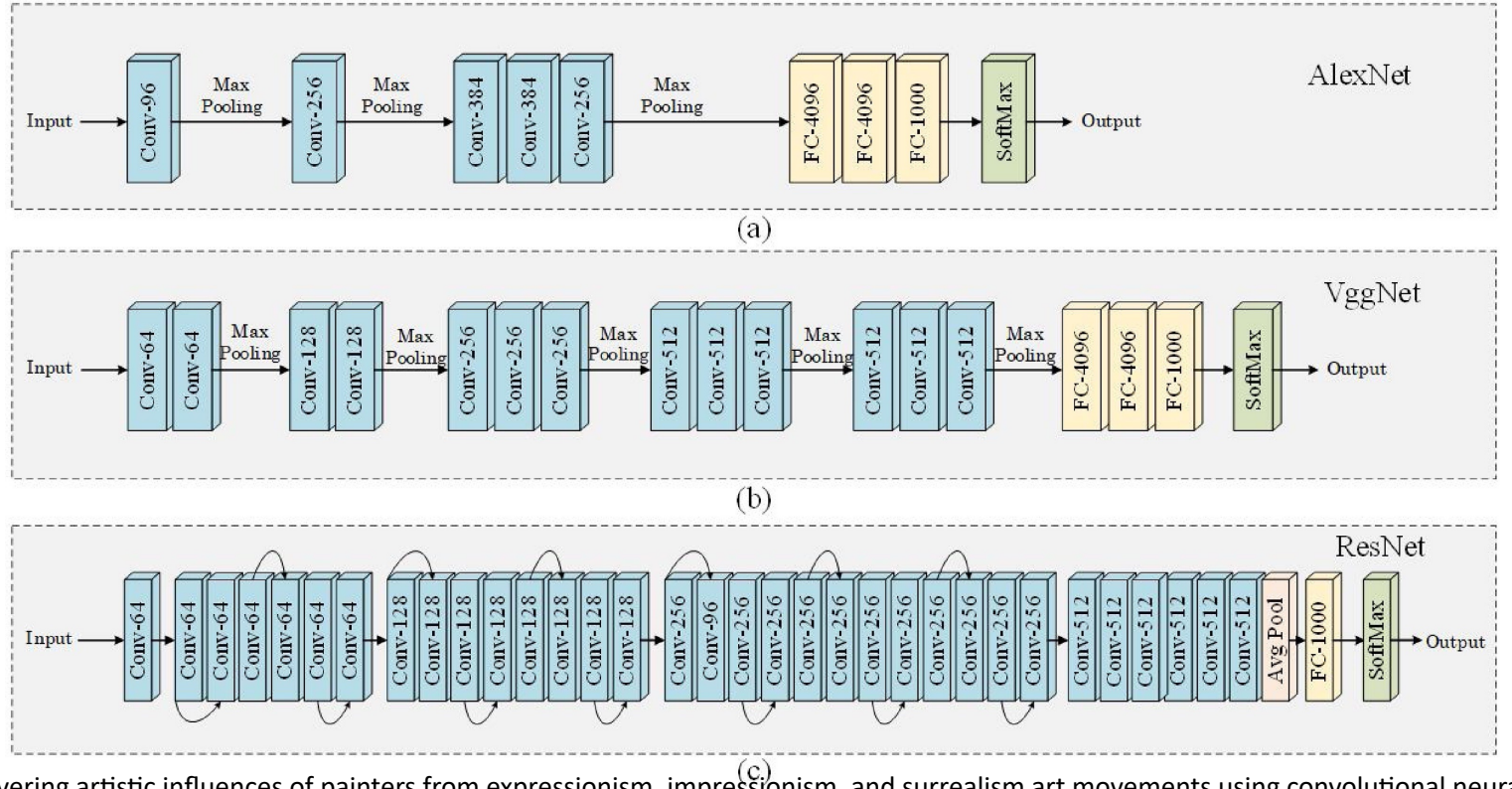
$$a_i^k = \sum_{j=0}^{r_{k-1}} w_{ij}^k \, o_j^{k-1} \qquad o_i^k = g(a_i^k)$$

[1] Narag, M.J., Unraveling the relationship: Discovering artistic influences of painters using Convolutional Neural Network. MS Thesis. National Institute of Physics, University of the Philippines Diliman, 2021.

1. Convolutional neural networks. *How does it work?*

There are many pretrained CNN models available on the internet. When we say pretrained, it means that the weights are already predetermined, we don't need to train it from scratch. Usually, these pretrained models are trained in **ImageNet** - a largescale database of images with annotated labels.

The three most commonly used CNN models in AI researches are Alexnet, VGG, and ResNet. Below is the architecture of **Alexnet**, **VGG16**, and **Resnet50** [2]. Modified from [3]. The models used in this image activity are VGG19 and Resnet101, which only means that there are more layers that the one presented in this figure (19 layers in VGG19 and 101 layers in Resnet101).



[2] M. J. Narag and M. Soriano. "Discovering artistic influences of painters from expressionism, impressionism, and surrealism art movements using convolutional neural network", Journal of Cultural Heritage, 51 (2021), pp. 182-193. https://doi.org/10.1016/j.culher.2021.08.015

[3] M. Talo, O. Yildirim, U.B. Baloglu, G. Aydin, U.R. Acharya, Convolutional neural networks for multi-class brain disease detection using MRI images, Comput. Med. Imaging Graph. 78 (2019) 101673, doi:10.1016/j.compmedimag.2019. 101673.

2. Matlab live object recognition

In the past years, artificial intelligence (A.I.) has been extensively explored by researchers. In effect, we have many pretrained deep neural network models available for public use. Matlab has 19 readily available models to date [4]. They are pretrained in Imagenet.

In this activity, we will focus on three (3) well-known CNN models as mentioned earlier. We have Alexnet *(line 2),* VGG19 *(line 3),* and ResNet101 *(line 4).* Note that for each models, the fixed input image varies so we need to resize the image first before feeding it to the classifier.

```
1   cam = webcam;
2   nnet = alexnet; %image size [227, 227]
3   %nnet = vgg19; %image size [224,224]
4   %nnet= resnet101; %image size [224,224]
5   %%
6   for i = 1:15
7       vidFrame = cam.snapshot;
8       pic = imresize(vidFrame,[224,224]);
9       label =  classify(nnet,pic);imshow(pic)
10      title(upper(char(label)))
11  end
```

Line 1 opens the webcam four our live object recognition.

Line 6 gives the time (in seconds) of how long the camera is used. We can alter this anytime we want.

Line 7 takes the image for each second.

Line 8 resizes the image according to the model used.

Line 9 classifies the image and gets its label from the model.

Remember that the classification time varies of each model. Typically, the longer the architecture of the model, the slower it is to classify the image.

[4] https://www.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html

2. Matlab live object recognition

Here are the results for each model on six objects of interest in a **controlled setup**. I used a box as a background.

Alexnet

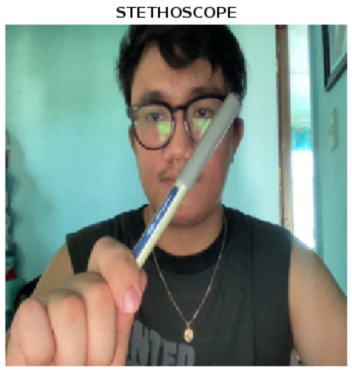
Resnet101

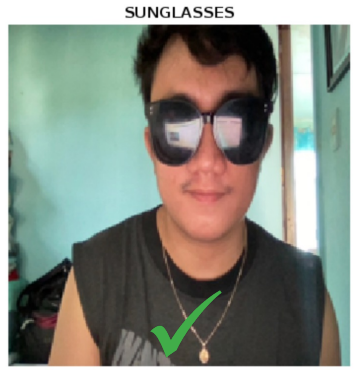
VGG19

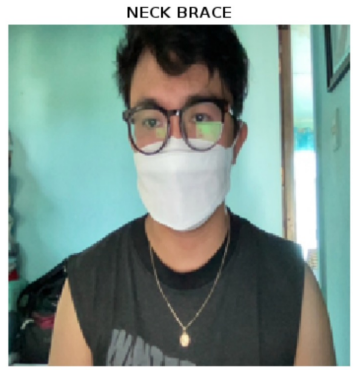
2. Matlab live object recognition

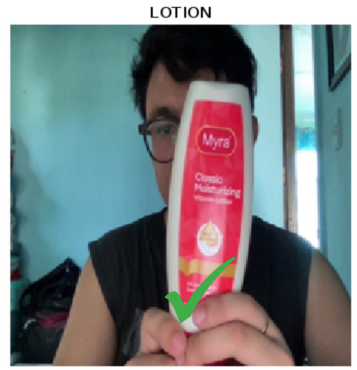
Here are the results for each model on six objects of interest in **uncontrolled setup (no plain background)**

Alexnet

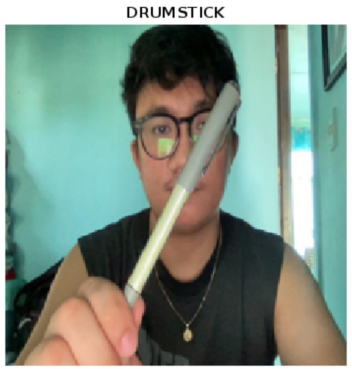


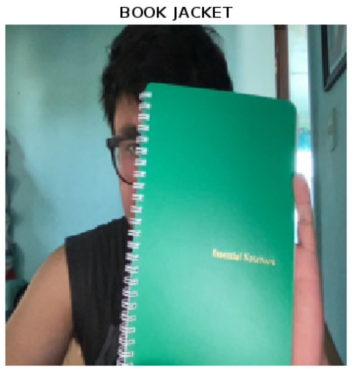


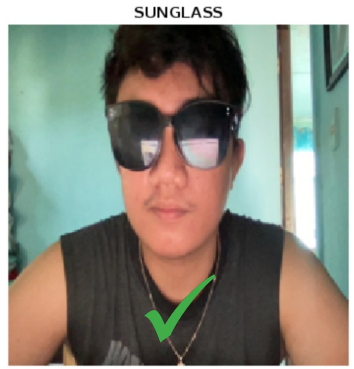




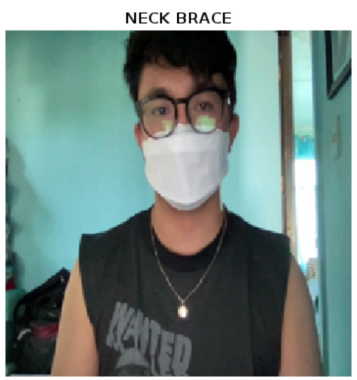
Resnet101

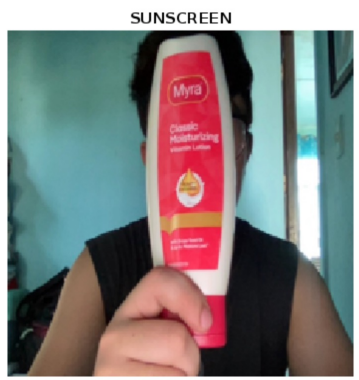






VGG19



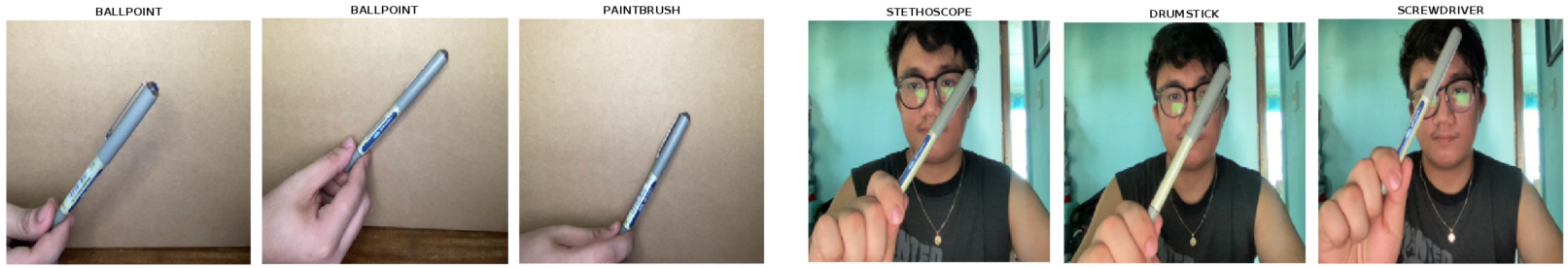


2. Matlab live object recognition

Observation:

1. Unofficially, from our six objects, Alexnet outperforms ResNet and VGG with a score 7/12. Resnet has 6/12 and VGG has 4/12.

2. We can see that the models, regardless of what it is, have higher accuracy for controlled setup where the background is plain. Case in point, the ballpen:



Except for facemask (in Resnet101) where it is recognized when worn.

3. The lotion is commonly misclassified as sunscreen. It is a no surprise since lotion and sunscreen are very much alike. The shape itself is the same.

4. The notebook was never labeled as notebook but instead labelled as binder. ImageNet may not have a notebook in their database but instead put it under binder.

3. Matlab object recognition from video

Instead of camera, let's try to classify the images/frames in a video! Remember that video is just series of images.

Video_Tracker.m | vid

1×1 VideoReader

Property	Value	Size	Class
Duration	36.1312	1×1	double
Name	'Video Tracker.mov'	1×17	char
Path	'/MATLAB Drive/6_Video…	1×29	char
BitsPerPixel	24	1×1	double
FrameRate	30.0391	1×1	double
Height	720	1×1	double
NumFrames	1084	1×1	double
VideoFormat	'RGB24'	1×5	char
Width	1080	1×1	double
Tag	''	0×0	char
UserData	[]	0×0	double
CurrentTime	36.1300	1×1	double

The video I created is 36.1312 seconds in total with frame-per-second (fps) of 30.

Doing the math, we have a total of 1084 frames in the video. Our deep neural network models will classify each of these frames! This is of course a lot to process for Matlab so we might as well just skip some frames.

For two seconds, the 60 frames there probably have minimal changes from one another so let's try to make the model classify the frame for every two seconds, skipping 60 frames.

3. Matlab object recognition from video

Below is the code to read a video and classify the frames using a deep neural network mode:

```
12  %% object recognition from a video
13  nnet = alexnet;
14  vid = VideoReader('Video Tracker.mov');
15  % classify the frames
16  n=60; %number of frames we want to skip
17  t = tiledlayout(3,6);
18  while hasFrame(vid)
19      vidFrame = readFrame(vid);
20      vid.CurrentTime = vid.CurrentTime + n/vid.FrameRate;
21      vidFrame = imresize(vidFrame,[227,227]);
22      label =  classify(nnet,vidFrame);
23      nexttile; imshow(vidFrame); title(upper(char(label)))
24  end
```

Line 14 reads the video file.

Line 16 is the number of frames we want to skip.

Line 18-24 is a while loop that loads all the frames in the video file. To skip the frame, we actually skip the time in the video as can be seen in line 20. In this line, it computes the time (in seconds) to skip the video based from the set number of frames to skip – **n/vid.FrameRate.** Based from the previous slide, the FrameRate is around 30fps. Thus, **n/vid.FrameRate = 60/30=2s.** We are then skipping the frames every 2 seconds. The next lines are the same as we did in the previous.

3. Matlab object recognition from video

Results for Alexnet:

3. Matlab object recognition from video

Results for VGG19:

3. Matlab object recognition from video

Results for ResNet101:

3. Matlab object recognition from video

Results for Alexnet

Let's try skipping 30 frames instead of 60

Observations:

- I was classified as a suit, perhaps person/human are not included in the model labels.
- Model was not able to classify the pen. Paintbrush is close.
- Fails to classify the notebook as well. Dumb bell and punching bag is too far fetched.
- Fails to classify the mug
- Lotion classified as Ice Lolly
- Sunglasses was correctly classified in most frames.
- Face mask was labeled as neck brace.

3. Matlab object recognition from video

Results for VGG19

Skipping 30 frames

Observations:

- I was classified as abaya (cloak used by Islam women).
- Sometimes, without the object, it is my sunglasses that was labeled.
- I am now sure that person/human are not included in the model labels.
- Model was not able to classify the pen but drumstick and screwdriver is close.
- Fails to classify the notebook as well.
- Successful to classify the mug and sunglasses!
- Lotion classified as Ice Lolly and hair spray.
- Face mask was labeled as neck brace. Oxygen mask came close.

3. Matlab object recognition from video

Results for ResNet101

Skipping 30 frames

Observations:

- I was classified as iron for unknown reason.
- Model was not able to classify the pen but drumstick and paintbrush is close.
- Fails to classify the notebook as well. But was able to classify it as binder in one of the frames.
- Successful to classify the mug, sunglasses, and even the mask!
- Lotion classified as Ice Lolly and hair spray.

So far, ResNet101 outperforms the other two models!

4. Python object recognition - *YOLO version 3 using Pytorch*

There are many publicly available codes for object detection, both for Python and Matlab users. I have been in the A.I. track for several years so I know some of the popular algorithms. One of them is **YOLO (You Only Look Once)** – a family of computer vision models for real time object detection which was introduced in 2016 by Redmon, Divvala, Girshick, and Farhadi. Their novel architecture was even awarded as OpenCV People's Choice Award in the same year!

Read their paper here ➡ arXiv:1506.02640 [5]

Their paper have over 20k plus citations! For a brief overview, below is the architecture of their very first model [5]. They have five versions of the model as of this year 2022.

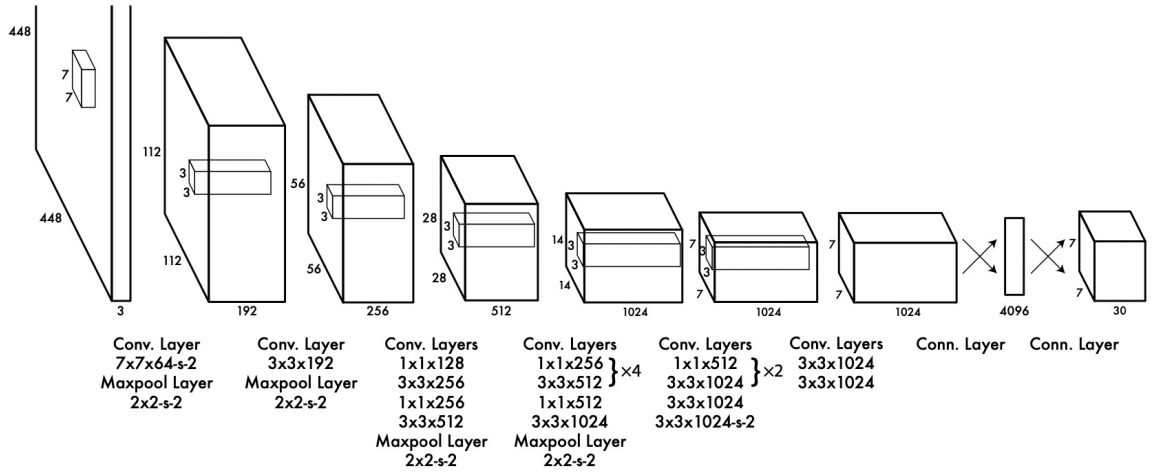


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

[5] Redmon, J., Divvala, S.K., Girshick, R.B., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788.

4. Python object recognition - *YOLO version 3 using Pytorch*

The best thing about YOLO is it can identify several objects in a picture. Unlike the models we used previously which gives singular label for the whole image.

Below is a screenshot of their paper which shows the result of their model [5].

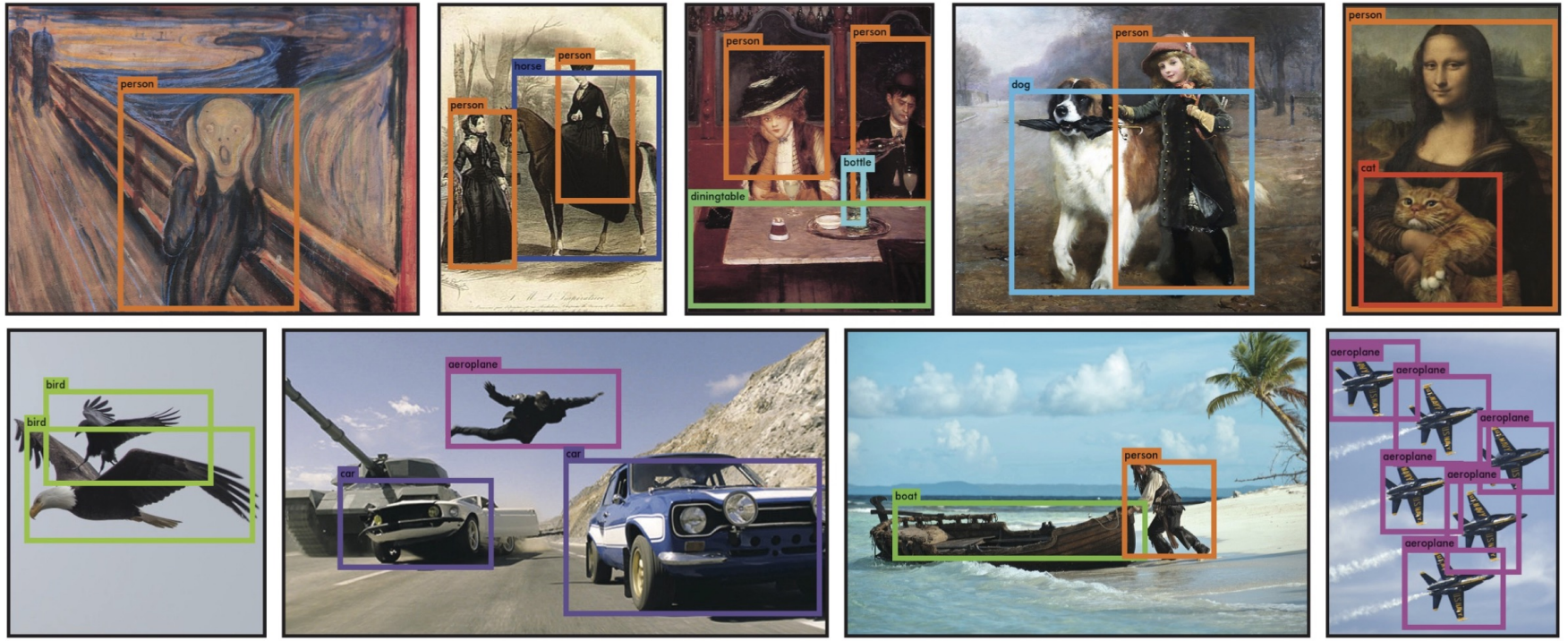


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.

[5] Redmon, J., Divvala, S.K., Girshick, R.B., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788.

4. Python object recognition - *YOLO version 3 using Pytorch*

Of course, the more popular the model becomes, the higher is its influence on A.I. researches in using their models and testing it on their own dataset. Gladly, the authors of YOLO created a website that guides researchers on how to implement their model (link: http://pjreddie.com/yolo/).

While there are many available codes online, not all of them will work on your computers. Most of them, if not all, will require you first to download and install many packages! It may even fail if your laptop can't take the processing energy. It is a tedious work. For someone with several experience regarding this, **I highly recommend using GOOGLE COLAB!**

Colab allows us to write and execute python code online! Its interface is similar to Jupyter. It is *easier* and *faster* to install packages here! Google colab is a life saver for someone who has a low-budget computer, it made me finish my graduate thesis!

Many developers use Google Colab. Here is one that I found online that implements YOLO version 3 using Pytorch executed in Google Colab: https://colab.research.google.com/github/tugstugi/dl-colab-notebooks/blob/master/notebooks/YOLOv3_PyTorch.ipynb#scrollTo=P6ETX8scB7oj

We will use their algorithm as an example. There are many more online!

4. Python object recognition - *YOLO version 3 using Pytorch*

Here is the screenshot of the code from google colab [6]

Install ayooshkathuria/pytorch-yolo-v3

```
[ ]  1 import os
     2 from os.path import exists, join, basename, splitext
     3
     4 git_repo_url = 'https://github.com/ayooshkathuria/pytorch-yolo-v3.git'
     5 project_name = splitext(basename(git_repo_url))[0]
     6 if not exists(project_name):
     7   # clone and install dependencies
     8   !git clone -q $git_repo_url
     9   #!cd $project_name && pip install -q -r requirement.txt
    10
    11 import sys
    12 sys.path.append(project_name)
    13 import time
    14 import matplotlib
    15 import matplotlib.pylab as plt
    16 plt.rcParams["axes.grid"] = False
```

Download official YOLO v3 pretrained weights

```
[ ]  1 if not exists('yolov3.weights'):
     2   !wget -q https://pjreddie.com/media/files/yolov3.weights
```

Detect objects on a test image

First, dowload a test image from internet:

Paste the link of your image here!

```
[ ]  1 IMAGE_URL = ' '
     2
     3 image_file = basename(IMAGE_URL)
     4 !wget -q -O $image_file $IMAGE_URL
     5 plt.imshow(matplotlib.image.imread(image_file))
```

Execute `detect.py` on that image and show the result:

```
[ ]  1 !cd pytorch-yolo-v3 && python detect.py --weights ../yolov3.weights --images ../$image_file --det ..
     2
     3 plt.figure(figsize=(20, 15))
     4 plt.imshow(matplotlib.image.imread('det_%s' % image_file))
```

The code only has three parts. First section is installing the necessary packages. Second section is getting the pretrained weights so we don't need to train the model from scratch. This technique is called ***transfer learning***. According to their paper, they pretrained it on ImageNet too!

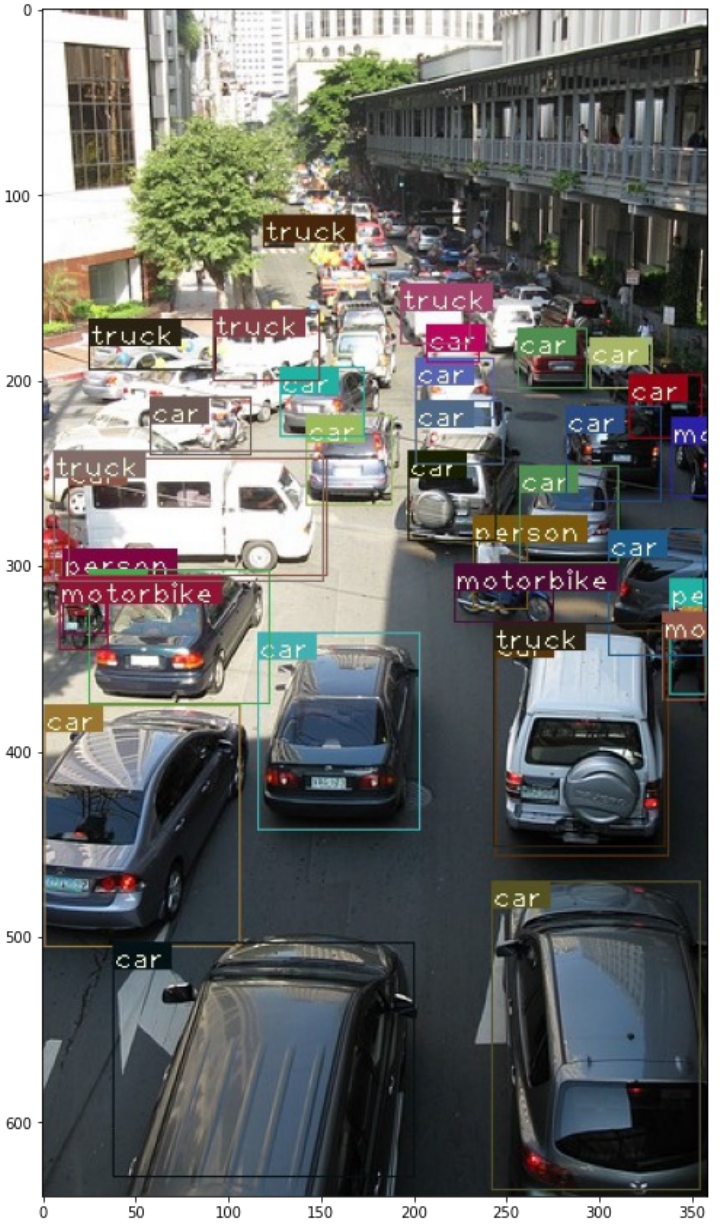
The last section is detecting our own images. You just need to put the link of your image you want to test in line 1 (red box). You may **upload your test images in github** and get the link there!

In GoogleImages, just right click the image of interest and select "Copy Image Address"

[6] https://colab.research.google.com/github/tugstugi/dl-colab-notebooks/blob/master/notebooks/YOLOv3_PyTorch.ipynb#scrollTo=P6ETX8scB7o

4. Python object recognition - *YOLO version 3 using Pytorch*

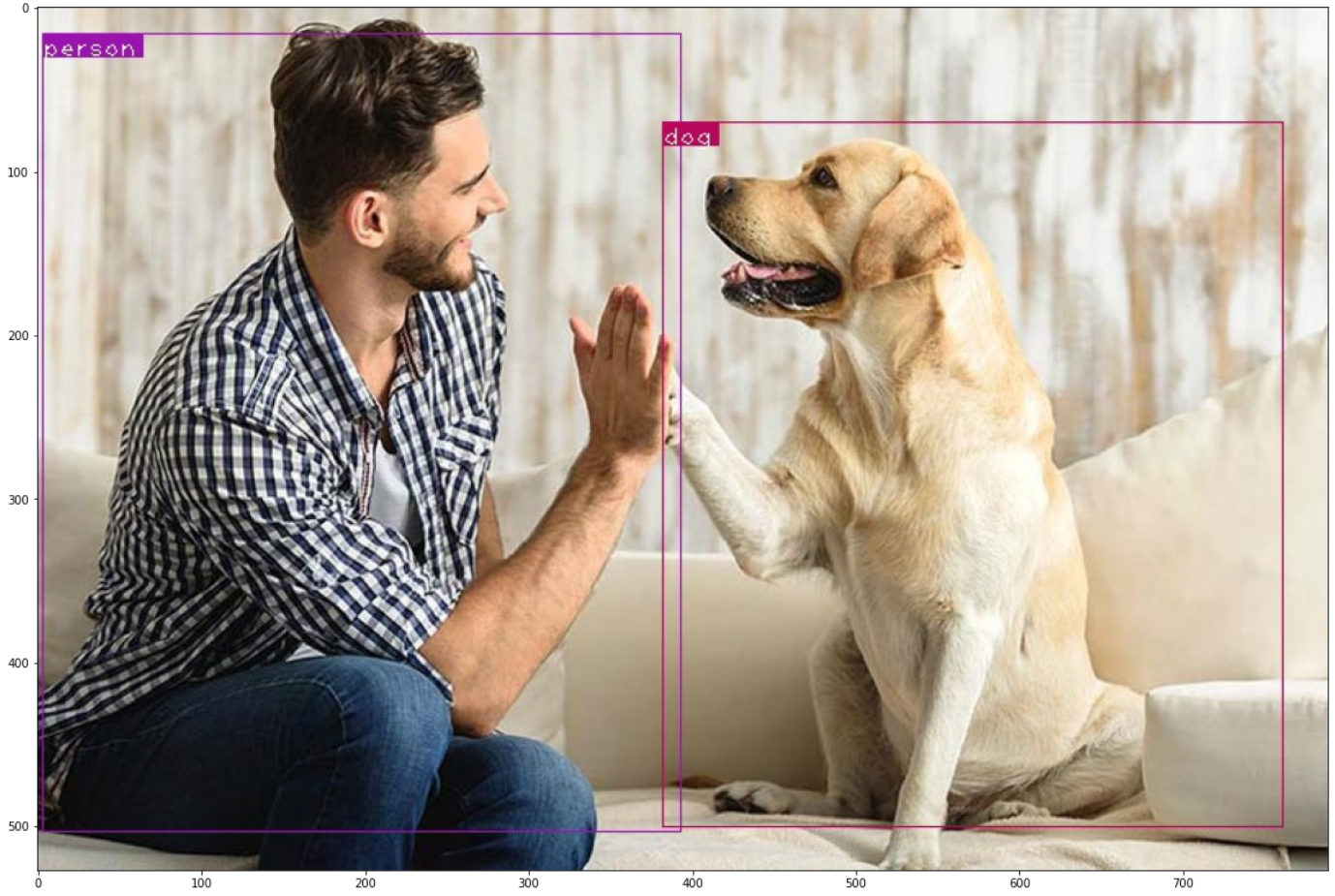
Results of object recognition using YOLOv3:



Model was able to label even multiple cars and persons. It can also distinguish the difference between truck, car, and motorbike! It can even identify the person in the motorbike!

4. Python object recognition - *YOLO version 3 using Pytorch*

Results of object recognition using YOLOv3:



Model was able to label person and dog. Unfortunately, it was not able to identify the flag and tree in the Oble pic.

4. Python object recognition - *YOLO version 3 using Pytorch*

Results of object recognition using YOLOv3:

Model was able to label individual persons and the truck. It was also able to recognize the handbag of that one person!

Code from: https://colab.research.google.com/github/tugstugi/dl-colab-notebooks/blob/master/notebooks/YOLOv3_PyTorch.ipynb#scrollTo=P6ETX8scB7o

CRITERIA	QUALIFICATIONS	SCORE
Technical correctness	☑ Met all objectives. ☑ Results are complete. Results are verifiably correct. ☑ Understood the lesson.	30
Quality of presentation.	☑ All text and images are of good quality. ☑ A picture or diagram of the setup is shown (if the activity has one). ☑ Code has sufficient comments and guides. ☑ All plots are properly labeled and are visually understandable. ☑ The report is clear.	30
Reflection	☑ Explained the validity of results. ☑ Discussed what went right or wrong in the activity. ☑ Justified the self score. ☑ Acknowledged sources (e.g. persons consulted, references, etc.)	30
Ownership	☑ Applied the technique on other data ☑ Discovered limitations in the technique (Pretrained models have better performance when background is plain) ☑ Introduced improvements to the technique (Not much on the architecture since you need to retrain it if you want to improve it, but I did find out that plain background can have better performance. ☑ Did it on YOLO	10

100