Week 1 & 2

OpenCV Installation Command:

> pip install opencv-python

Importing an Image

```
import cv2
from google.colab.patches import cv2_imshow
img = cv2.imread("Lena.png")   cv2_imshow(img)
```

Converting image into RGB

```
import cv2  import
matplotlib.pyplot as plt img =
cv2.imread("Lena.png")
cv2_imshow(img)
```

```
image_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB) cv2_imshow(img)
```

Converting image into GRAY

```
import cv2  import
matplotlib.pyplot as plt img =
cv2.imread("Lena.png")
print("BGR")
cv2_imshow(img)
          print("GRAY")
 image_gray =
cv2.cvtColor(img,
```

```
cv2.COLOR_BGR2GRAY)
cv2_imshow(image_gray)
```

BGR

GRAY

Plot the three channels of the image

```
import numpy as np
img = cv2.imread("Lena.png", cv2.IMREAD_UNCHANGED)
```

Blue Channel

```
blue_channel = img[:,:,0] blue_img
= np.zeros(img.shape)
blue_img[:,:,0] = blue_channel
cv2_imshow(blue_img)
```

Green Channel

```
green_channel = img[:,:,1] green_img
= np.zeros(img.shape)
green_img[:,:,1] = green_channel
cv2_imshow(green_img)
```

Red Channel

```
red_channel = img[:,:,2] red_img
= np.zeros(img.shape)
red_img[:,:,2] = red_channel
cv2_imshow(red_img)
```

Transform the image into HLS

```
img = cv2.imread("Lena.png")   img_hls =
cv2.cvtColor(img, cv2.COLOR_BGR2HLS)
cv2_imshow(img_hls)
```

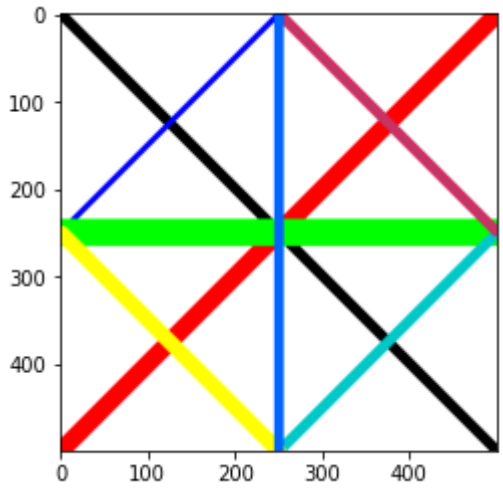
Transform the image into HSV

```
img = cv2.imread("Lena.png")   img_hsv =
cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
cv2_imshow(img_hsv)
```

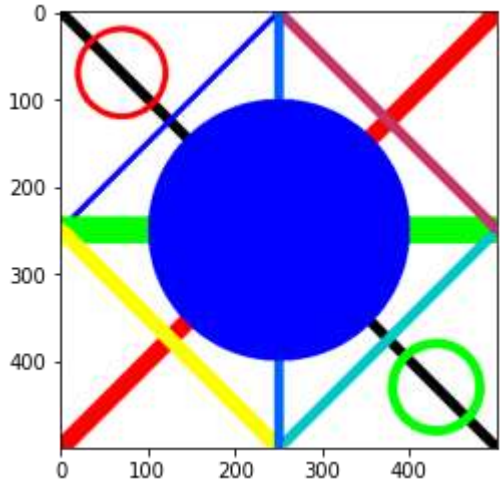
Drawing rectangle, lines, circle on Image

```
img = np.zeros((500,500,3), dtype="uint8")
# Changing the color of the image
img[:] = (255,255,255) # Drawing
a black line
cv2.line(img, (0,0), (500,500), (0,0,0), (10))
# Drawing a blue line
cv2.line(img, (0,500), (500,0), (255,0,0), (20))
# Drawing a red line
cv2.line(img, (0,250), (250,0), (0,0,255), (5))
# Drawing a green line
cv2.line(img, (500,250), (0,250), (0,255,0), (30))
# Drawing a yellow line
cv2.line(img, (500,250), (250,500), (0,200,200), (10))
# Drawing a violet line
cv2.line(img, (250,0), (500,250), (200,50,100), (10))
# Drawing a cyan line
cv2.line(img, (0,250), (250,500), (255,255,0), (15))
# Drawing a orange line
cv2.line(img, (250,0), (250,500), (0,100,255), (10))
```

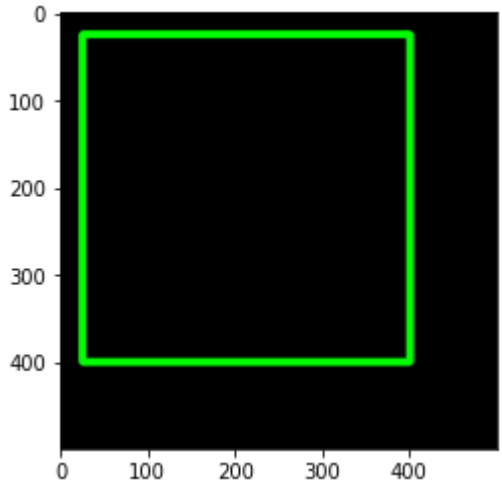
```
plt.imshow(img)
<matplotlib.image.AxesIm
age at 0x7ff65415c710>
```




```
cv2.circle(img, (250,250), 150, (0,0,255), (-1))
# Drawing a blue circle
cv2.circle(img, (70,70), 50, (255,0,0), (5))
# Drawing a green circle
cv2.circle(img, (430,430), 50, (0,255,0), (10))
plt.imshow(img) <matplotlib.image.AxesImage at
0x7ff654124e90>
```




```
img = np.zeros((500,500,3),dtype='uint8')
image = cv2.rectangle(img, (25,25), (400,400), (0,255,0), 7)
plt.imshow(image)

<matplotlib.image.AxesImage at 0x7ff653f138d0>
```



Adding Text on Image

```
image = np.zeros((200,500))
font = cv2.FONT_HERSHEY_SIMPLEX
 org = (50, 50)    fontScale =
1      color = (255, 0, 0)
thickness = 2
image = cv2.putText(image, 'Computer Vision Lab', org, font,
fontScale, (255,0,0), thickness, cv2.LINE_AA) cv2_imshow(image)
```

Week 3 & 4

Storing and Accessing many Images using Python (pickle)

```
from PIL import Image    import
pickle
```

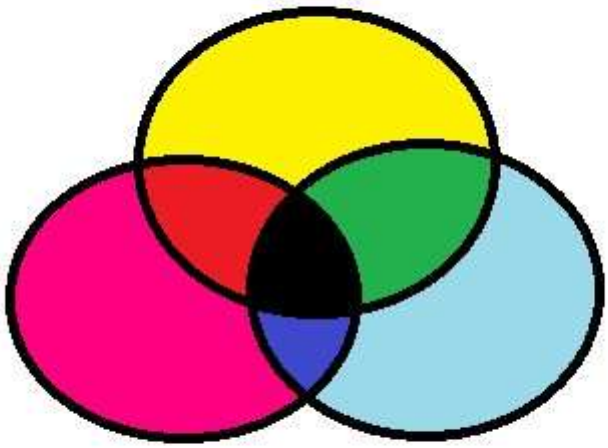
```
img = Image.open('Lena.png')   with
open('savedimage.pkl', 'wb') as f:
pickle.dump(img, f)
```

Week 5 & 6

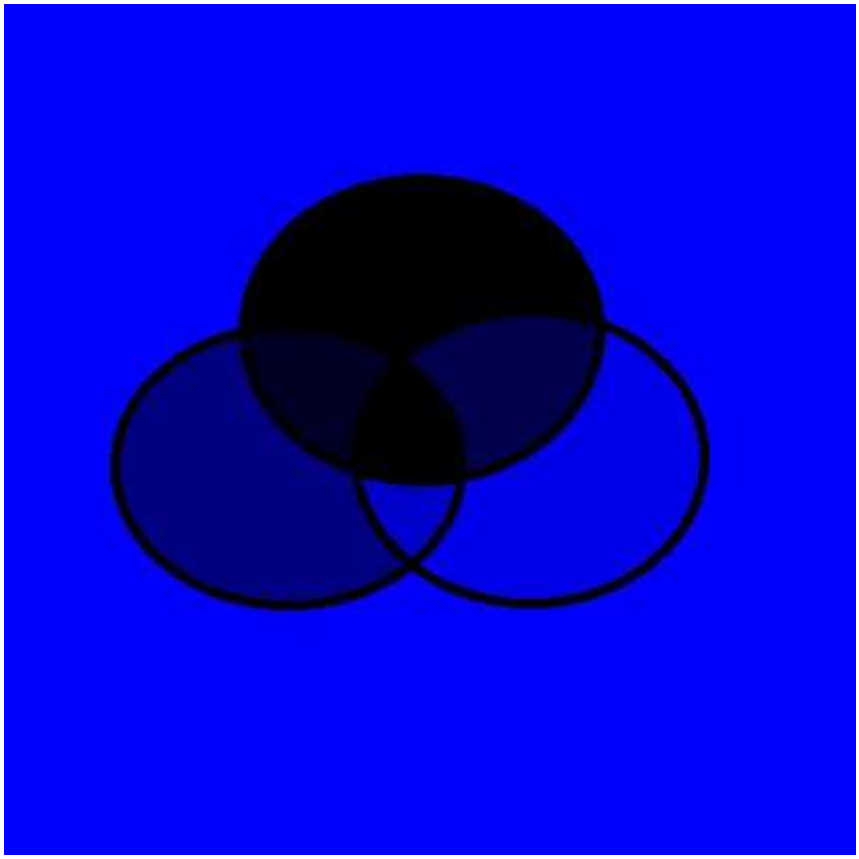
Image Segmentation Using Color Spaces in OpenCV + Python

Color Spaces and Reading Images in OpenCV

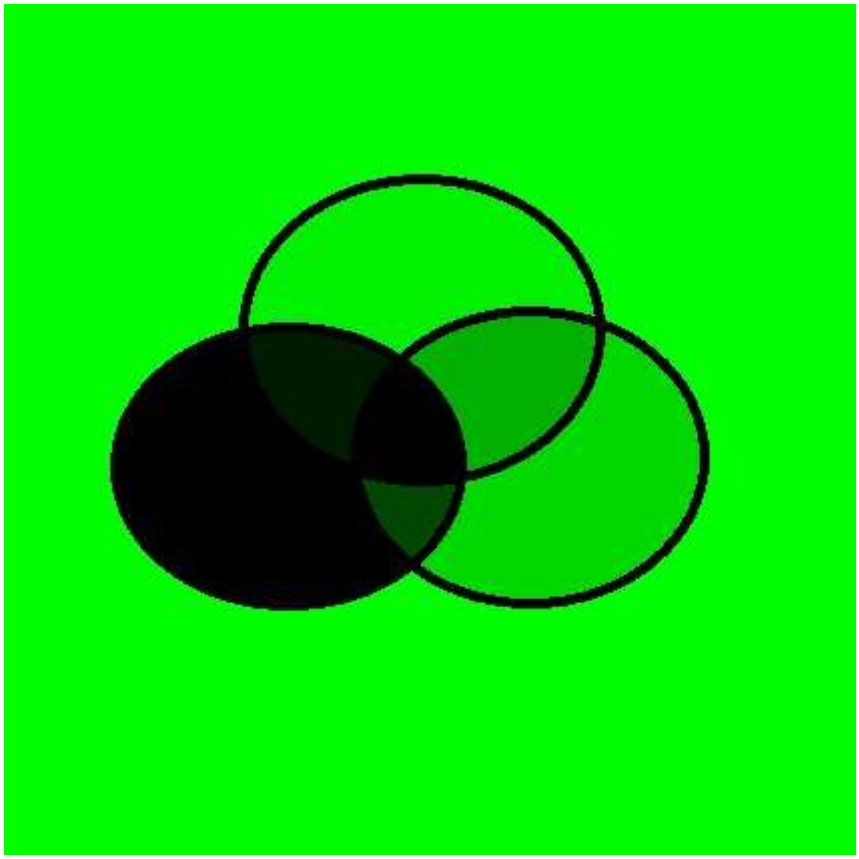
```
img = cv2.imread("cmyk_paint.png", cv2.IMREAD_UNCHANGED)
cv2_imshow(img)
```



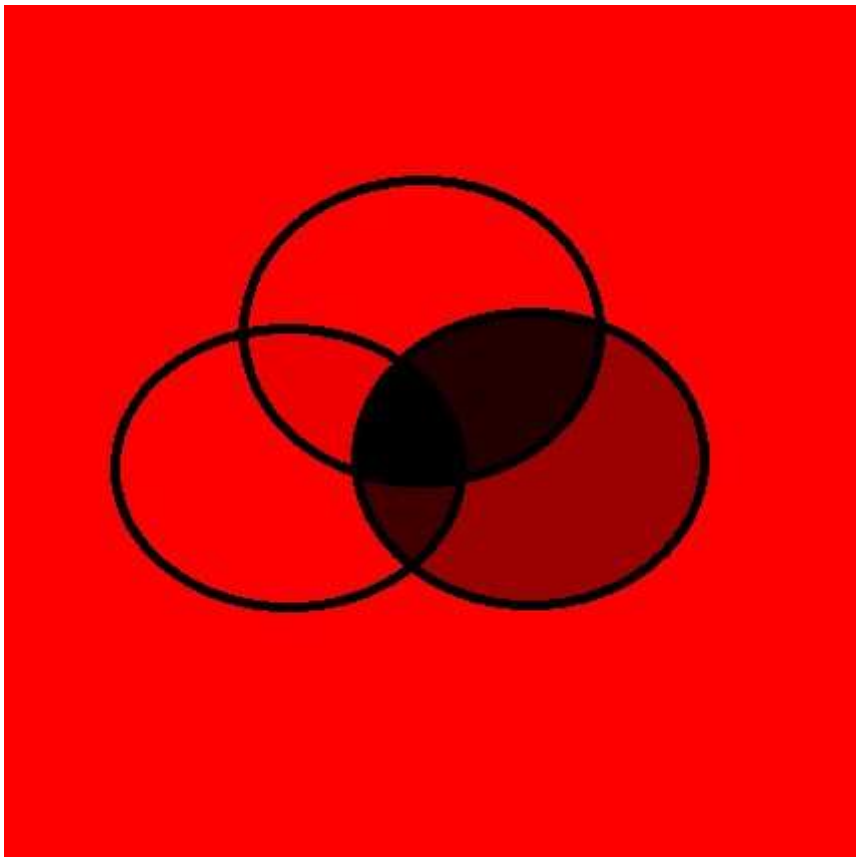
```
img = cv2.imread("cmyk_paint.png", cv2.IMREAD_UNCHANGED)
 blue_channel = img[:,:,0] blue_img = np.zeros(img.shape)
blue_img[:,:,0] = blue_channel print("Blue Color Space")
 cv2_imshow(blue_img) Blue Color Space
```



```
img = cv2.imread("cmyk_paint.png", cv2.IMREAD_UNCHANGED)
 green_channel = img[:,:,1] green_img =
np.zeros(img.shape) green_img[:,:,1] = green_channel
print("Green Color Space")     cv2_imshow(green_img)
Green Color Space
```

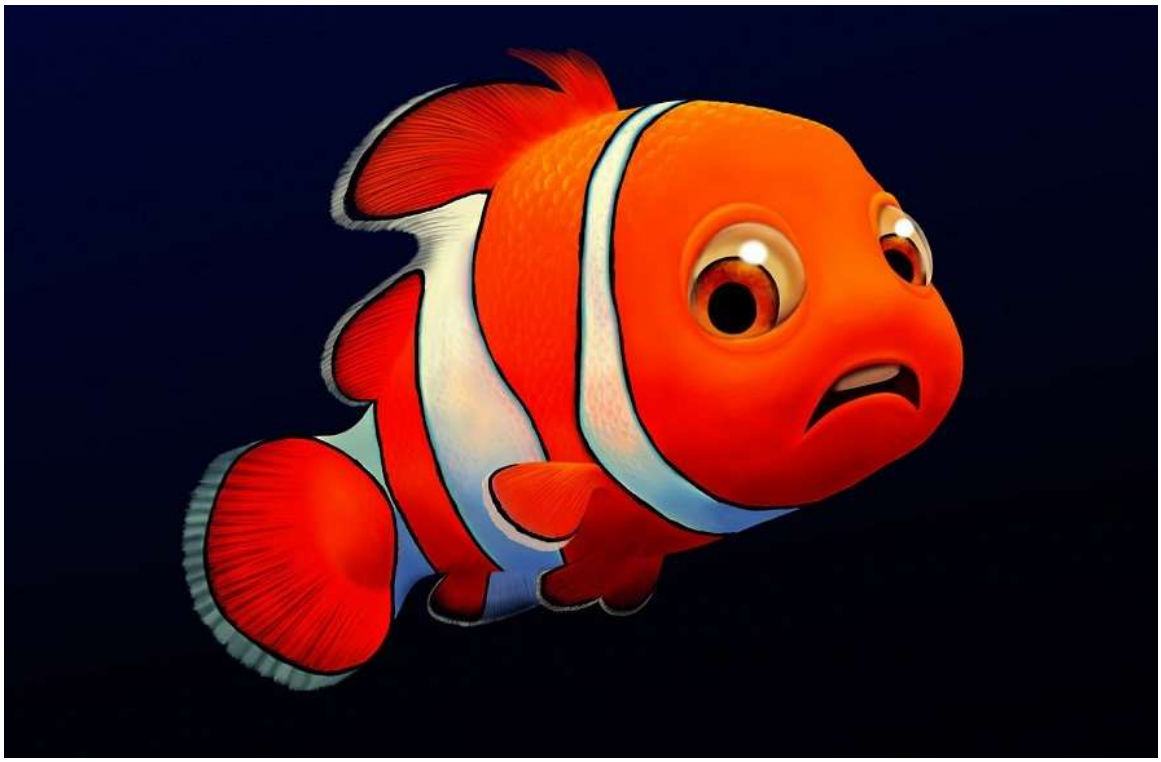


```
img = cv2.imread("cmyk_paint.png", cv2.IMREAD_UNCHANGED)
 red_channel = img[:,:,2] red_img = np.zeros(img.shape)
red_img[:,:,2] = red_channel print("Red Color Space")
cv2_imshow(red_img) Red Color Space
```



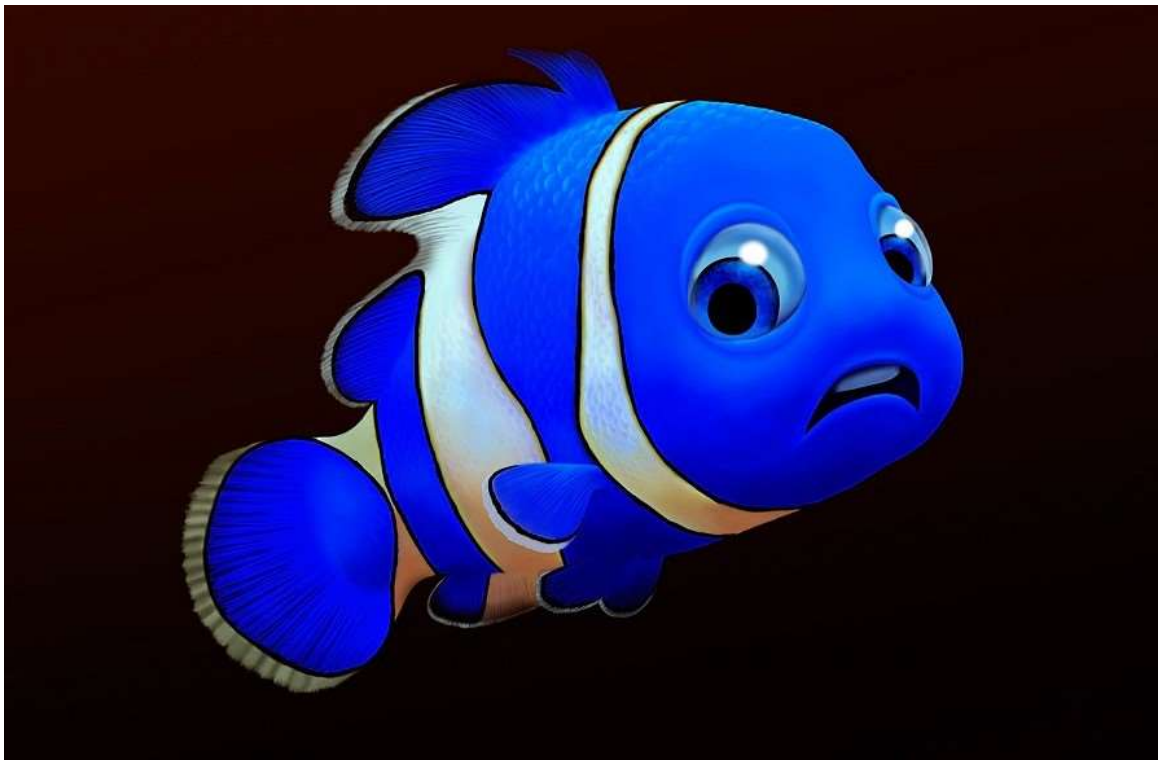
Visualizing Nemo in RGB Color Space

```
nemo_img = cv2.imread("nemo5.jpg") cv2_imshow(nemo_img)
```



```
print("Visualizing Nemo in RGB Color Space")      nemo_rgb
= cv2.cvtColor(nemo_img, cv2.COLOR_BGR2RGB)
cv2_imshow(nemo_rgb)
```

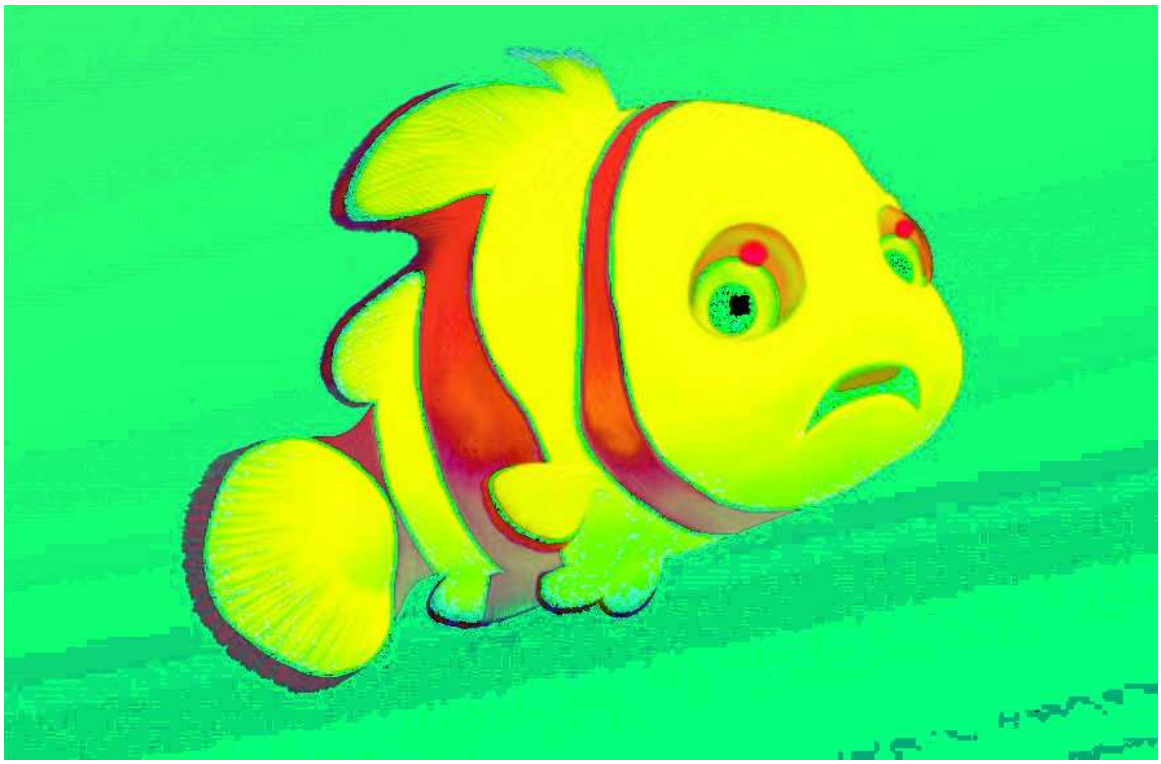
```
Visualizing Nemo in RGB Color Space
```



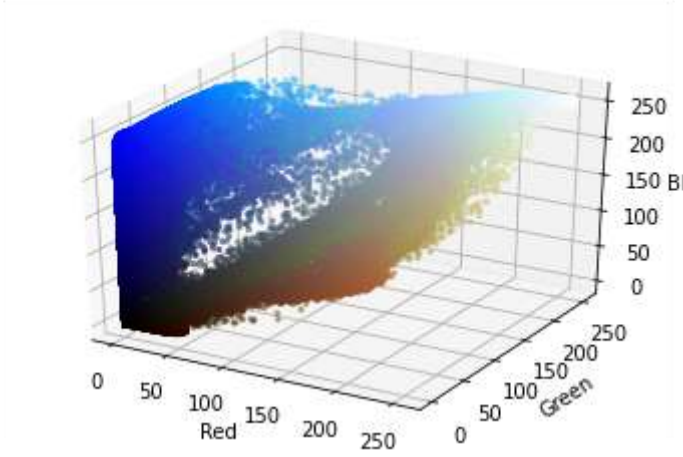
Visualizing Nemo in HSV Color Space

```
print("Visualizing Nemo in HSV Color Space")     nemo_hsv
= cv2.cvtColor(nemo_img, cv2.COLOR_BGR2HSV)
cv2_imshow(nemo_hsv)
```

```
Visualizing Nemo in HSV Color Space
```



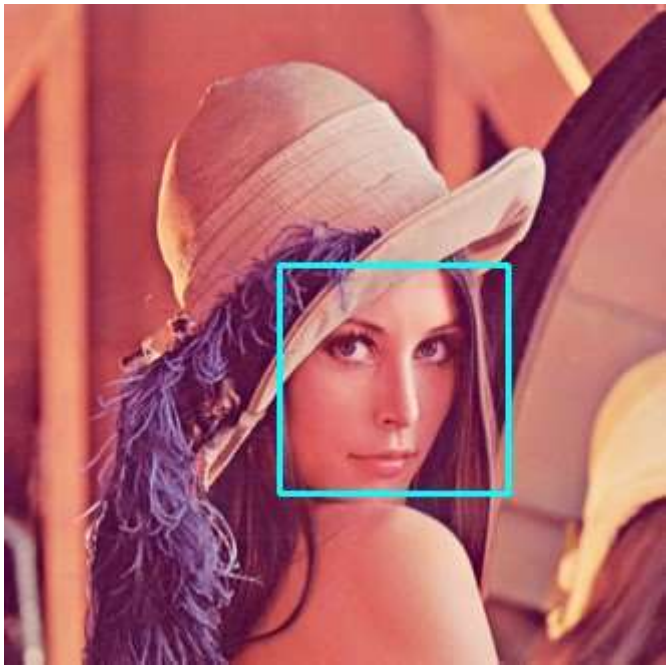
```
from mpl_toolkits.mplot3d import Axes3D
 from matplotlib import cm     from
matplotlib import colors
          r, g, b =
cv2.split(nemo_img) fig =
plt.figure()
axis = fig.add_subplot(1, 1, 1, projection="3d")
pixel_colors = nemo_img.reshape((np.shape(nemo_img)
         [0]*np.shape(nemo_img)[1],
                                    3)) norm =
colors.Normalize(vmin=-1.,vmax=1.)
norm.autoscale(pixel_colors) pixel_colors =
norm(pixel_colors).tolist()
axis.scatter(r.flatten(), g.flatten(), b.flatten(),
facecolors=pixel_colors, marker=".")
axis.set_xlabel("Red") axis.set_ylabel("Green")
axis.set_zlabel("Blue") plt.show()
```



Week 7 & 8

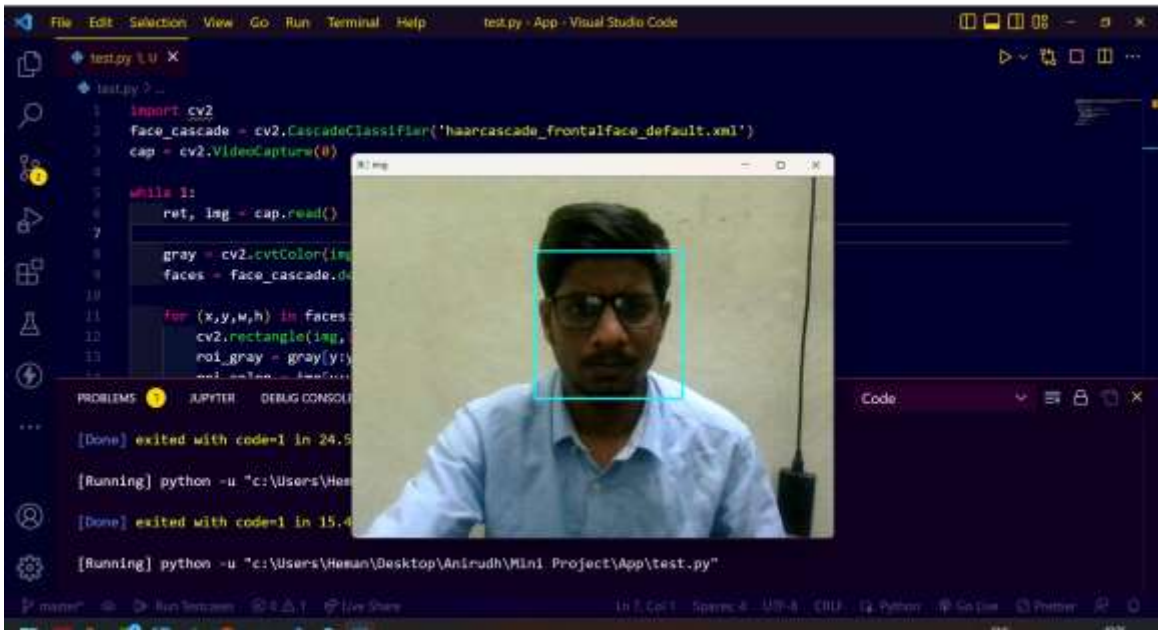
Face Detection in Python, OpenCV

```
face_cascade =
        cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
img = cv2.imread("Lena.png")   gray =
cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)      faces =
face_cascade.detectMultiScale(gray, 1.3, 5)
         for (x,y,w,h)
in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)
roi_gray = gray[y:y+h, x:x+w]     roi_color =
img[y:y+h, x:x+w]

cv2_imshow(img)
```



Face Detection in Python, OpenCV - Using a Webcam

```
import cv2
face_cascade =
        cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)

while 1:
        ret, img = cap.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, 1.3, 5)

        for (x,y,w,h) in faces:
            cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)
      roi_gray = gray[y:y+h, x:x+w]              roi_color =
img[y:y+h, x:x+w]       cv2.imshow('img',img)
            k = cv2.waitKey(30) &
0xff  if k == 27:
                break
cap.release() cv2.destroyAllWindows()
```



Week 9 & 10

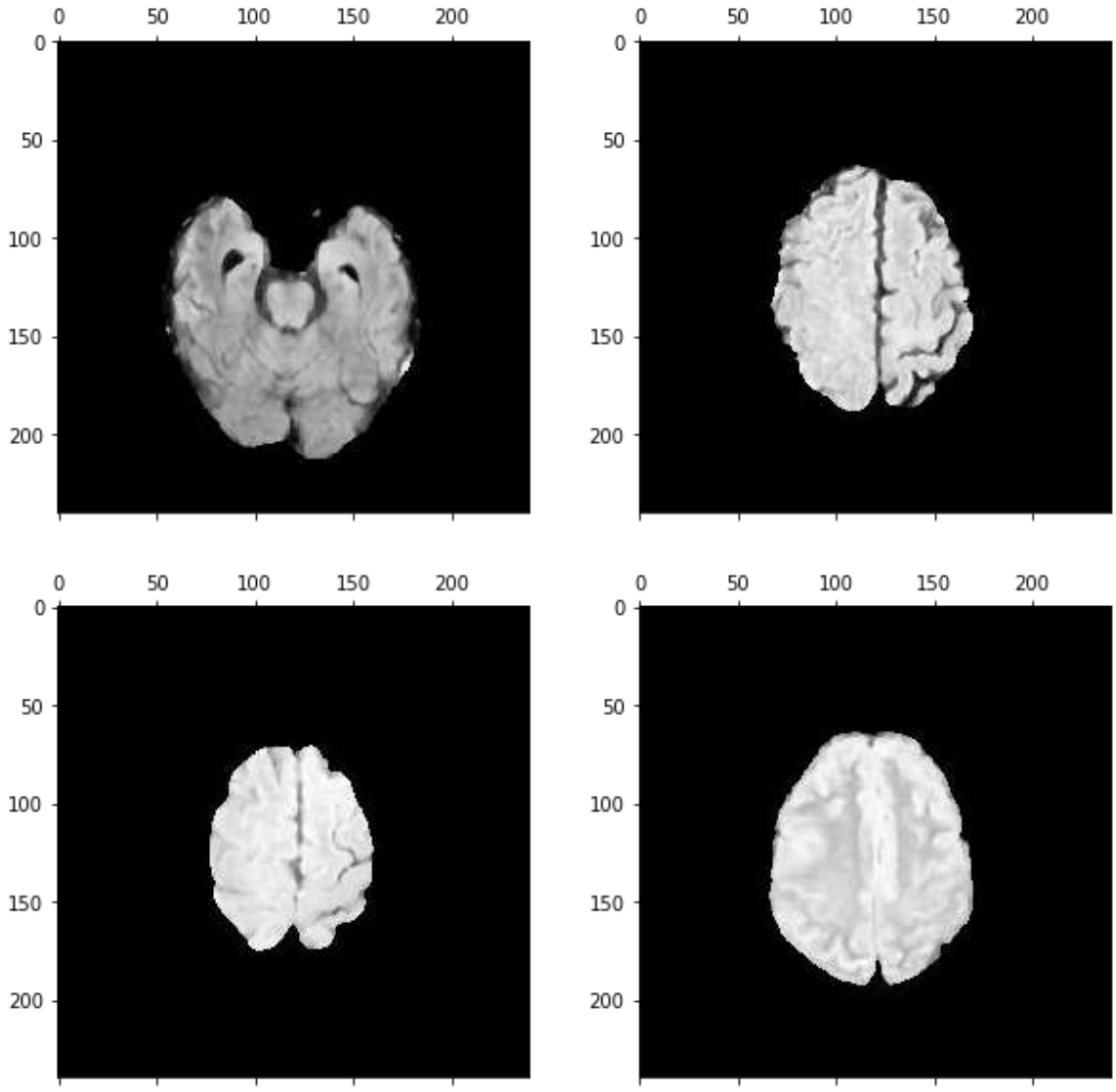
Handling Images with Tensorflow over CoLab

Loading Data Set (Brain Tumor Dataset)

```
import keras
from keras.preprocessing.image import ImageDataGenerator
 import matplotlib.pyplot as plt import numpy as np
import cv2
import tensorflow as tf training_datagen

= ImageDataGenerator()

training_data_path = "/content/drive/MyDrive/Data Sets/Brain Tumor
        Dataset/Data"

training_data = training_datagen.flow_from_directory(
training_data_path,
    target_size=(224, 224), class_mode='binary'
    )

Found 3772 images belonging to 2 classes.

plot_image = plt.figure(figsize=(10,10))
         plot1 =
plot_image.add_subplot(2,2,1) plot2 =
```

```
plot_image.add_subplot(2,2,2) plot3 =
plot_image.add_subplot(2,2,3) plot4 =
plot_image.add_subplot(2,2,4)
          plot1.matshow(plt.imread(training_data.filepaths[20]))
plot2.matshow(plt.imread(training_data.filepaths[128]))
plot3.matshow(plt.imread(training_data.filepaths[287]))
plot4.matshow(plt.imread(training_data.filepaths[189]))
```

```
<matplotlib.image.AxesImage at 0x7fba2c968b90>
```



CNN Model

```
model = keras.models.Sequential([
          #CNN layers
          keras.layers.Conv2D(filters=32,
         kernel_size=3,activation='relu',
                              input_shape=[224, 224, 3]),
keras.layers.MaxPooling2D(pool_size=(2,2)),
keras.layers.Conv2D(filters=64,activation='relu',
```

```
        kernel_size=3),
         keras.layers.MaxPooling2D(pool_size=(2,2)),
         #ANN layers for classification
keras.layers.Flatten(),
         keras.layers.Dense(units=64, activation='relu'),
keras.layers.Dropout(0.1),
keras.layers.Dense(units=128, activation='relu'),
keras.layers.Dropout(0.25),
keras.layers.Dense(units=1, activation='sigmoid')
])

model.compile(optimizer = tf.optimizers.Adam(0.0001),
loss='binary_crossentropy', metrics=['accuracy']) model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 222, 222, 32)      896
 max_pooling2d (MaxPooling2D  (None, 111, 111, 32)     0          )
 conv2d_1 (Conv2D)           (None, 109, 109, 64)      18496
 max_pooling2d_1 (MaxPooling  (None, 54, 54, 64)       0
 2D)
 flatten (Flatten)           (None, 186624)            0
 dense (Dense)               (None, 64)                11944000
 dropout (Dropout)           (None, 64)                0
 dense_1 (Dense)             (None, 128)               8320
 dropout_1 (Dropout)         (None, 128)               0
 dense_2 (Dense)             (None, 1)                 129
=================================================================
Total params: 11,971,841
Trainable params: 11,971,841
Non-trainable params: 0
_________________________________________________________________

history = model.fit(training_data, epochs=5)

Epoch 1/5
118/118 [==============================] - 585s 5s/step - loss: 3.2654
- accuracy: 0.6707
Epoch 2/5
118/118 [==============================] - 249s 2s/step - loss: 0.5554
- accuracy: 0.7192
Epoch 3/5
```

```
118/118 [==============================] - 249s 2s/step - loss: 0.4930
- accuracy: 0.7471
Epoch 4/5
118/118 [==============================] - 249s 2s/step - loss: 0.4380
- accuracy: 0.7633
Epoch 5/5
118/118 [==============================] - 246s 2s/step - loss: 0.3918
- accuracy: 0.7892 training_data
```

ResNet50

```
from tensorflow.keras.applications.resnet50 import ResNet50  from
tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input,
decode_predictions
from tensorflow.keras.optimizers import Adam

resnet50 = ResNet50(
include_top=True,
weights="imagenet",
input_tensor=None,
input_shape=None,
pooling=None,     classes=1000,)

Downloading data from
https://storage.googleapis.com/tensorflow/kerasapplications/resnet/resn
et50_weights_tf_dim_ordering_tf_kernels.h5 102967424/102967424
[==============================] - 1s 0us/step for layer in
resnet50.layers:   layer.trainable = False

resnet_model = keras.Sequential() resnet_model.add(resnet50)

resnet_model.add(keras.layers.Dense(units=64,activation="relu"))
resnet_model.add(keras.layers.Dense(units=128,activation="relu"))
resnet_model.add(keras.layers.Dense(units=1,activation="sigmoid"))
resnet_model.summary()

Model: "sequential_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Functional)        (None, 1000)              25636712
dense_3 (Dense)              (None, 64)                64064
```

```
dense_4 (Dense)              (None, 128)               8320
dense_5 (Dense)              (None, 1)                 129
=================================================================
Total params: 25,709,225
Trainable params: 72,513
Non-trainable params: 25,636,712
_________________________________________________________________
```

```
resnet_model.compile(optimizer = Adam(learning_rate = 0.0001),
loss='binary_crossentropy', metrics=['accuracy'])
resnet_model.fit(training_data, epochs=1)
```

```
118/118 [==============================] - 557s 5s/step - loss: 0.6872
- accuracy: 0.6686 <keras.callbacks.History at
  0x7fba2285e250> resnet_model.fit(training_data,
  epochs=1)
118/118 [==============================] - 511s 4s/step - loss: 0.6631
- accuracy: 0.6461
<keras.callbacks.History at 0x7f36b6dee250>
```

Mobile Net

```
from tensorflow.keras.applications.mobilenet import MobileNet

mobilenet = MobileNet(
input_shape=None,
include_top=True,
weights="imagenet",
input_tensor=None,
pooling=None,     classes=1000,
)

for layer in mobilenet.layers:   layer.trainalbe =
False

mobilenet_model = keras.Sequential() mobilenet_model.add(mobilenet)

mobilenet_model.add(keras.layers.Dense(units=64,activation="relu"))
mobilenet_model.add(keras.layers.Dense(units=128,activation="relu"))
mobilenet_model.add(keras.layers.Dense(units=1,activation="sigmoid"))
mobilenet_model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
mobilenet_1.00_224 (Functio  (None, 1000)             4253864
nal)
dense_9 (Dense)             (None, 64)                64064
dense_10 (Dense)            (None, 128)               8320
dense_11 (Dense)            (None, 1)                 129
=================================================================
Total params: 4,326,377
Trainable params: 4,304,489
Non-trainable params: 21,888
_________________________________________________________________
mobilenet_model.compile(optimizer = Adam(learning_rate = 0.0001),
loss='binary_crossentropy', metrics= ['accuracy'])
mobilenet_model.fit(training_data, epochs=1)

118/118 [==============================] - 627s 5s/step - loss: 0.6453
- accuracy: 0.8598 <keras.callbacks.History

at 0x7f36b4cabbd0>

mobilenet_model.fit(training_data, epochs=1)
```

VGG16

```
from tensorflow.keras.applications.vgg16 import VGG16

vgg16 = VGG16(
input_shape=None,
include_top=True,
weights="imagenet",
input_tensor=None,
pooling=None,     classes=1000,
)

Downloading data from
https://storage.googleapis.com/tensorflow/kerasapplications/vgg16/vgg16
_weights_tf_dim_ordering_tf_kernels.h5
553467904/553467096 [==============================] - 5s 0us/step
553476096/553467096 [==============================] - 5s 0us/step
```

```
for layer in vgg16.layers:   layer.trainalbe =
False

vgg16_model = keras.Sequential() vgg16_model.add(vgg16)

vgg16_model.add(keras.layers.Dense(units=64,activation="relu"))

vgg16_model.add(keras.layers.Dense(units=128,activation="relu"))

vgg16_model.add(keras.layers.Dense(units=1,activation="sigmoid"))

vgg16_model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
================================================================= vgg16
(Functional)          (None, 1000)              138357544
dense_12 (Dense)             (None, 64)                64064
dense_13 (Dense)             (None, 128)               8320
dense_14 (Dense)             (None, 1)                 129
=================================================================
Total params: 138,430,057
Trainable params: 138,430,057
Non-trainable params: 0
_________________________________________________________________
```

```
vgg16_model.compile(optimizer = Adam(learning_rate = 0.0001),

loss='binary_crossentropy', metrics=['accuracy'])

vgg16_model.fit(training_data, epochs=1)
```

```
102/118 [=======================>.....] - ETA: 14:05 - loss: 0.6897 -
accuracy: 0.5494
```

Converting normal Image to cartoon image using OpenCV & Python

```
import cv2 import numpy as np
import matplotlib.pyplot as plt
          img = cv2.imread("Lena.png")
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
plt.figure(figsize=(6,6)) plt.imshow(img)
plt.axis("off") plt.title("Original
Image") plt.show()
```

Original Image

```
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
 gray = cv2.medianBlur(gray, 5)
plt.figure(figsize=(6,6))
plt.imshow(gray,cmap="gray") plt.axis("off")
plt.title("Grayscale Image") plt.show()
```

Grayscale Image

```
edges = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
        cv2.THRESH_BINARY, 9, 9)
plt.figure(figsize=(6,6)) plt.imshow(edges,cmap="gray")
plt.axis("off") plt.title("Edged Image") plt.show()
```

Edged Image

```
color = cv2.bilateralFilter(img, 9, 250, 250)
cartoon = cv2.bitwise_and(color, color, mask=edges)
plt.figure(figsize=(6,16))
plt.imshow(cartoon,cmap="gray") plt.axis("off")
plt.title("Cartoon Image") plt.show()
```

Cartoon Image

Line detection in python with OpenCV | Hough line method

```
import cv2 import numpy as np img =
cv2.imread("window.jpg")       gray =
```

```
cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 75, 150)
lines = cv2.HoughLinesP(edges, 1, np.pi/180, 30, maxLineGap=250) for
line in lines:
   x1, y1, x2, y2 = line[0]
   cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 1)
cv2_imshow(edges) cv2_imshow(img)
```