

# UNIVERSITY OF TECHNOLOGY & EDUCATION

FACULTY FOR HIGH-QUALITY TRAINING



## Artificial Intelligent

### RECOGNIZE 10 ANIMAL IN REAL-TIME

TEACHER: PGS. TS Nguyễn Trường Thịnh

NAME: Đinh Thanh Cường

MSSV: 19146110

Năm học: Học kì II/ 2021 – 2022

## CONTENT

CHAPTER 1: ABOUT Artificial Intelligent	3
1. About Artificial Intelligent (AI)	
2. How does AI work?	4
3. Why is artificial intelligence important?	
CHAPTER 2:INTRODUCING ABOUT CONVOLUTIONAL NEURAL NETWORK	5
2.1 Introducing about convolutional neural network	5
2.2 The structure of the CNN network:	6
CHAPTER 3: RESULT	15
3.1 Accuracy:	15
3.2. TESTING	16
3.3 General accuracy results:	20
3.4 CODE:	21
Chapter 4: Conclusion and development direction	28
4.1 Conclusion:	28
4.2 Development direction:	30
REFERENCES	32

## CHAPTER 1: ABOUT Artificial Intelligent (AI):

### 1.1 ABOUT Artificial Intelligent (AI):

Artificial Intelligent is: Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. Specific applications of AI include [expert systems](#), natural language processing, speech recognition and [machine vision](#).

### 1.2: How does AI work?

As the hype around AI has accelerated, vendors have been scrambling to promote how their products and services use AI. Often what they refer to as AI is simply one component of AI, such as machine learning. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. No one programming language is synonymous with AI, but a few, including Python, R and Java, are popular.

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chatbot that is fed examples of text chats can learn to produce lifelike exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples.

AI programming focuses on three cognitive skills: learning, reasoning and self-correction.

**Learning processes.** This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

**Reasoning processes.** This aspect of AI programming focuses on choosing the right algorithm to reach a desired outcome.

**Self-correction processes.** This aspect of AI programming is designed to continually fine-tune algorithms and ensure they provide the most accurate results possible.

## **1.3 Why is artificial intelligence important?**

AI is important because it can give enterprises insights into their operations that they may not have been aware of previously and because, in some cases, AI can perform tasks better than humans. Particularly when it comes to repetitive, detail-oriented tasks like analyzing large numbers of legal documents to ensure relevant fields are filled in properly, AI tools often complete jobs quickly and with relatively few errors.

This has helped fuel an explosion in efficiency and opened the door to entirely new business opportunities for some larger enterprises. Prior to the current wave of AI, it would have been hard to imagine using computer software to connect riders to taxis, but today Uber has become one of the largest companies in the world by doing just that. It utilizes sophisticated machine learning algorithms to predict when people are likely to need rides in certain areas, which helps proactively get drivers on the road before they're needed. As another example, Google has become one of the

largest players for a range of online services by using machine learning to understand how people use their services and then improving them. In 2017, the company's CEO, Sundar Pichai, pronounced that Google would operate as an "AI first" company.

Today's largest and most successful enterprises have used AI to improve their operations and gain advantage on their competitors.



Figure1: AI

## **CHAPTER 2: INTRODUCING ABOUT CONVOLUTIONAL NEURAL NETWORK**

### **2.1 : INTRODUCING ABOUT CONVOLUTIONAL NEURAL NETWORK:**

The field of machine learning has taken a significant turn in recent times, with the emergence of Artificial Neural Networks (ANNs). These bio-inspired computational models can outperform the performance of previous forms of artificial intelligence in conventional machine learning tasks. One of the most impressive forms of ANN architecture is that of a Constituent Neural Network (CNN). CNNs are mainly used to solve difficult image-oriented pattern recognition tasks, and with their precise yet simple architecture, provide a simple method to get started with ANNs.

This document provides a brief introduction to CNN, discusses recently published papers, and emerging techniques in the development of these amazingly amazing image recognition models. This introduction assumes that you are familiar with the fundamentals of ANN and machine learning.

## 2.2: The structure of the CNN network:

A CNN network is a collection of Convolution layers that overlap and use nonlinear activation of functions such as ReLU and tanh to activate significant numbers in nodes. Each layer after passing the functional operation will create more information utilities for the next layers.

CNN have link together by convolution. Next new layer is result of old layer, so we get local connections. Thus, each neuron in the next layer arises from the result of a filter applied to a local image region of the previous neuron.

Each class that uses different filters usually takes hundreds of thousands of such filters and combines their results. In addition, there are some other layers such as pooling/subsampling layer used to filter out more useful information (remove noise information).

During the training of the network, the CNN automatically learns the values through the filter classes based on how you do it. For example, in the image classification task, CNNs will try to find the optimal parameters for the corresponding filters in the order raw pixels > edges > shapes > facial > high-level features. The last layer is used to classify the image.

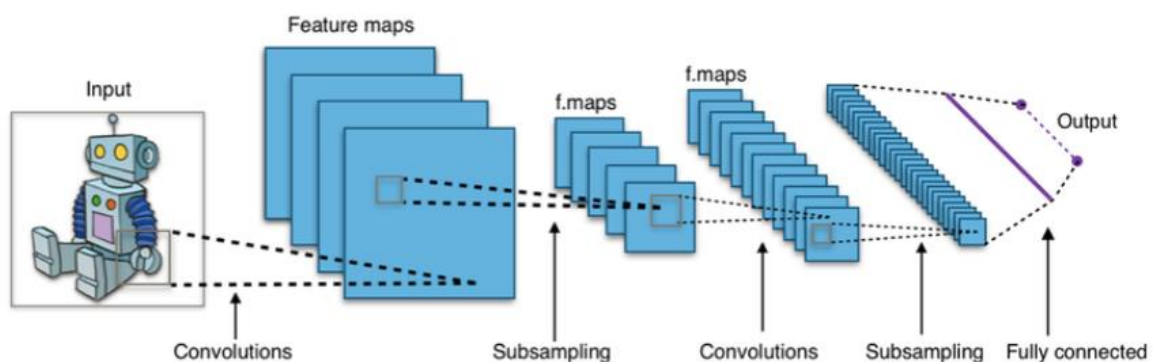


Figure2 : 5 layer CNN

In the CNN model, there are two aspects that need attention: invariance (Location Invariance) and coherence (Compositionality). With the same object, if this object is projected in different degrees (translation, rotation, scaling), the accuracy of the algorithm will be significantly affected.

Pooling layer will give you invariant to translation, rotation and scaling. Local associativity gives us lower-to-higher and more abstract levels of information representation through convolution from filters.

That is why CNNs produce models with very high accuracy. Just like how humans perceive objects in nature.

The CNN network uses three basic ideas:

- Local receptive fields

- Shared weights

- Pooling.

- How to choose parameters for CNN:

- +Number of convolution layers: the more convolution layers, the better the performance.

- +After about 3 or 4 layers, the effects are significantly reduced

- +Filter size: usually filter by size  $5 \times 5$  or  $3 \times 3$

- +Pooling size: usually  $2 \times 2$  or  $4 \times 4$  for large input images

+The last way is to perform multiple training tests to choose the best param.

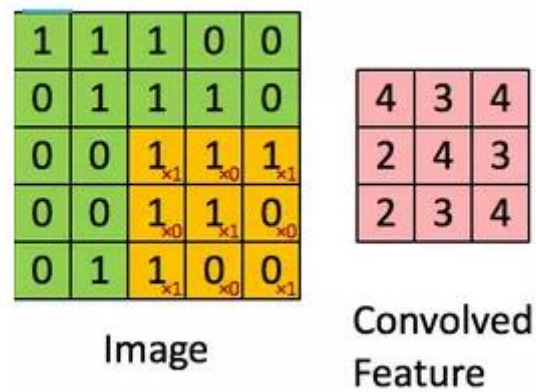
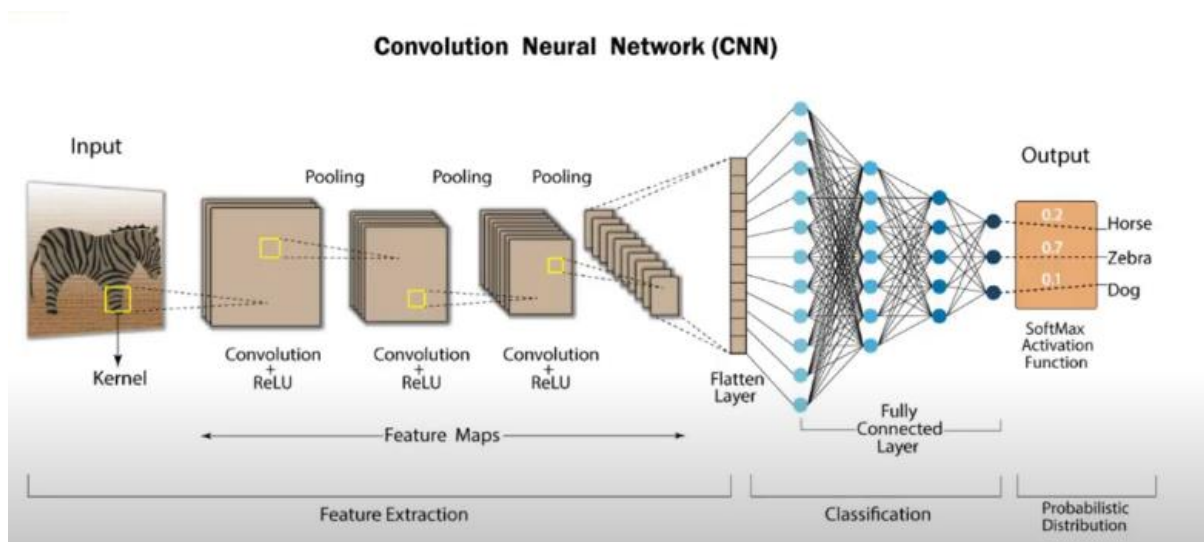


figure 3: matrix 3x3



## Chapter 2: LITERATURE REVIEW

### 2.1 Create DATASETS:







Figure 2.2.1 Create animals datasets.

## 2.2 CHOOSING THE APPROPRIATE LIBRARY AND PATH TO THE TRAINING/TESTING DATASET:

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import pandas as pd

import tensorflow as tf

from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
from tensorflow.keras.optimizers import Adam
from keras.utils.np_utils import to_categorical
from keras.layers.convolutional import Conv2D, MaxPooling2D
from sklearn.model_selection import train_test_split

import os
import random
from keras.preprocessing.image import ImageDataGenerator

physical_devices = tf.config.list_physical_devices("GPU")
tf.config.experimental.set_memory_growth(tf.config.list_physical_devices("GPU")[0], True)
```

*Figure 2.2.2 Necessary libraries and tools.*

```
##### Parameters #####
path = "/content/drive/MyDrive/mydata/images" # folder chứa 10 folder của các classes
testRatio = 0.1 # Chia 10% số ảnh cho testing
validationRatio = 0.1 # Còn lại 90% cho train => Chia 20% cho validation
# Import Ảnh
count = 0
images = []
classNo = []
myList = os.listdir(path)
print("Tổng số Classes: ", len(myList))
noOfClasses = len(myList)
print("Importing Classes.....")
for x in range(0, len(myList)):
    myPicList = os.listdir(path + "/" + str(count))
    for y in myPicList:
        curImg = cv2.imread(path + "/" + str(count) + "/" + y)
        curImg = cv2.resize(curImg, (224,224))
        images.append(curImg)
        classNo.append(count)
    print(count, end=" ")
    count += 1
print(" ")
images = np.array(images)
classNo = np.array(classNo)
```

## Import cv2

OpenCV (Open Source Computer Vision Library) is a free software library for computer vision and machine learning. OpenCV was created to offer a standard infrastructure for computer vision applications and to let commercial goods incorporate machine perception more quickly. Because OpenCV is a BSD-licensed software, it is simple for companies to use and alter the code.

## Import os

In Python, the OS module has methods for dealing with the operating system. Python's basic utility modules include OS. This module allows you to use operating system-dependent functions on the go. Many functions to interface with the file system are included in the `*os*` and `*os.path*` modules.

## Import numpy as np

NumPy is a Python-based open source project that aims to make numerical computation easier. It was established in 2005, based on the Numeric and

Numarray libraries' earlier work. NumPy will always be 100 percent open source software that is free to use and distributed under the modified BSD license.

## Import matplotlib.pyplot as plt

matplotlib.pyplot is a state-based matplotlib interface. It has an implicit graphing system that is similar to MATLAB. It also works as the figure GUI manager and opens figures on your screen.

```
from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
model5 = load_model(model_file_path)
classes = ['bird', 'cat', 'cow', 'dog', 'elephant', 'giraffe', 'horse', 'lion', 'squirrel', 'lion']
```

*Figure 2.2.3 Give access to Google Drive*

Give Google Colab access to the "Animals" folder on Google Drive.

## 2.3. DATA PROCESSING AND DATA AUGMENTATION

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
X_TRAIN = ImageDataGenerator(rescale=1/255)
X_TEST = ImageDataGenerator(rescale=1/255)
```

*Figure 2.3.1 Data processing and data augmentation*

Because the data is in the RGB color spectrum, it will be divided by 255 to scale down the values. It was rescaled to be in the 0 to 1 range. Data augmentation will add 6 more photographs to every picture taken throughout the training phase, enhancing the data with a random pick from a set of categories

```
TRAIN = X_TRAIN.flow_from_directory(directory=r"/content/drive/MyDrive/Animals",target_size=(150, 150),batch_size=10,class_mode="categorical",color_mode="rgb",)
TEST = X_TEST.flow_from_directory(directory=r"/content/drive/MyDrive/Animals",target_size=(150, 150),batch_size=10,class_mode="categorical",color_mode="rgb",)
```



```
Found 999 images belonging to 10 classes.  
Found 999 images belonging to 10 classes.
```

*Figure 2.3.2 Data input*

With a data input size of 150x150 pixels, the model will require 10 samples for training and 10 samples for testing each time it accepts the data. The color is represented using the RGB color mode. There are 999 train photos available, with 999 test photos.

## 2.4. CREATING NEURAL NETWORK

```
TRAIN.class_indices  
  
{ 'Camap': 0,  
  'Chaucau': 1,  
  'Daibang': 2,  
  'Ga': 3,  
  'Ho': 4,  
  'Huucaoco': 5,  
  'Nguavan': 6,  
  'Rua': 7,  
  'Tegiac': 8,  
  'Voi': 9}
```

*Figure 2.4.1 Show classes*

Show the names of all the classes, there are a total of 10.

```
from sklearn.model_selection import train_test_split  
from keras.models import Sequential  
from keras.layers import Dense, Activation, Dropout, Conv2D, MaxPooling2D, Flatten  
  
model=Sequential()  
model.add(Conv2D(13,(3,3), activation='relu',kernel_initializer='he_uniform',padding='same',input_shape=(150,150,3)))  
model.add(Conv2D(4,(3,3), activation='relu',kernel_initializer='he_uniform',padding='same'))  
model.add(MaxPooling2D((2,2)))  
model.add(Flatten())  
model.add(Dense(1000,activation='relu',kernel_initializer='he_uniform'))  
model.add(Dense(5000,activation='relu',kernel_initializer='he_uniform'))  
model.add(Dense(2500,activation='relu',kernel_initializer='he_uniform'))  
model.add(Dense(1000,activation='relu',kernel_initializer='he_uniform'))  
model.add(Dense(100,activation='relu',kernel_initializer='he_uniform'))  
model.add(Dense(10,activation='softmax'))  
model.summary()
```

Model: "sequential"		
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 150, 150, 13)	364
conv2d_1 (Conv2D)	(None, 150, 150, 4)	472
max_pooling2d (MaxPooling2D)	(None, 75, 75, 4)	0
flatten (Flatten)	(None, 22500)	0
dense (Dense)	(None, 1000)	22501000
dense_1 (Dense)	(None, 5000)	5005000
dense_2 (Dense)	(None, 2500)	12502500
dense_3 (Dense)	(None, 1000)	2501000
dense_4 (Dense)	(None, 100)	100100
dense_5 (Dense)	(None, 10)	1010
Total params: 42,611,446		
Trainable params: 42,611,446		
Non-trainable params: 0		

*Figure 2.4.2 Neural network*

## 2.5. TRAINING

```
from tensorflow.keras.optimizers import SGD
model.compile(optimizer='adam',loss='categorical_crossentropy',metrics=['accuracy'])
history=model.fit(TRAIN,epochs=50,batch_size=20,verbose=1,validation_data=TEST)
```

*Figure 2.5 Training*

This training was improved using "ADAM," which allowed us to train 50 times, collect 20 samples at a time, and demonstrate the process.

## CHAPTER 3: RESULT

### 3.1 Accuracy:

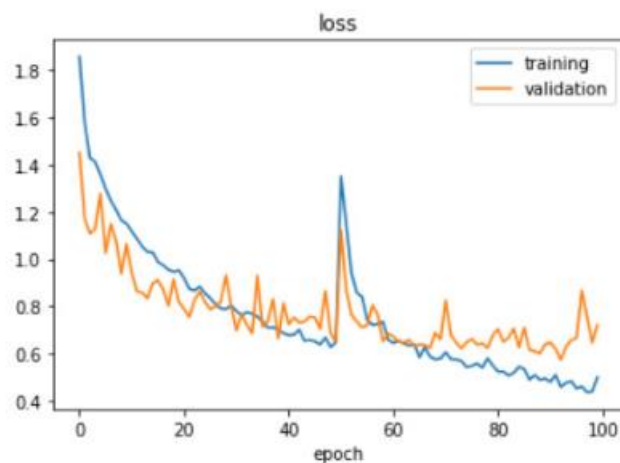
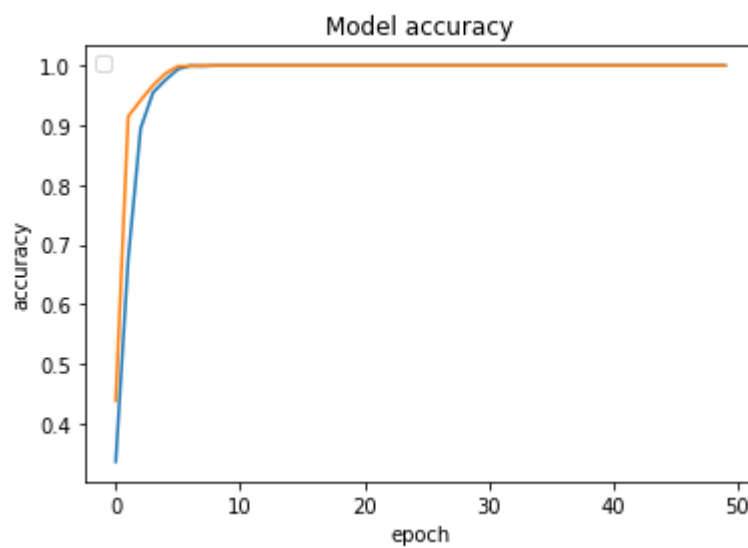
```
score=model.evaluate(TEST,verbose=1)
print('Test loss=',score[0])
print('Test accuracy=',score[1])

100/100 [=====] - 22s 218ms/step - loss: 3.4247e-08 - accuracy: 1.0000
Test loss= 3.4247250368935056e-08
Test accuracy= 1.0
```

*Figure 3.1.1 Accuracy*

The accuracy of the model was percent.

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend('train', 'validation', loc='upper left')
plt.show()
```



*Figure 3.1.2 Accuracy graph*



## 3.2. TESTING



Status:Đang lấy hình ảnh...



Bấm vào video để dừng



```
# update bbox so next frame gets new overlay  
bbox = bbox_bytes
```

...

Status:Đang lấy hình ảnh...



Bấm vào video để dừng

Đang thực thi (31 phút 55 giây) Cell > video\_frame() > eval\_js() > read\_reply\_from\_input()

+ Mã + Văn bản



Status: Đang lấy hình ảnh...



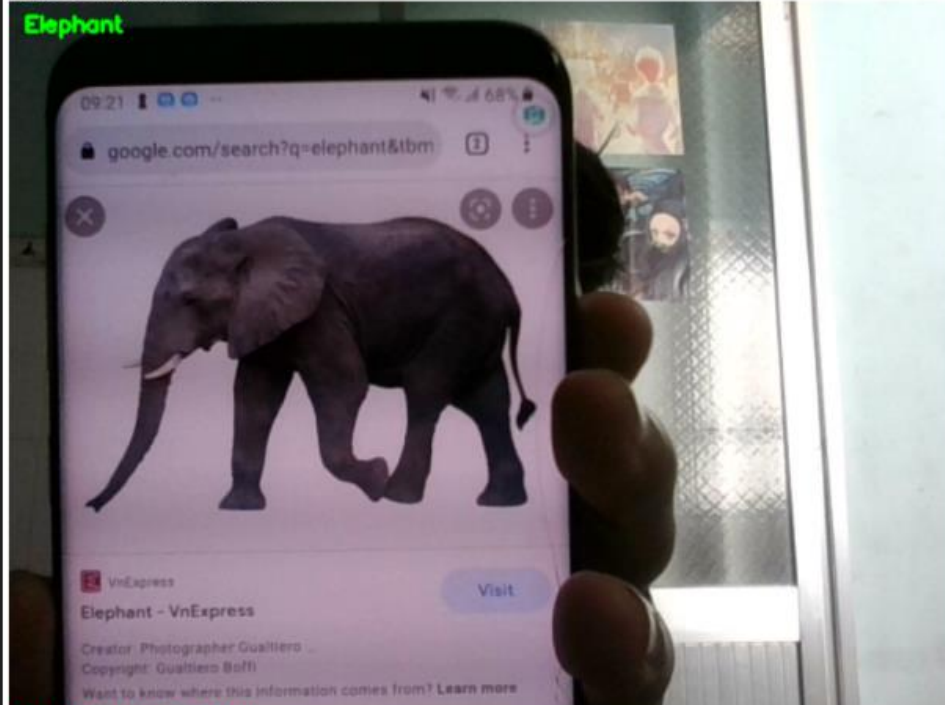
Status:Đang lấy hình ảnh...



Bấm vào video để dừng

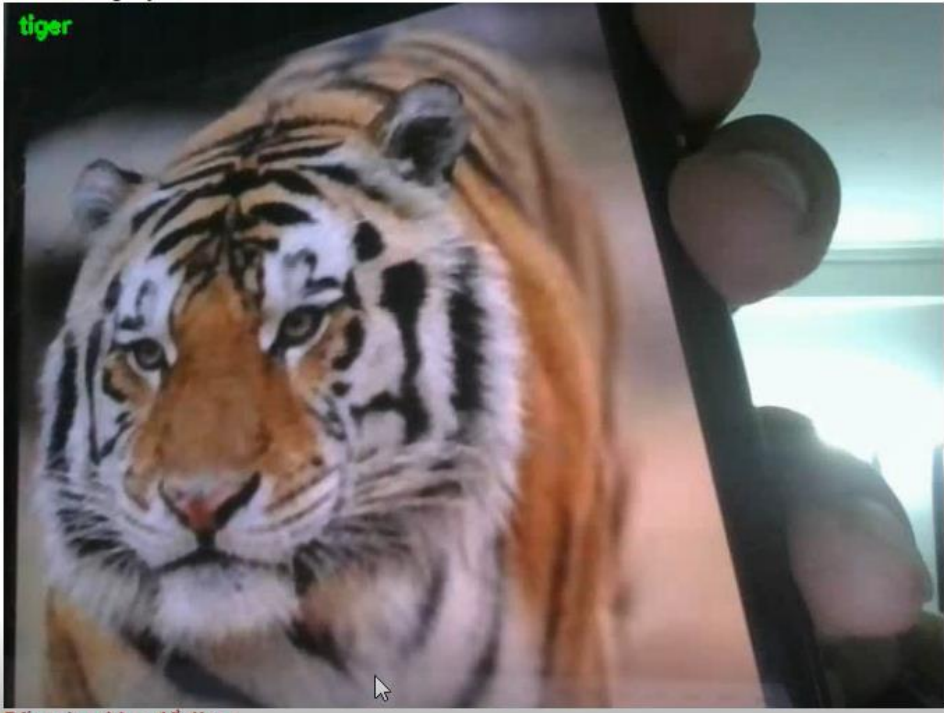


Status:Đang lấy hình ảnh...



Bấm vào video để dừng

+ Code + Text



Status: **Đang lấy hình ảnh...**

tiger

Bấm vào video để dừng

+ Code + Text

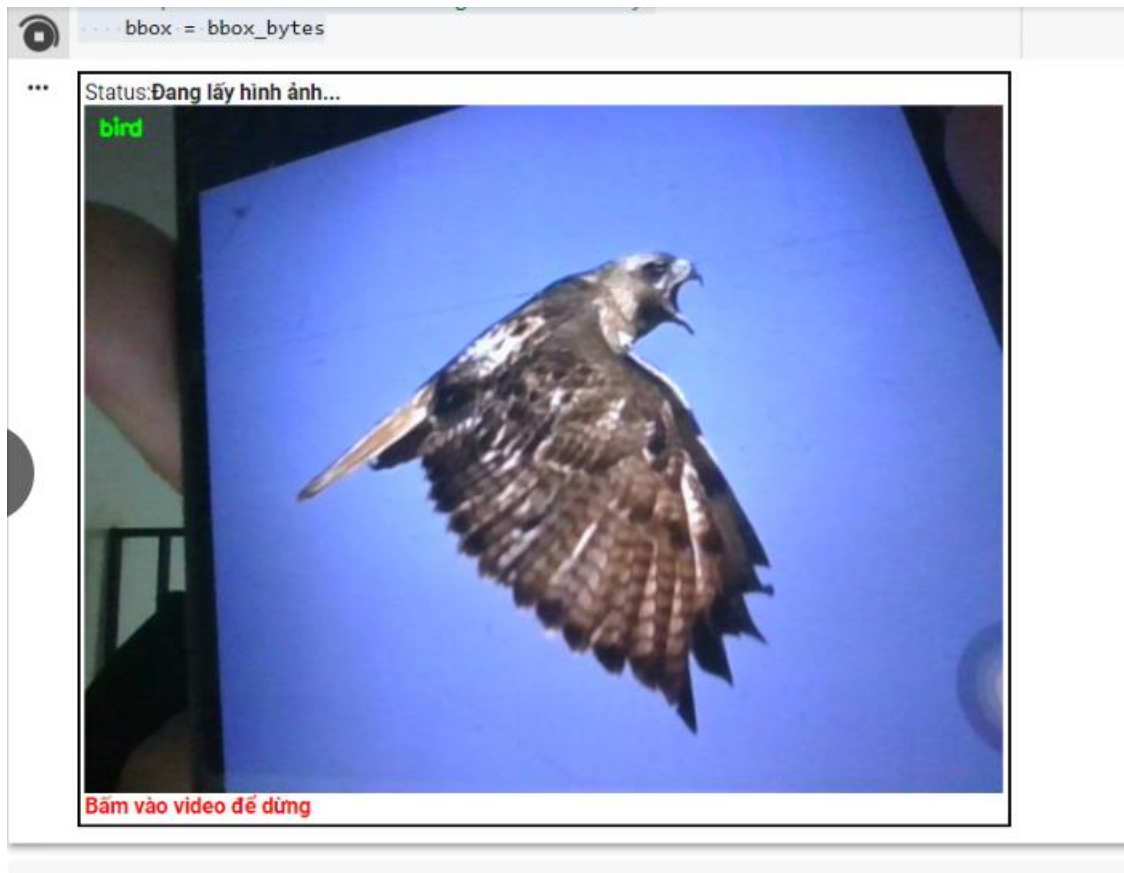


Status:Đang lấy hình ảnh...

cat



Bấm vào video để dừng



*Figure 3.2 Testing results*

Ten letters were picked at random for testing, and the results were 100 percent accurate on all ten Animals letters.

### **3.3 General accuracy results:**

Test photo set	Right	Wrong	Exact ratio
3710	87%	13%	87%



- Accuracy rate reaches 100%

### 3.4 CODE:

```
from google.colab import drive
drive.mount('/content/drive')
import numpy as np
import matplotlib.pyplot as plt
import cv2
import pandas as pd

import tensorflow as tf

from keras.models import Sequential
from keras.layers import Dropout, Flatten, Dense
from tensorflow.keras.optimizers import Adam
from keras.utils.np_utils import to_categorical
from keras.layers.convolutional import Conv2D, MaxPooling2D
from sklearn.model_selection import train_test_split

import os
import random
from keras.preprocessing.image import ImageDataGenerator

physical_devices = tf.config.list_physical_devices("GPU")
tf.config.experimental.set_memory_growth(tf.config.list_physical_devices("GPU")
)[0], True)

##### Parameters #####
path = "/content/drive/MyDrive/mydata/images" # folder chứa 10 folder của các
classes
testRatio = 0.1 # Chia 10% số ảnh cho testing
validationRatio = 0.1 # Còn lại 90% cho train => Chia 10% cho validation
# Import Ảnh
count = 0
images = []
classNo = []
myList = os.listdir(path)
print("Tổng số Classes: ", len(myList))
noOfClasses = len(myList)
print("Importing Classes.....")
for x in range(0, len(myList)):
    myPicList = os.listdir(path + "/" + str(count))
    for y in myPicList:
        curImg = cv2.imread(path + "/" + str(count) + "/" + y)
        curImg = cv2.resize(curImg, (224,224))
```



```

        images.append(curImg)
        classNo.append(count)
    print(count, end=" ")
    count += 1
print(" ")
images = np.array(images)
classNo = np.array(classNo)
# Split Data
X_train, X_test, y_train, y_test = train_test_split(images, classNo,
test_size=testRatio)
X_train, X_validation, y_train, y_validation = train_test_split(X_train,
y_train, test_size=validationRatio)

# X_train = ARRAY OF IMAGES TO TRAIN
# y_train = CORRESPONDING CLASS ID
# Kiểm tra số lượng ảnh có bằng với số LABELS cho mỗi dataset
print("Data Shapes")
print("Train", end="");
print(X_train.shape, y_train.shape)
print("Validation", end="");
print(X_validation.shape, y_validation.shape)
print("Test", end="");
print(X_test.shape, y_test.shape)
assert (X_train.shape[0] == y_train.shape[0]), " TRAINING SET: Số lượng ảnh
khác số lượng lables!"
assert (X_validation.shape[0] == y_validation.shape[0]), " VALIDATION SET: Số
lượng ảnh khác số lượng lables!"
assert (X_test.shape[0] == y_test.shape[0]), " TESTING SET: Số lượng ảnh khác
số lượng lables!"
assert (X_train.shape[1:] == (224,224,3)), " Kích thước ảnh Training SAI! "
assert (X_validation.shape[1:] == (224,224,3)), " Kích thước ảnh Validation
SAI! "
assert (X_test.shape[1:] == (224,224,3)), " Kích thước ảnh Test SAI!"
# READ CSV FILE
data = pd.read_csv('/content/drive/MyDrive/mydata/labels/bird.csv') # File
chứa tên và ID các classes
print("data shape ", data.shape, type(data))
# Xử lý ảnh
def preprocessing(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = cv2.equalizeHist(img) # Cân bằng sáng cho ảnh
    img = img / 255 # Scale ảnh về giá trị 0-1
    return img
# Lấy ảnh đã xử lý và đưa vào lại các tập
X_train = np.array(list(map(preprocessing, X_train)))
X_validation = np.array(list(map(preprocessing, X_validation)))
X_test = np.array(list(map(preprocessing, X_test)))
# Reshape về (224,224,1)

```

```

X_train = X_train.reshape(X_train.shape[0], X_train.shape[1],
X_train.shape[2], 1)
X_validation = X_validation.reshape(X_validation.shape[0],
X_validation.shape[1], X_validation.shape[2], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], X_test.shape[2], 1)
# Rotate, shift-L/R, zoom images
dataGen = ImageDataGenerator(width_shift_range=0.1,      # 0.1 = 10%      IF MORE
THAN 1 E.G 10 THEN IT REFFERS TO NO. OF PIXELS EG 10 PIXELS
                             height_shift_range=0.1,
                             zoom_range=0.2, # 0.2 MEANS CAN GO FROM 0.8 TO
1.2
                             shear_range=0.1, # MAGNITUDE OF SHEAR ANGLE
                             rotation_range=10) # DEGREES

dataGen.fit(X_train)
batches = dataGen.flow(X_train, y_train, batch_size=20) # REQUESTING DATA
GENERATOR TO GENERATE IMAGES BATCH SIZE = NO. OF IMAGES CREAED EACH TIME ITS
CALLED
X_batch, y_batch = next(batches)
# Chuyển thành one-hot
y_train = to_categorical(y_train, noOfClasses)
y_validation = to_categorical(y_validation, noOfClasses)
y_test = to_categorical(y_test, noOfClasses)
# CNN Model dựa theo LeNet
def myModel():
    num_Filters = 60
    size_of_Filter = (5, 5) # KERNEL size.
    size_of_Filter2 = (3, 3)
    size_of_pool = (2, 2) # Pooling size
    model = Sequential()
    model.add((Conv2D(num_Filters, size_of_Filter, input_shape=((224, 224,
3)[0], (224, 224, 3)[1], 1), activation='relu'))
    model.add((Conv2D(num_Filters, size_of_Filter, activation='relu'))
    model.add(MaxPooling2D(pool_size=size_of_pool))

    model.add((Conv2D(num_Filters // 2, size_of_Filter2, activation='relu'))
    model.add((Conv2D(num_Filters // 2, size_of_Filter2, activation='relu'))
    model.add(MaxPooling2D(pool_size=size_of_pool))
    model.add(Dropout(0.5)) # => Giảm Overfitting

    model.add(Flatten())
    model.add(Dense(500, activation='relu')) # HIDDEN LAYER
    model.add(Dropout(0.5))
    model.add(Dense(noOfClasses, activation='softmax')) # OUTPUT LAYER
    # COMPILÉ MODEL
    model.compile(Adam(learning_rate=0.0005), loss='categorical_crossentropy',
metrics=['accuracy'])
    return model
model=myModel()

```

```

#Train model

print(model.summary())
history = model.fit(dataGen.flow(X_train, y_train, batch_size=50),
                    epochs=170, validation_data=(X_validation, y_validation),
                    shuffle=1)

# PLOT
plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training', 'validation'])
plt.title('loss')
plt.xlabel('epoch')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'])
plt.title('Acurracy')
plt.xlabel('epoch')
plt.show()
score = model.evaluate(X_test, y_test, verbose=0)
print('Test Score:', score[0])
print('Test Accurarcy:', score[1])
# Lưu model vào file .h5 để sử dụng cho real-time
model.save('/content/drive/MyDrive/data/model.h5')
cv2.waitKey(0)
import numpy as np
import tensorflow as tf
import cv2
from keras.models import load_model
physical_devices = tf.config.list_physical_devices("GPU")
tf.config.experimental.set_memory_growth(tf.config.list_physical_devices("GPU")
)[0], True)
threshold = 0.75 #THRESHOLD của Xác Suất
font = cv2.FONT_HERSHEY_SIMPLEX
# function to convert the JavaScript object into an OpenCV image
def js_to_image(js_reply):
    """
    Params:
        js_reply: JavaScript object containing image from webcam
    Returns:
        img: OpenCV BGR image
    """
    # decode base64 image
    image_bytes = b64decode(js_reply.split(',')[1])
    # convert bytes to numpy array
    jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)

```

```

# decode numpy array into OpenCV BGR image
img = cv2.imdecode(jpg_as_np, flags=1)

return img

# function to convert OpenCV Rectangle bounding box image into base64 byte
string to be overlayed on video stream
def bbox_to_bytes(bbox_array):
    """
    Params:
        bbox_array: Numpy array (pixels) containing rectangle to overlay on
video stream.
    Returns:
        bytes: Base64 image byte string
    """
    # convert array into PIL image
    bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
    iobuf = io.BytesIO()
    # format bbox into png for return
    bbox_PIL.save(iobuf, format='png')
    # format return string
    bbox_bytes =
'data:image/png;base64,{}'.format((str(b64encode(iobuf.getvalue()), 'utf-8')))

    return bbox_bytes
from IPython.display import display, Javascript, Image
from google.colab.output import eval_js
from google.colab.patches import cv2_imshow
from base64 import b64decode, b64encode
import numpy as np
import PIL
import io
import cv2
from keras.models import load_model

# JavaScript to properly create our live video stream using our webcam as
input
def preprocessing(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = cv2.equalizeHist(img)
    img = img / 255
    return img
def video_stream():
    js = Javascript(''
        var video;
        var div = null;
        var stream;
        var captureCanvas;

```

```

var imgElement;
var labelElement;

var pendingResolve = null;
var shutdown = false;

function removeDom() {
    stream.getVideoTracks()[0].stop();
    video.remove();
    div.remove();
    video = null;
    div = null;
    stream = null;
    imgElement = null;
    captureCanvas = null;
    labelElement = null;
}

function onAnimationFrame() {
    if (!shutdown) {
        window.requestAnimationFrame(onAnimationFrame);
    }
    if (pendingResolve) {
        var result = "";
        if (!shutdown) {
            captureCanvas.getContext('2d').drawImage(video, 0, 0, 640, 480);
            result = captureCanvas.toDataURL('image/jpeg', 0.8)
        }
        var lp = pendingResolve;
        pendingResolve = null;
        lp(result);
    }
}

async function createDom() {
    if (div !== null) {
        return stream;
    }

    div = document.createElement('div');
    div.style.border = '2px solid black';
    div.style.padding = '3px';
    div.style.width = '100%';
    div.style.maxWidth = '600px';
    document.body.appendChild(div);

    const modelOut = document.createElement('div');
    modelOut.innerHTML = "<span>Status:</span>";

```

```

    labelElement = document.createElement('span');
    labelElement.innerText = 'No data';
    labelElement.style.fontWeight = 'bold';
    modelOut.appendChild(labelElement);
    div.appendChild(modelOut);

    video = document.createElement('video');
    video.style.display = 'block';
    video.width = div.clientWidth - 6;
    video.setAttribute('playsinline', '');
    video.onclick = () => { shutdown = true; };
    stream = await navigator.mediaDevices.getUserMedia(
        {video: { facingMode: "environment"}});
    div.appendChild(video);

    imgElement = document.createElement('img');
    imgElement.style.position = 'absolute';
    imgElement.style.zIndex = 1;
    imgElement.onclick = () => { shutdown = true; };
    div.appendChild(imgElement);

    const instruction = document.createElement('div');
    instruction.innerHTML =
        '<span style="color: red; font-weight: bold;">' +
        'Bấm vào video để dừng</span>';
    div.appendChild(instruction);
    instruction.onclick = () => { shutdown = true; };

    video.srcObject = stream;
    await video.play();

    captureCanvas = document.createElement('canvas');
    captureCanvas.width = 640; //video.videoWidth;
    captureCanvas.height = 480; //video.videoHeight;
    window.requestAnimationFrame(onAnimationFrame);

    return stream;
}
async function stream_frame(label, imgData) {
    if (shutdown) {
        removeDom();
        shutdown = false;
        return '';
    }
}

var preCreate = Date.now();
stream = await createDom();

```

```

        var preShow = Date.now();
        if (label != "") {
            labelElement.innerHTML = label;
        }

        if (imgData != "") {
            var videoRect = video.getClientRects()[0];
            imgElement.style.top = videoRect.top + "px";
            imgElement.style.left = videoRect.left + "px";
            imgElement.style.width = videoRect.width + "px";
            imgElement.style.height = videoRect.height + "px";
            imgElement.src = imgData;
        }

        var preCapture = Date.now();
        var result = await new Promise(function(resolve, reject) {
            pendingResolve = resolve;
        });
        shutdown = false;

        return {'create': preShow - preCreate,
                'show': preCapture - preShow,
                'capture': Date.now() - preCapture,
                'img': result};
    }
    '')

display(js)

def video_frame(label, bbox):
    data = eval_js('stream_frame("{}","{}").format(label, bbox)')
    return data
# start streaming video from webcam
video_stream()
# label for video
label_html = 'Đang lấy hình ảnh...'
# initialize bounding box to empty
bbox = ''
count = 0
#Load model nhận diện biển báo
model_file_path = "/content/drive/MyDrive/data/model.h5"
model5 = load_model(model_file_path)
classes =
['bird','cat','dog','giraffe','horse','lion','monkey','squirrel','tiger']

while True:
    # Đọc ảnh trả về từ JS
    js_reply = video_frame(label_html, bbox)

```

```

if not js_reply:
    break
# Convert JS response to OpenCV image
frame = js_to_image(js_reply["img"])

# Resize để đưa vào model
frame_p = cv2.resize(frame, dsize =(224, 224))
frame_p = preprocessing(frame_p)
tensor = np.expand_dims(frame_p, axis =0)
# Feed vào mạng
pred = model5.predict(tensor)
class_id = np.argmax(pred)
class_name = classes[class_id]
#Vẽ lên 1 ảnh để tạo nửa overlay
bbox_array = np.zeros([480,640,4], dtype=np.uint8)

bbox_array = cv2.putText(bbox_array, "{}".format(class_name),
                        (10, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.5,
                        (0, 255,0), 2)

bbox_array[:, :,3] = (bbox_array.max(axis = 2) > 0 ).astype(int) * 255
# convert overlay of bbox into bytes
bbox_bytes = bbox_to_bytes(bbox_array)
# update bbox so next frame gets new overlay
bbox = bbox_bytes
### CREATE VIRTUAL DISPLAY ###
!apt-get install -y xvfb # Install X Virtual Frame Buffer
import os
os.system('Xvfb :1 -screen 0 1600x1200x16 &') # create virtual display
with size 1600x1200 and 16 bit color. Color can be changed to 24 or 8
os.environ['DISPLAY']=':1.0' # tell X clients to use our virtual DISPLAY
:1.0.
### INSTALL GHOSTSCRIPT (Required to display NLTK trees) ###
!apt install ghostscript python3-tk

```

## Chapter 4: Conclusion and development direction

### 4.1 Conclusion:

- The CNN network model works well to identify very accurately, the reason is because of the number of matching layers.
- Can recognize both color and grayscale images.



- Although the percentage of accuracy is high, there are still some misidentified images (not completely accurate).

#### **4.2 Development direction:**

- This topic can develop the emotional recognition of animals through each face for biologists.
- Further improvements to make this recognition CNN's accuracy higher and as accurate as possible.

## REFERENCES

- [1] [Convolutional neural network - Wikipedia](#)
- [2] **Paul Michael**, How Fast is Technology Advancing? [Growth Charts & Statistics] 2022, MediaPeanut  
[How Fast Is Technology Advancing? \[Growth Charts & Statistics\] 2022 \(mediapeanut.com\)](#)
- [3] [Artificial intelligence - Wikipedia](#)
- [4] **Nguyen Ba Hung B**, Tăng cường dữ liệu trong deep learning, VIBLO ASIA  
[Tăng cường dữ liệu trong deep learning \(viblo.asia\)](#)
- [5] **Nguyen Chien Thang**, Có ít dữ liệu, làm sao train model? – Chương 1. Data Augment cho ảnh, MIAI.VN  
[Có ít dữ liệu, làm sao train model? - Chương 1. Data Augment - Mì AI \(miai.vn\)](#)
- [6] **The TensorFlow Authors**, 105c02\_dogs\_vs\_cats\_with\_augmentation, Tensorflow  
[105c02\\_dogs\\_vs\\_cats\\_with\\_augmentation.ipynb - Colaboratory \(google.com\)](#)
- [7] **OPENCV.ORG**, About  
[About - OpenCV](#)
- [8] **GeeksforGeeks**, OS Module in Python with Examples, geeksforgeeks.org  
[OS Module in Python with Examples - GeeksforGeeks](#)
- [9] **Numpy.org**, numpy  
[NumPy](#)
- [10] **matplotlib.org**, matplotlib.pyplot  
[matplotlib.pyplot — Matplotlib 3.5.2 documentation](#)