

**UNIVERSITY OF TECHNOLOGY AND EDUCATION
HIGH QUALITY TRAINING FACULTY**



HCMUTE

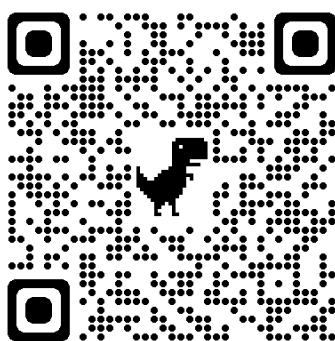
ARTIFICIAL INTELLIGENCE

FINAL PROJECT REPORT

**IDENTIFY 15 PETS & THEIR
CHARACTERISTICS REAL-TIME**

ADVISOR: Nguyen Truong Thinh

STUDENT: Vo Dinh Nghi ID: 19146097



Ho Chi Minh city, 22nd June, 2022

INTRODUCTION

Deep Learning is an algorithm based on a number of ideas from the brain to the acquisition of many layers of representation, both concrete and abstract, thereby clarifying the meaning of data types. Deep Learning is applied in image recognition, speech recognition, natural language processing.

Currently, a lot of recognition problems use Deep Learning, because it can solve problems with a large number of variables, large input size parameters with superior performance and accuracy compared to other algorithms. Traditional classification method, building intelligent systems with high accuracy.

In this paper, author studies a convolutional neural network (CNN - Convolutional Neural Network) which is one of the advanced Deep Learning models for the problem of animal recognition from video. Currently, society is gradually approaching the era of industrial revolution 4.0.

With the industrial revolution 4.0, automation levels, as well as machine learning, are at a high level, which can replace humans from many jobs, contributing to liberating labor. In addition, data mining also brings many optimizations to business models, as well as to society.

Towards the industrial revolution 4.0, automation solutions are needed for public or business models such as hospitals, stores or supermarkets. In these solutions, the intelligent system will automatically analyze the number of people going in/out, or identify the time of arrival of loyal customers. Within the scope of the topic, the thesis applies two solutions for human identification and face recognition using CNN to apply and build real systems. The methods of recognizing people as well as faces through reality have been highly accurate and commercialized.

Contents

CHAPTER 1: AN OVERVIEW OF NEURAL NETWORKS AND INTRODUCTION OF CONVOLUTIONAL NEURAL NETWORKS	1
1.1. Neural Networks	1
a) History of Neural Networks.....	1
b) Biological Neuron	2
c) Artificial Neuron Networks	2
d) Activation function models of artificial neural networks	4
1.2 Artificial Neural Network	8
a) Introduction to artificial neural networks	8
b) Some types of neural networks	8
CHAPTER 2: PET IDENTIFICATION USING CONVOLUTIONAL NEURAL NETWORKS	13
2.1 Introduction	13
2.2 Using convolutional neural networks to identify pet.....	13
a) Image data processing	13
b) Create and Train model.....	16
c) Program to run real-time recognition	18
d) Model 's accuracy	22
e) Result.....	23
f) Web App	25

CHAPTER 1: AN OVERVIEW OF NEURAL NETWORKS AND INTRODUCTION OF CONVOLUTIONAL NEURAL NETWORKS

1.1. Neural Networks

a) History of Neural Networks

In 1943, neuroscientist Warren McCulloch and mathematician Walter Pitts wrote a book on how neural networks work. And they performed a simulation of a simple neural network on an electrical circuit.

In 1949, Donald Hebb wrote the book Organization of Behavior. The main emphasis is that whichever neural network is used the most will be enhanced.

In 1959, David Hubel and Torsten Wiesel published the book "Receptive fields of single neurons in the cat's striate cortex", describing the response of optic neurons in cats, as well as how cats record remembers and recognizes shapes on its cortical architecture.

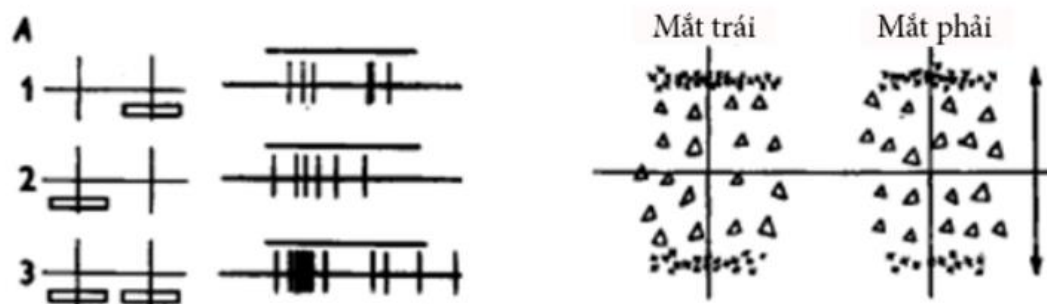


Fig 1.1 Image of David Hubel and Torsten Wiesel's experiments on cats

In 1989, Yann LeCun applied a back-propagation neural network learning algorithm to Fukushima's convolutional neural network architecture. A few years later, LeCun announced LeNet-5.

It can be said that LeNet-5 is one of the most primitive convolutional neural networks, but its imprints still exist today, which can be seen through some of the essential components that neural networks have. Convolution of today is still in use.

And thus by 1990's, Neural networks were definitely back, this time truly catching the imagination of the world and finally coming to par with, if not overtaking, its expectations. Yet again, we are asking the same questions of AI, and projecting onto it, our all too human fears, and yet again we are farther than we think from bowing in deference to our digital overlords.

b) Biological Neuron

Neuron

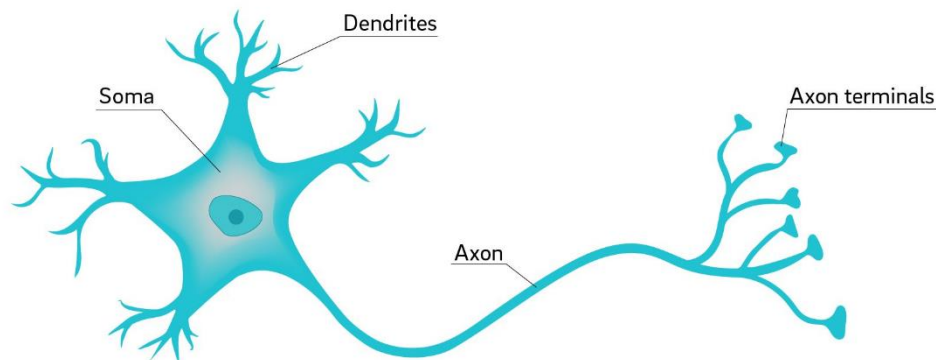


Fig 1.2: Structure of a biological neuron

A neuron comprises three major parts:

The cell body (also called Soma): collecting the signal coming through dendrites (input bres) and distributing the signal further through axons

The dendrites : are like fibers branched in different directions and are connected to many cells in that cluster. Dendrites receive the signals from surrounding neurons

The axon : transmits the signal to the other neurons .Axon is a long fiber that transports the output signal as electric impulses along its length. Each neuron has one axon. Axons pass impulses from one neuron to another like a domino effect.

c) Artificial Neuron Networks

Structure and formula of an artificial neuron

Based on the structure of a biological neuron, research and programming scientists have come up with the architecture of an artificial neuron

networks

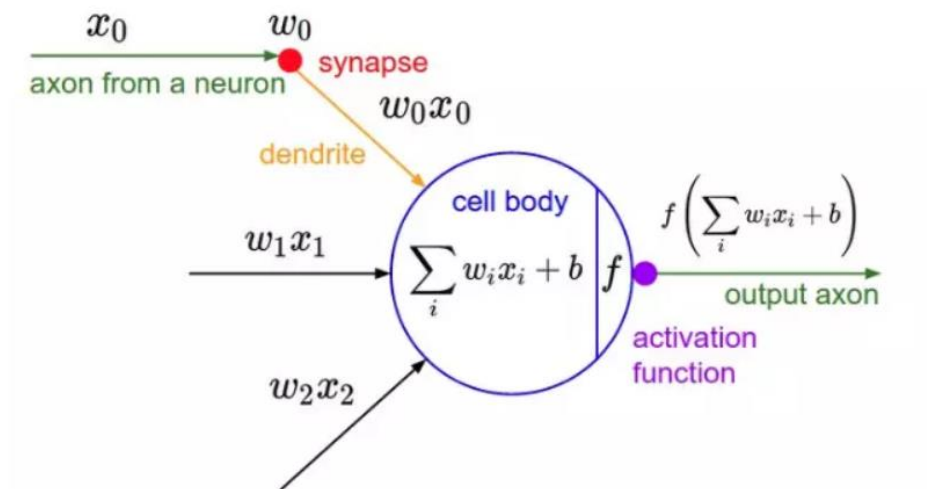


Fig 1.3: The formula and structure of an artificial neuron
An artificial neural network can be simply described as follows:

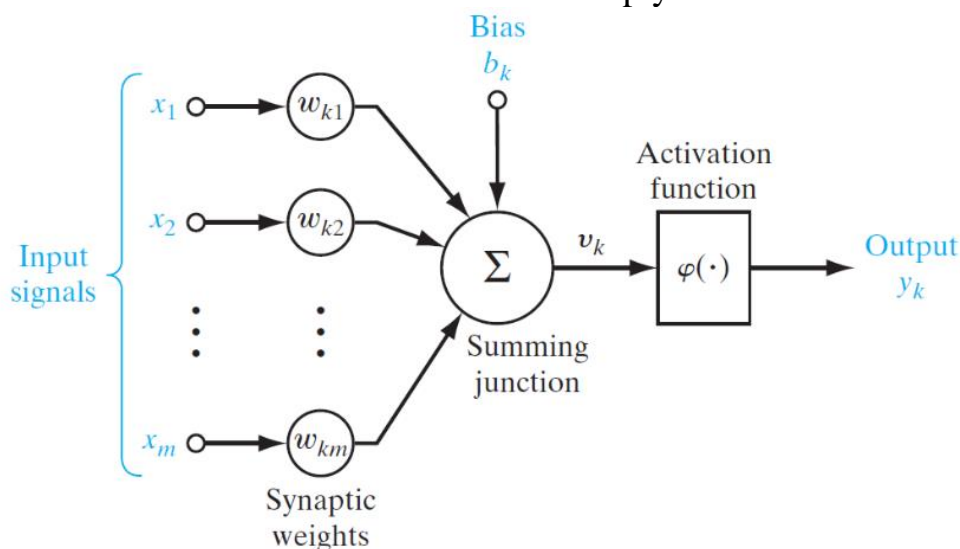


Fig 1.4: Image of an artificial neuron

Explain:

- Input signals : Input attributes of a neuron. The number of input attributes is usually more than one, because the input raw data is usually a multi-dimensional vector, or many front-layer neurons connect to a later-layer neuron.

- Synaptic Weights: Links are expressed in terms of strength and weakness by a value called the link weight. Combined with the transmitters, the signal to other artificial neurons will be calculated by w_i ,

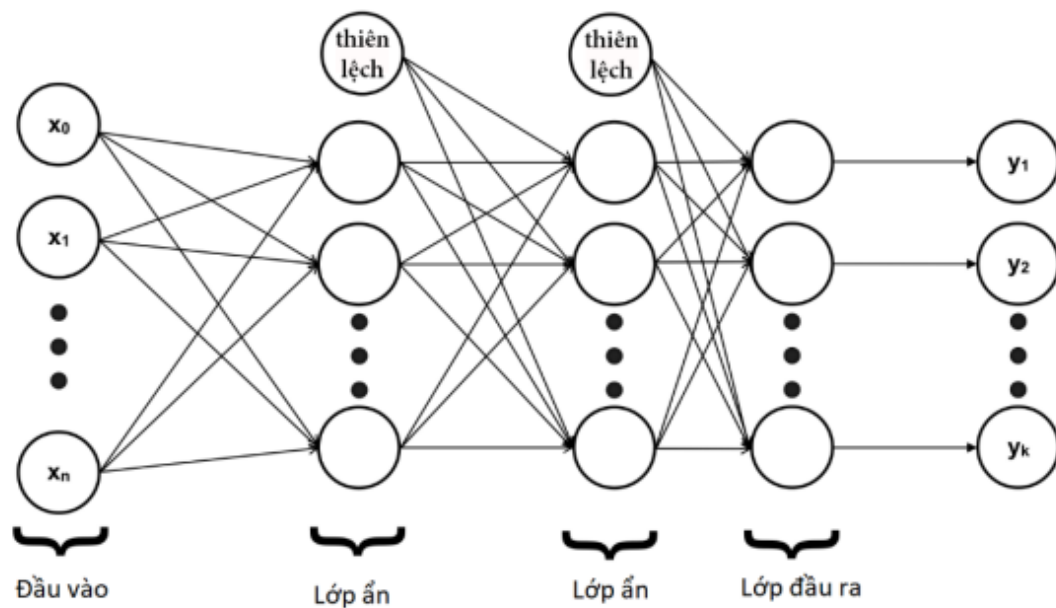


Fig 1.5: Image of biased position added in reality

- Summing function: The sum of the products of the inputs with the link weights simulates the joints. Then go through the sum function to calculate the value before putting it into the transfer function
- Bias: The bias is inserted after the sum function is calculated, producing the final value before being fed to the transfer function. The purpose of adding bias is to shift the function of the activation function to the left or right, which is useful when the network is trained.
- Activation function : This function is used to calculate the value of the output based on the value of the Total function.

d) Activation function models $f(x) = \frac{1}{1 + e^{-x}}$ al neural networks

Sigmoid

- Function :
- Derivative of the function: $f'(x) = f(x)(1 - f(x))$

The Sigmoid function is used because its threshold is in the range (0, 1). Therefore, this function is used a lot for output probabilistic prediction models, i.e. the result only exists between 0 and 1: when the input is a large positive number, the output of the sigmoid function is close to 1. When less than 0, the output is close to 0. However, the optimization of this function is difficult, because if the input value of the function is a very large number, the output of the function will be as close to the two ends as possible. 1 or 0, so the convergence speed will be very slow.

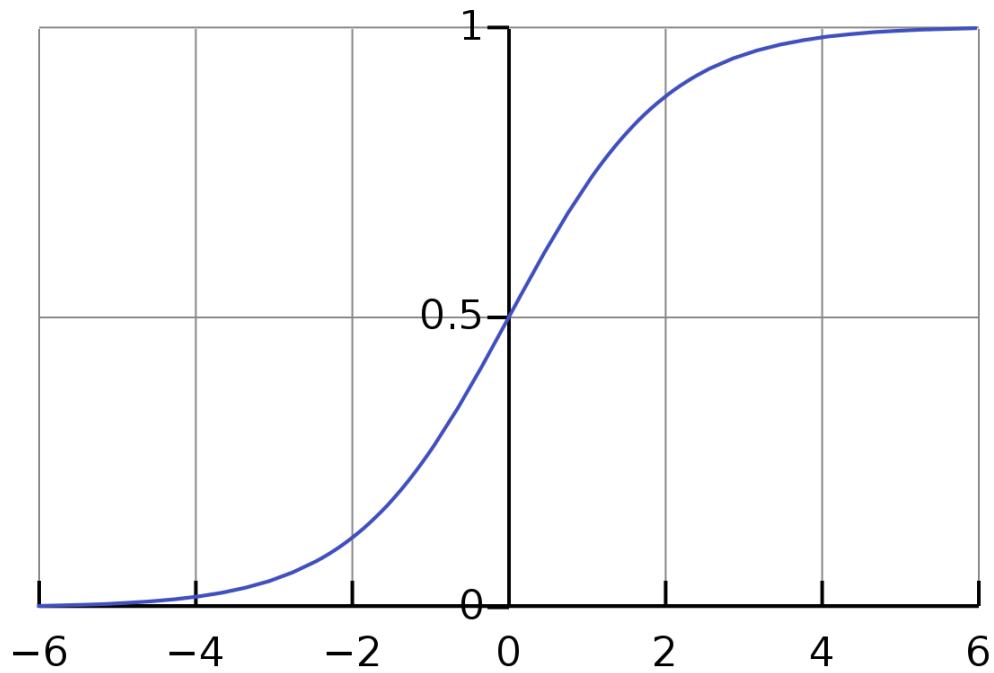


Fig 1.6 : Graph of the sigmoid function

TanH

- Function: $f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$
- Derivative of the function: $f'(x) = 1 - f(x)^2$

The TanH function is used because its output is in the range $[-1, 1]$, suitable for models with three output values: negative, neutral (0), and positive. We can see this more clearly in the illustration.

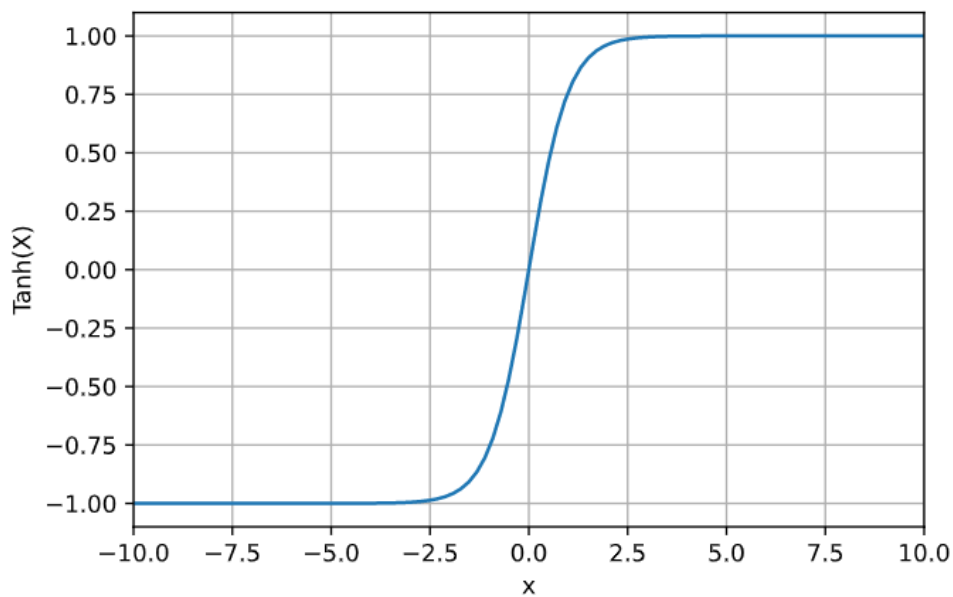


Fig 1.7: Graph of the TanH function

Linear

- Function: $f(x) = x$
- Derivative of the function: $f'(x) = 1$

The linear function applies an identity operation on the data with the output data proportional to the input data.

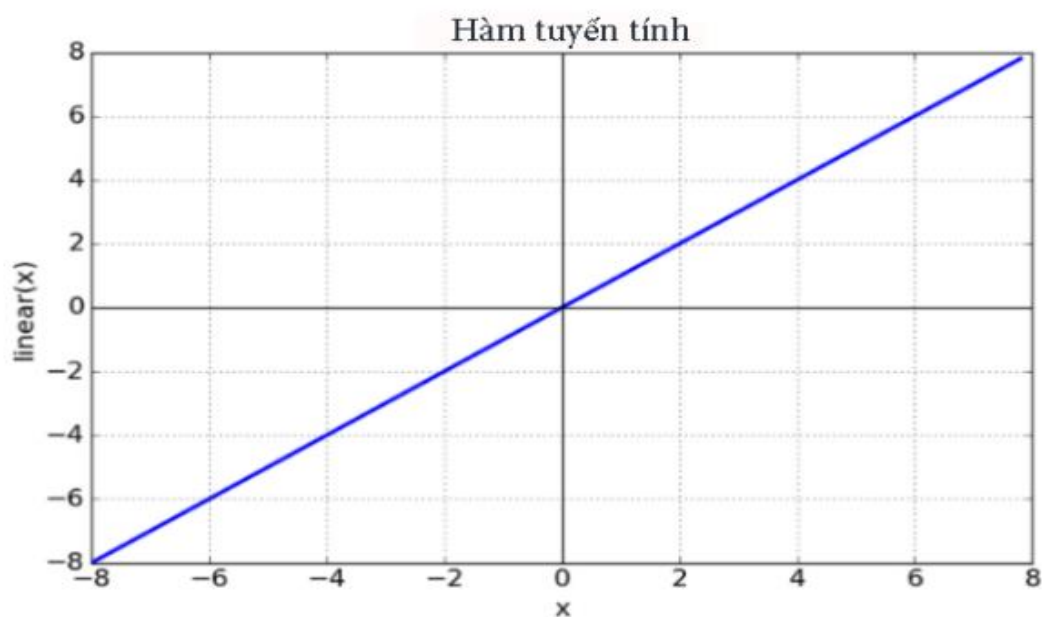


Fig 1.8: Graph of the Linear function

RELU

- Function: $f(x) = \begin{cases} 0 & \text{khi } x < 0 \\ x & \text{khi } x \geq 0 \end{cases}$
- Derivative of the function: $f'(x) = \begin{cases} 0 & \text{khi } x < 0 \\ 1 & \text{khi } x \geq 0 \end{cases}$

The RELU function applies to cases where the output is in the range $(0, +\infty)$. The RELU function has a very fast computation speed, assigning negative values to become 0 instantly, suitable for training from standard data. However, this causes the RELU function to not properly map negative values.

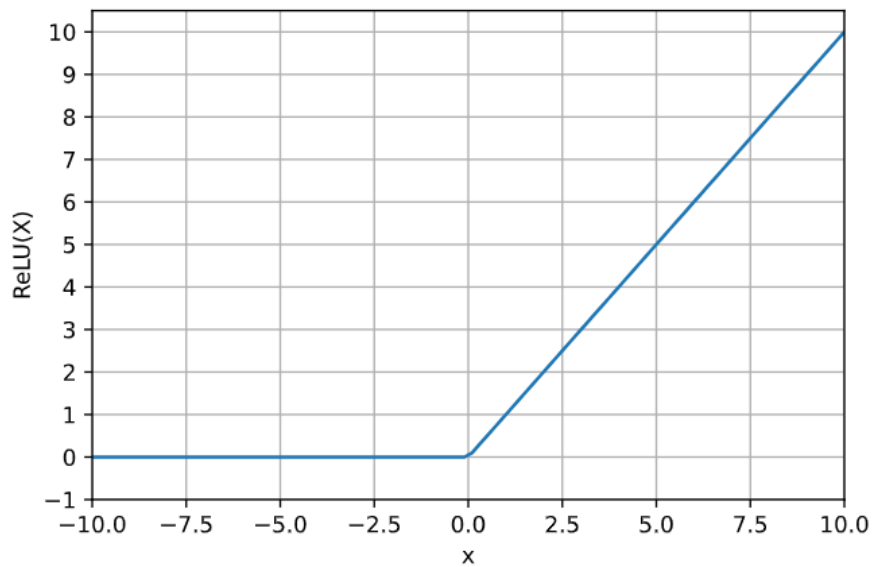


Fig 1.9: Graph of the RELU function

ELU

- Function:
$$f(x) = \begin{cases} \alpha(e^x - 1) & \text{when } x < 0 \\ x & \text{when } x \geq 0 \end{cases}$$
- Derivative of the function:
$$f'(x) = \begin{cases} f(x) + \alpha & \text{when } x < 0 \\ 1 & \text{when } x \geq 0 \end{cases}$$

The ELU function is a variant of the RELU function. The function is usually used when its output threshold is in the range $(-1, +\infty)$. The ELU function overcomes the RELU function's limitation of mapping negative values.

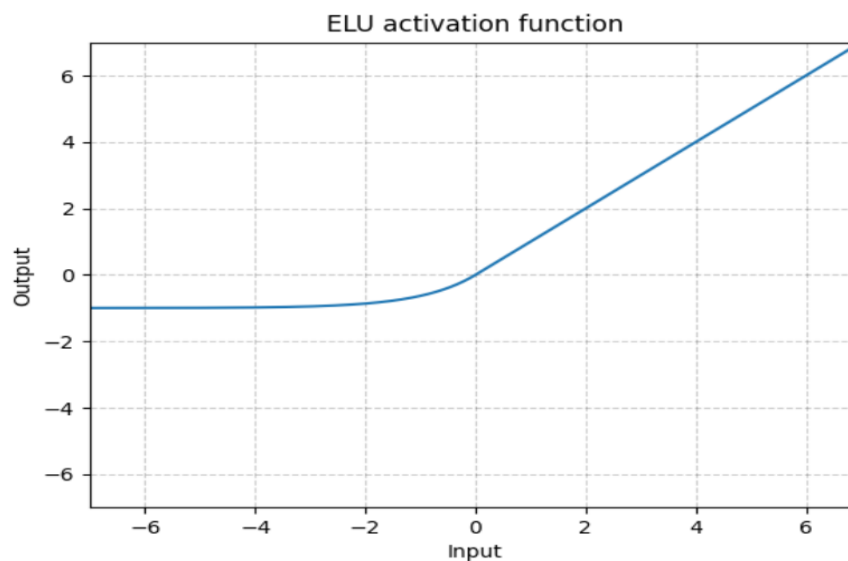


Fig 1.10: Graph of the ELU function

1.2 Artificial Neural Network

a) Introduction to artificial neural networks

Artificial Neural Network (ANN) is a series of programming algorithms, simulating based on how neural networks work in the brains of living organisms. An artificial neural network is used to find out the relationship of a data set through an architectural design that contains many hidden layers, each layer containing many neurons. The neurons are connected to each other, and the strength of the links is expressed by the link weights.

Conventional programming can do a lot of large software, like calculating simulations of nuclear explosions in supercomputers in laboratories, or reconstructing cells at the molecular level for analysis of experiments. drug test.

A supercomputer can do many billions of calculations per second, but conventional programmers have difficulty recognizing simple patterns, such as human face recognition, which a biological brain cannot do. much faster and more accurate processing. Applied with deep learning techniques, artificial neural networks are now being applied to solve problems that are difficult to solve by conventional logic programming. Therefore, artificial neural networks are rapidly gaining popularity, and are trending in many fields.

b) Some types of neural networks

There are two main types of neural networks:

- + **Feedforward neural networks**
- + **Recurrent neural networks.**

The feedforward and regression networks are illustrated as follows:

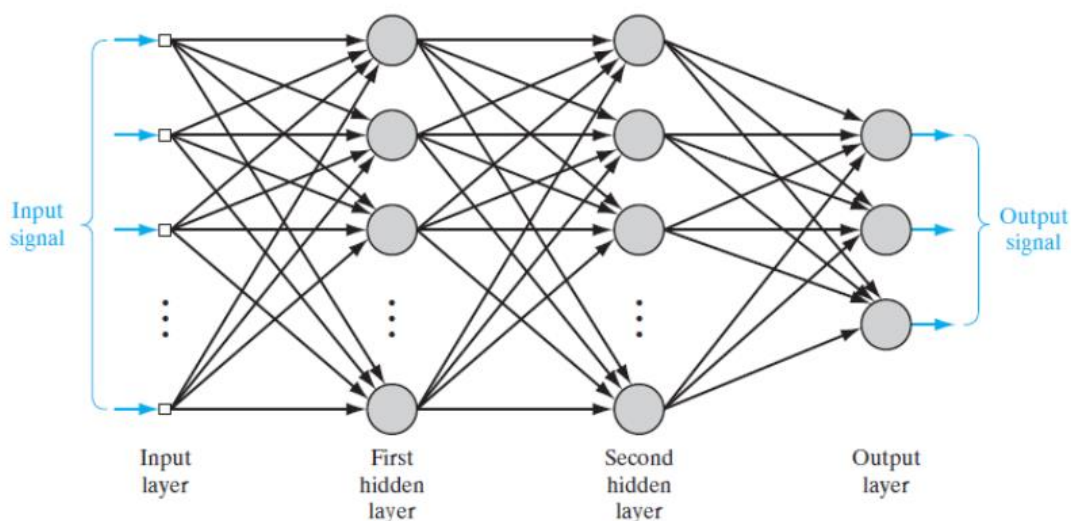


Fig 1.11: illustrate the architecture of a feedforward neural network with 4 layers

It is easy to see that in a feedforward neural network, the neurons in the hidden layer are connected to the neurons in the $n+1$ layer. Since there are many hidden layers, we can see that the linear network extends in space, and that there is no cyclic path in the network. Neural networks are very popular nowadays.



Fig 1.12: Regressive Neural Network

Another type is the regressive neural network. Unlike feedforward neural networks, recurrent neural networks have at least one cyclic path. We can see it in the illustration above. Since there is a cyclic path, the recurrent neural network can cause an infinity loop. However, cyclic neural networks have an important application that they can recognize for different time periods, as shown in the following illustration:

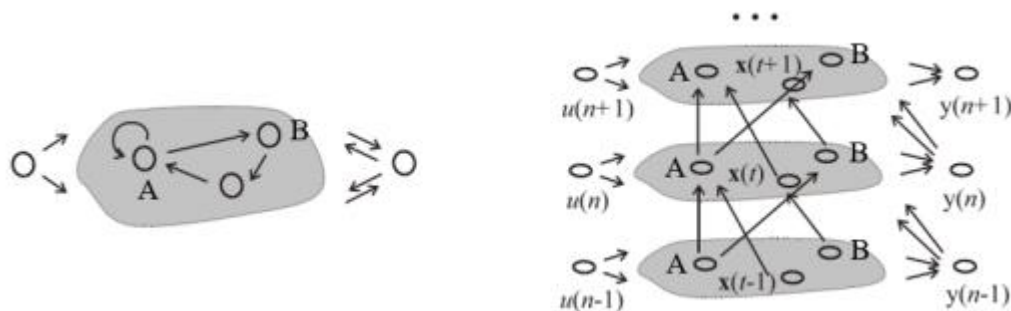


Fig 1.13: How to train a regressive neural network

As the example above, there is a node A that connects to node B and a cycle to node A itself. Regressive neural networks do not process cyclic paths and connections at the same time. Regression neural network assumes that the output of node A in time n is the input of node B and node A in time $n + 1$.

Therefore, in addition to the spatial stretching property when connecting to the layers next neuron, the recurrent neural network is also deep in time. So, regressive neural networks can model context-changing systems. For example, recurrent neural networks are often used in contextual language processing. A 13-neural regressor network can deal with long-term dependencies over time, such as Long Short Term Memory networks.

+ **Neural network back propagation**

The back-propagation algorithm is briefly described as follows:

Step 1: Spread. The propagation phase has two steps, forward propagation and back propagation. The forward propagation step is to input the training data into the neural networks and compute the output. Then, based on the output, compare with the training data. We can use backpropagation to inversely update the weights of neurons in previous layers.

Step 2: Update weights. The network updates the values of the neuron's weights according to the error of the output.

Step 3: Repeat the above two steps. Repeat steps one and two until the error is minimal. Then finish the training.

Convolutional Neural Network

Convolutional neural networks are one of the special feedforward networks. Convolutional neural networks are one of the most popular and advanced deep learning models available today. Most of today's image recognition and processing systems use convolutional neural networks because of their fast processing speed and high accuracy.

In traditional neural networks, layers are considered one-dimensional, while in convolutional neural networks, layers are considered three-dimensional, including: height, width and depth (Figure 1.11). Convolutional neural networks have two important concepts: local connectivity and parameter sharing. These concepts contribute to reducing the number of weights that need to be trained, thereby increasing the speed of computation.

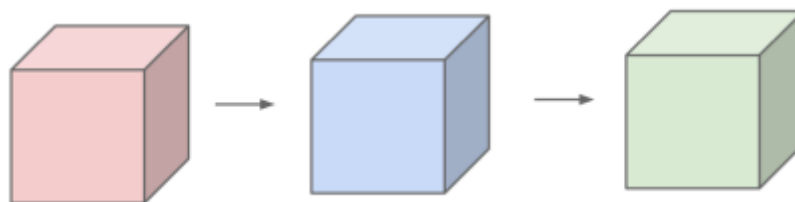


Fig 1.14: Layers in CNN are 3-dimensional

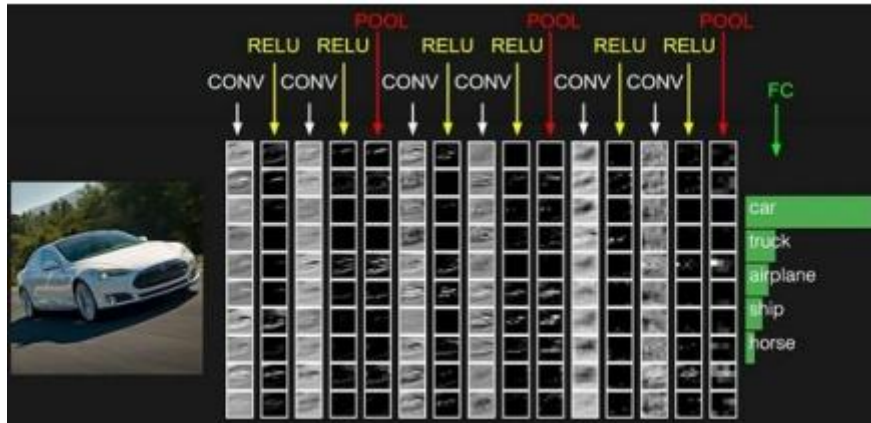


Fig 1.15: The illustration shows an example using CNN to classify objects

Convolutional neural network model

There are three main layers to building the architecture for a convolutional neural network:

1. Convolutional layer;
2. Pooling layer;
3. Floor is fully-connected.

The fully connected layer is like regular neural networks, and the convolutional layer performs multiple convolutions on top of the previous layer. Pooling can reduce the sample size per 2x2 block of the previous layer. In convolutional neural networks, the network architecture often overlaps these three layers to build the full architecture. An illustrative example of a full convolutional neural network architecture:

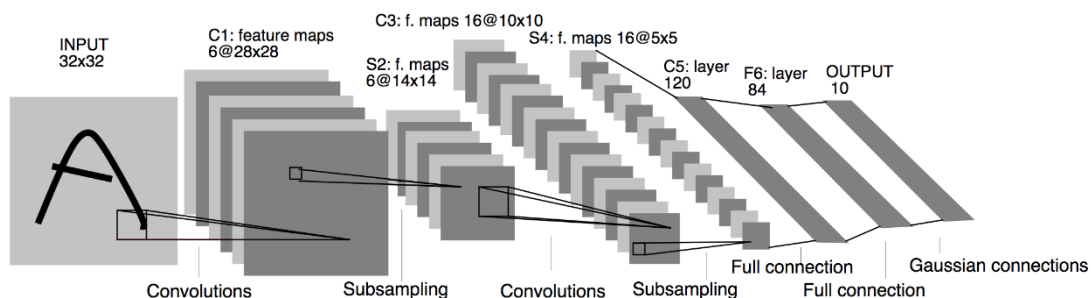


Fig 1.16: Example illustrating the structure of CNNs – LeNet – 5

Build a convolutional neural network

A fully connected network

In image processing, the information of an image is pixels. If we use a fully connected network, we will have a lot of parameters. For example, an RGB image of 512x512 pixels will have 786432 ($= 512 \times 512 \times 3$)

input parameters. So, if we use the neural network architecture in the following figure:

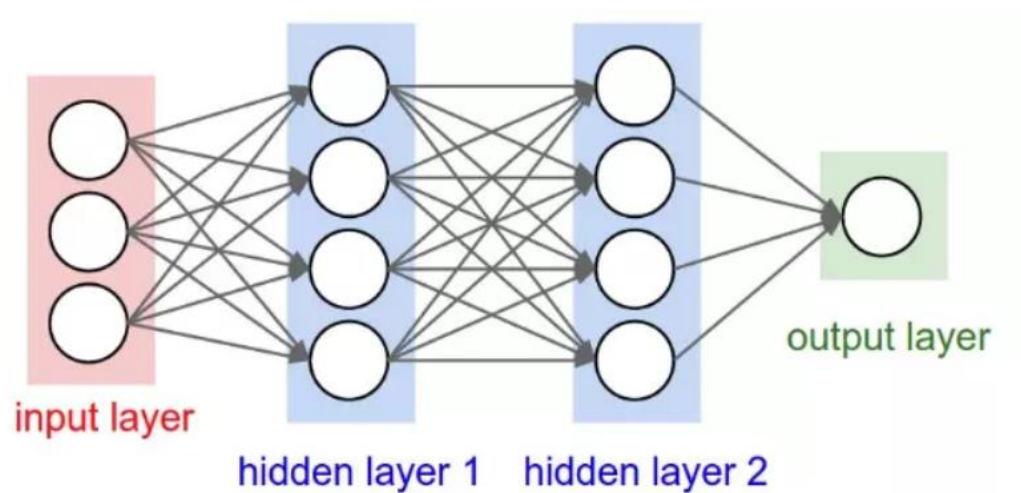


Fig 1.17: Image of a fully connected neural network

The figure above shows that if a fully connected neural network is applied, the entire network architecture will need to compute more than 3 million neurons. The large number of neurons makes the whole learning process very slow and leads to an overload compared to the computing capacity of the current computer.

Through several studies of image processing, researchers have found that features in an image are often local, and researchers pay attention to low-level features first when processing images. Thus, the network architecture can convert a fully connected network to a locally connected network, reducing computational complexity. This is one of the main ideas in CNN. We can see it more clearly in the following figure:

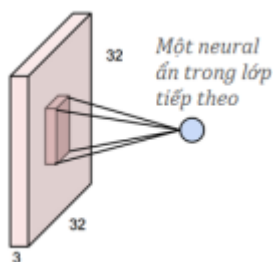


Fig 1.18: Convolution of a small matrix to generate input for a hidden layer neuron

Like normal image processing, we can locally connect a block of matrix to a neuron. Typical block sizes are 3x3, 5x5, or 7x7. The physical meaning of the block is like a sliding window (sliding window is one of the methods of image processing).

That way, the number of parameters can be reduced to a very small amount without causing any loss or loss of information, since the normal image is often spatially iterative.

To extract more information, neural networks connect the block together with another neuron. Depth in layers is the number of times we connect an area to different neurons. For example, the network connects the same area with 5 different neurons. So the depth is five in the new stratum. We can see it more clearly in the following figure:

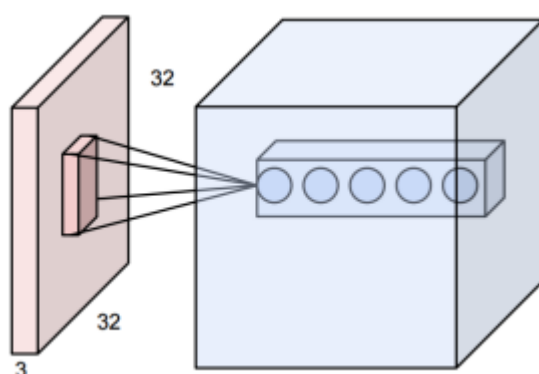


Fig 1.19: Example of a convolutional layer

CHAPTER 2: PET IDENTIFICATION USING CONVOLUTIONAL NEURAL NETWORKS

2.1 Introduction

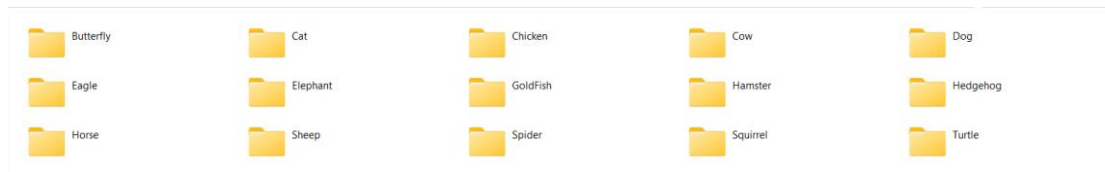
Currently, society is gradually approaching the era of industrial revolution 4.0. With the industrial revolution 4.0, automation levels, as well as machine learning, are at a high level, which can replace humans from many jobs, contributing to liberating labor. In addition, data mining also brings many optimizations to business models, as well as to society.

Towards the industrial revolution 4.0, automation solutions are needed for public or business models such as hospitals, stores or supermarkets. In these solutions, the intelligent system will automatically analyze the number of people going in/out, or identify the time of arrival of loyal customers.

2.2 Using convolutional neural networks to identify pet

a) Image data processing

Build a machine learning model using CNN that is able to identify different animals from photographs. Because there are hundreds of species of animals in the wild so in this article .15 typical animal species will be selected to be included in the model.



Learn about data: Data set - data includes 15 types of animals: Butterfly, Cat, Chicken, Cow, Dog, Eagle, Elephant, Gold Fish, Hamster, Sheep, Hedgehog, Horse, Spider, Squirrel, Turtle. The total number of images for each animal species is 100 photos, some species have more than 3000 photos, the number of photos in the whole dataset is 28,867 photos of the 15 species mentioned above.

The total number of images in the Train set is 18,474 images, each animal species is about 300-500 images, the total number of images in the validation file is 5774 images, each animal species is 200-300 images.

```
##### Parameters #####
path = "/content/drive/MyDrive/final_data" # folder chứa 15 folder của các classes
testRatio = 0.2 # Chia 20% số ảnh cho testing
validationRatio = 0.2 # Còn lại 80% cho train => Chia 20% cho validation
```

The dataset is divided into two parts on validation and test, each part has images of 15 species, in which the rate of training accounts for 80%, validation accounts for 20%.

Declare the directory path containing the data image used for training and testing the step model.

```
# 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, (32,32))
        images.append(curImg)
        classNo.append(count)
    print(count, end=" ")
    count += 1
print(" ")
images = np.array(images)
classNo = np.array(classNo)
```

```
Tổng số Classes: 15
Importing Classes.....
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```

Import images using os and resize images using cv2 in 32x32 format and name the data folder with numbers from 0 to 14 for easy running of the cv2 algorithm.

```
# 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
```

Split Data into train and test

```
[ ]
# 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 labels!"
assert (X_validation.shape[0] == y_validation.shape[0]), " VALIDATION SET: Số lượng ảnh khác số lượng labels!"
assert (X_test.shape[0] == y_test.shape[0]), " TESTING SET: Số lượng ảnh khác số lượng labels!"
assert (X_train.shape[1:] == (32,32,3)), " Kích thước ảnh Training SAI! "
assert (X_validation.shape[1:] == (32,32,3)), " Kích thước ảnh Validation SAI! "
assert (X_test.shape[1:] == (32,32,3)), " Kích thước ảnh Test SAI!"
```

```
Data Shapes
Train(18474, 32, 32, 3) (18474,)
Validation(4619, 32, 32, 3) (4619,)
Test(5774, 32, 32, 3) (5774,)
```

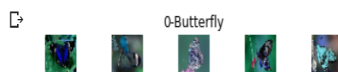
Check the amount of data used for training and validation and testing

```
# READ CSV FILE
data = pd.read_csv('/content/drive/MyDrive/labes.csv') # File chứa tên và ID các classes
print("data shape ", data.shape, type(data))
```

```
data shape (15, 2) <class 'pandas.core.frame.DataFrame'>
```

	A	B
1	Name	ID
2	Butterfly	0
3	Cat	1
4	Chicken	2
5	Cow	3
6	Dog	4
7	Elephant	5
8	Horse	6
9	Sheep	7
10	Spider	8
11	Squirrel	9
12	Hamster	10
13	Eagle	11
14	GoldFish	12
15	Hedgehog	13
16	Turtle	14
17		

```
# Biểu diễn ngẫu nhiên một số ảnh mẫu
num_of_samples = []
cols = 5
num_classes = noOfClasses
fig, axs = plt.subplots(nrows=num_classes, ncols=cols, figsize=(5, 300))
fig.tight_layout()
for i in range(cols):
    for j, row in data.iterrows():
        x_selected = X_train[y_train == j]
        axs[j][i].imshow(x_selected[random.randint(0, len(x_selected) - 1), :, :], cmap=plt.get_cmap("gray"))
        axs[j][i].axis("off")
    if i == 2:
        axs[j][i].set_title(str(j) + "-" + row["Name"])
        num_of_samples.append(len(x_selected))
```



Read CSV File and labels of classes . Check if the device has the correct label. Image classification recognition is a supervised math problem, so the training and testing datasets must be labeled.

```

[8] # 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

[9] # 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)))

[10] # Reshape về (32,32,1)
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)

[11] # Rotate, shift-L/R, zoom images # nó sẽ quay hình qua lại cho mình để có thêm data
dataGen = ImageDataGenerator(width_shift_range=0.1, # 0.1 = 10% IF MORE THAN 1 E.G 10 THEN IT REFERS 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 GENRATOR TO GENERATE IMAGES BATCH SIZE = NO. OF IMAGES CREAED EACH TIME ITS CALLED
X_batch, y_batch = next(batches)

[ ] # Chuyển thành one-hot # chuyển từ numerical sang array để có thể train được
y_train = to_categorical(y_train, noOfClasses)
y_validation = to_categorical(y_validation, noOfClasses)
y_test = to_categorical(y_test, noOfClasses)

```

Synthesize the steps and how to process images for the model

b) Create and Train model

The model includes two CNN layers:

... Model: "sequential_6"

Layer (type)	Output Shape	Param #
conv2d_20 (Conv2D)	(None, 220, 220, 60)	1560
conv2d_21 (Conv2D)	(None, 216, 216, 60)	90060
max_pooling2d_10 (MaxPooling2D)	(None, 108, 108, 60)	0
conv2d_22 (Conv2D)	(None, 106, 106, 30)	16230
conv2d_23 (Conv2D)	(None, 104, 104, 30)	8130
max_pooling2d_11 (MaxPooling2D)	(None, 52, 52, 30)	0
dropout_10 (Dropout)	(None, 52, 52, 30)	0
flatten_5 (Flatten)	(None, 81120)	0
dense_10 (Dense)	(None, 500)	40560500
dropout_11 (Dropout)	(None, 500)	0
dense_11 (Dense)	(None, 10)	5010
=====		
Total params: 40,681,490		
Trainable params: 40,681,490		
Non-trainable params: 0		

The first CNN layer consists of 60 filters with dimensions of 5x5. The first CNN layer connects the input to the second CNN layer. Using the Conv2D function is a method to create a complex where the first parameter is the number of filters, the second parameter is the filter size. The MaxPooling2D function is used to reduce the size of the data while preserving important properties. The relu function converts a negative value to zero and keeps the positive value unchanged.

```
# 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=((32, 32, 3)[0], (32, 32, 3)[1], 1), activation='relu'))
    model.add(Conv2D(num_Filters, size_of_Filter, activation='relu'))
    model.add(MaxPooling2D(pool_size=size_of_pool))
```

The second CNN layer consists of 60 filters with dimensions of 3x3. The first CNN layer connects the input to the second CNN layer. Using the Conv2D function is a method to create a complex where the first parameter is the number of filters, the second parameter is the filter size. The MaxPooling2D function is used to reduce the size of the data while preserving important properties. The relu function converts a negative value to zero and keeps the positive value unchanged.

```
# 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.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
```

Flatten function to convert an image from a matrix to a one-dimensional array. The Dense function is a hidden layer containing neurons that connect all the neurons in the front.

```
model.add(Flatten())
model.add(Dense(500, activation='relu')) # HIDDEN LAYER
model.add(Dropout(0.5))
model.add(Dense(noOfClasses, activation='softmax')) # OUTPUT LAYER
```

Set parameters to train the model: use the compile function to set parameters, train and save the model.

```
# COMPILER MODEL
model.compile(Adam(learning_rate=0.01,momentum=0.9), loss='categorical_crossentropy', metrics=['accuracy'])
return model

#Train model
model = myModel()
print(model.summary())
history = model.fit(dataGen.flow(X_train, y_train, batch_size=50),
                    epochs=200, validation_data=(X_validation, y_validation), shuffle=1))

# Lưu model vào file .h5 để sử dụng cho real-time
model.save('/content/drive/MyDrive/data/model.h5')
cv2.waitKey(0)
```

↩ -1

c) Program to run real-time recognition

```
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 # là mức để máy xác định sự tự tin
font = cv2.FONT_HERSHEY_SIMPLEX
```

Import required library and use GPU command and set threshold to 0.75

```
# 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 the JavaScript object into an OpenCV image

```
# function to convert OpenCV Rectangle bounding box image into base64 byte string to be overlaid 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
```

Function to convert OpenCV Rectangle bounding box image into base64 byte string to be overlaid on video stream

```
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):# xử lý hình ảnh để đưa vào model dự đoán
    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() {
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);

```

JavaScript to properly create our live video stream using our webcam as input

```

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 thú
model_file_path = "/content/drive/MyDrive/data/model.h5"
model5 = load_model(model_file_path)
classes = ['Butterfly','Cat','Chicken','Cow','Dog','Elephant','Horse','Sheep','Spider','Squirrel','Hamster','Eagle','Goldfish','Hedgehog','Turtle']
```

- Command start streaming video from webcam
- Label for video
- Initialize bounding box to empty and load model to recognize pets


```

▶ 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=(32, 32))
    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

```

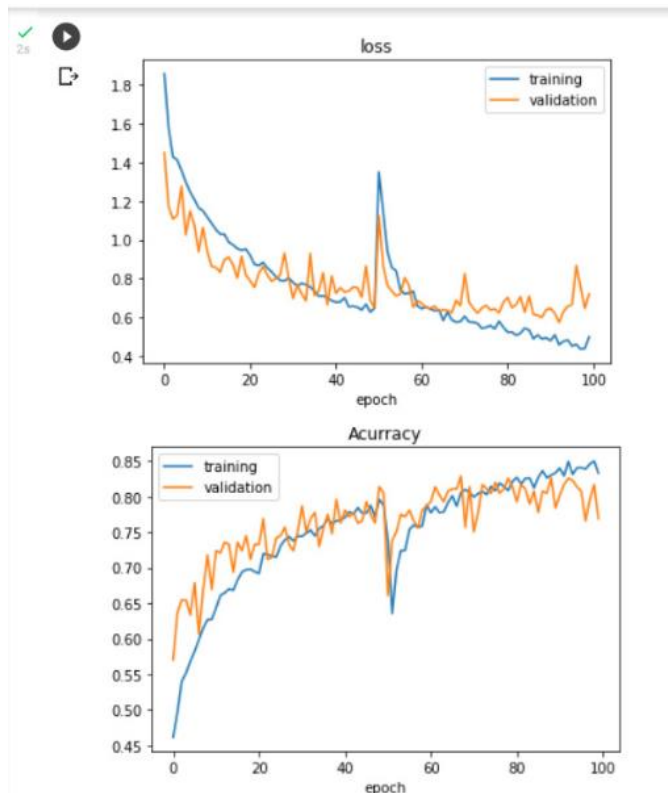
Conduct prediction results, prediction results will be displayed directly on the left top corner of webcam.

d) Model 's accuracy

```

▶ # 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 Accuracy:', score[1])

```



The accuracy of the model 0.8, this result is quite low, so the prediction can lead to many wrong results.

e) Result

The accuracy rate of the model is still low, so in the process of predicting the end is too much wrong. This error can be partly from the data source is quite small, making the learning process of the machine not enough to accurately detect the animals.

- Some of the predictions are correct:



Status Đang lấy hình ảnh...

Butterfly



Bấm vào video để dừng

Status Đang lấy hình ảnh...

Horse



Bấm vào video để dừng

Status Đang lấy hình ảnh...

Dog



Bấm vào video để dừng

Status Đang lấy hình ảnh...

Chicken



Bấm vào video để dừng

Status Đang lấy hình ảnh...

Sheep



Bấm vào video để dừng

Status Đang lấy hình ảnh...

Cat



Bấm vào video để dừng

Status Đang lấy hình ảnh...

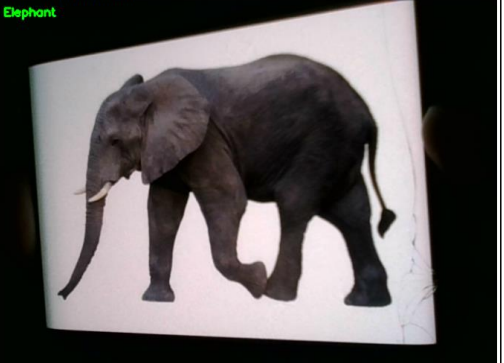
Spider



Bấm vào video để dừng

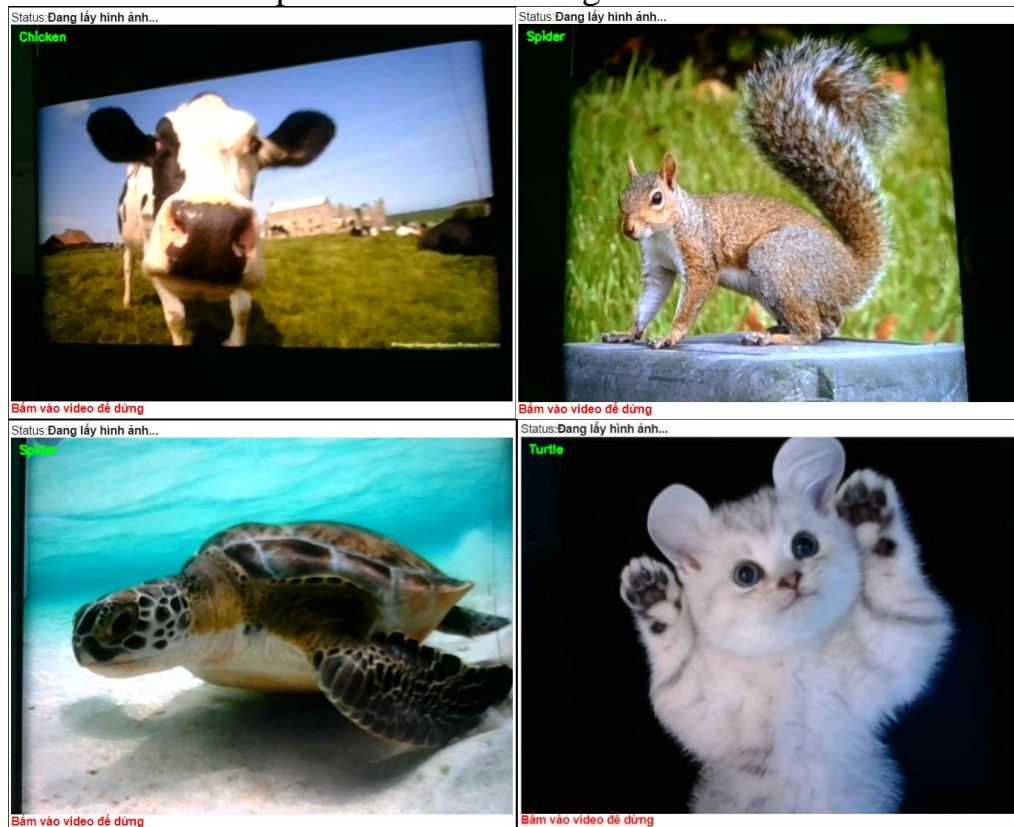
Status Đang lấy hình ảnh...

Elephant



Bấm vào video để dừng

- Some of the predictions are wrong :



f) Web App

```
[61] import gradio as gr
import tensorflow as tf
import numpy as np
import cv2
```

```
Labels = { 'Butterfly':0,
           'Cat':1,
           'Chicken':2,
           'Cow':3,
           'Dog':4,
           'Elephant':5,
           'Horse':6,
           'Sheep':7,
           'Spider':8,
           'Squirrel':9,
           'Hamster':10,
           'Eagle':11,
           'GoldFish':12,
           'Hedgehog':13,
           'Turtle':14,}
```

- Import necessary library and use gradio to make web app
- make label

```

✓ [63] from keras.models import load_model
0      model15= load_model('/content/drive/MyDrive/data/model.h5')
giây

```

- Load model

```

def predict_image(img):
    img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img=cv2.resize(img,(32,32))
    img = np.array(img)/255.0
    img =img.reshape(1,32,32,1)

    predict = np.argmax(model15.predict(img))

    if predict==0:
        return('Butterfly')
    elif predict==1:
        return('Cat')
    elif predict==2:
        return('Chicken')
    elif predict==3:
        return('Cow')
    elif predict==4:
        return('Dog')
    elif predict==5:
        return('Elephant')
    elif predict==6:
        return('Horse')
    elif predict==7:
        return('Sheep')
    elif predict==8:
        return('Spider')
    elif predict==9:
        return('Squirrel')
    elif predict==10:
        return('Hamster')
    elif predict==11:
        return('Eagle')
    elif predict==12:
        return('GoldFish')

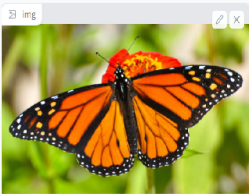
```

```

image = gr.inputs.Image()
label = gr.outputs.Label(num_top_classes=15)
gr.Interface(predict_image, inputs=image, outputs="text",interpretation='default').launch(debug=True)

```

/usr/local/lib/python3.7/dist-packages/gradio/deprecation.py:40: UserWarning: 'optional' parameter is deprecated, and it has no effect
 warnings.warn(value)
 /usr/local/lib/python3.7/dist-packages/gradio/deprecation.py:40: UserWarning: The 'type' parameter has been deprecated. Use the Number component instead.
 warnings.warn(value)
 Colab notebook detected. This cell will run indefinitely so that you can see errors and logs. To turn off, set debug=False in launch().
 Running on public URL: <https://42436.gradio.app>
 This share link expires in 72 hours. For free permanent hosting, check out Spaces (<https://huggingface.co/spaces>)



output

Butterfly

Flag Interpret

Resize image and Test Web App