

Bộ giáo dục và đào tạo
Trường Đại học Ngoại ngữ - Tin học TP.HCM



Đề tài:

Phân lớp hình ảnh với Neutral Network

GVHD: Th.s Vũ Đình Ái

Sinh viên thực hiện : 22DH113672 – Nguyễn Đình Thượng
22DH114572 – Bành Vĩnh Khang

Ngày 04 Tháng 11 Năm 2024

Mục lục

I. Giới thiệu.....	1
II. Cơ sở lý thuyết.....	1
II.1. Thuật toán lan truyền ngược (Backpropagation).....	1
II.1.1 Tóm tắt về thuật toán	1
II.1.2 Nguồn gốc của thuật toán.....	1
II.1.3 Các nghiên cứu liên quan	2
II.1.4 Ưu và khuyết điểm của thuật toán	2
II.1.5 Các bước thực hiện	3
III. Tập dữ liệu	4
IV. Phương pháp tiếp cận	4
IV.1. Thông tin về dữ liệu.....	4
IV.2. Cắt dữ liệu	5
IV.2.1 Cắt dữ liệu train.....	5
IV.2.2 Cắt dữ liệu test	6
IV.3. Lấy tên folder của hình ảnh.....	7
IV.4. Tạo file csv cho dữ liệu train.....	8
IV.5. Trực quan hóa dữ liệu.....	9
IV.6. Cắt hình ảnh về chung một kích thước.....	11
IV.7. Chuẩn hóa hình ảnh thành các mảng	11
IV.8. Mã hóa nhãn	12
IV.9. Xây dựng mô hình	13
V. Xây dựng demo.....	19
VI. Kết luận	20
Tài liệu tham khảo (IEE).....	22

I. Giới thiệu

Chó và mèo là hai loại động vật được yêu thích trên thế giới. Vì vậy việc giúp cho máy tính phân biệt biệt chó mèo một cách tự động đã trở thành ứng dụng quan trọng cho máy học. Giúp máy tính có thể dễ dàng phân biệt chó mèo giúp cải thiện các dịch vụ chăm sóc thú cưng, quản lý các loài động vật dễ dàng hơn và ứng dụng cho các ngành công nghiệp thú y. Dựa trên các hình dạng màu sắc của chó và mèo, mô hình phân loại với mạng nơ ron nhân tạo nhằm thiết kế tự động hóa quá trình phân loại này.

II. Cơ sở lý thuyết

II.1. Thuật toán lan truyền ngược (Backpropagation)

II.1.1 Tóm tắt về thuật toán

Backpropagation là thuật toán cơ bản trong học sâu, một phương pháp tính đạo hàm ngược được áp dụng trong mạng nơ ron nhân tạo. Đây là quá trình học có giám sát trong đó mạng nơ-ron điều chỉnh các trọng số dựa trên sai số giữa dự đoán của nó và giá trị thực tế

Giải thuật học Backpropagation được sử dụng để học các trọng số của một mạng nơ-ron nhiều tầng

Cấu trúc mạng cố định (các nơ-ron và các liên kết giữa chúng là cố định)

Đối với mỗi nơ-ron, hàm tác động phải có đạo hàm liên tục

Giải thuật Backpropagation áp dụng chiến lược gradient descent trong quy tắc cập nhật các trọng số

Để cực tiểu hóa lỗi (khác biệt) giữa các giá trị đầu ra thực tế và các giá trị đầu ra mong muốn, đối với các ví dụ học

II.1.2 Nguồn gốc của thuật toán

Ý tưởng đầu tiên về thuật toán Backpropagation được đề xuất bởi Paul Werbos vào năm 1974 trong bài báo "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences". Tuy nhiên ý tưởng này chưa được biết đến rộng rãi cho đến khi nó được các tác giả David Rumelhart, Geoffrey Hinton và Ronald Williams phát triển và áp dụng vào mạng nơ-ron nhân tạo trong bài báo "Learning representations by back-propagating errors" vào năm 1986. Bài báo này đã giúp cho thuật toán Backpropagation trở thành một phương pháp quan trọng để huấn luyện các mô hình mạng nơ-ron nhân tạo.

II.1.3 Các nghiên cứu liên quan

Công trình cơ bản về lan truyền ngược:

LeCun et al. (1998): Bài báo "Gradient-based learning applied to document recognition" đã áp dụng lan truyền ngược trong mạng nơ-ron tích chập (convolutional neural networks - CNN) cho nhận diện văn bản. [1]

Kingma and Ba (2015): Bài viết "Adam: A method for stochastic optimization" giới thiệu phương pháp tối ưu Adam, cải thiện tốc độ và hiệu suất của lan truyền ngược trong huấn luyện mạng nơ-ron. [2]

Pascanu et al. (2013): Nghiên cứu "On the difficulty of training recurrent neural networks" đã chỉ ra những thách thức trong việc áp dụng lan truyền ngược cho mạng nơ-ron hồi tiếp (recurrent neural networks - RNN) và đề xuất các phương pháp giải quyết. [3]

II.1.4 Ưu và khuyết điểm của thuật toán

- Ưu Điểm:

- **Hiệu quả trong huấn luyện:** Lan truyền ngược cho phép tính toán nhanh chóng và hiệu quả gradient của hàm mất mát đối với trọng số của mạng nơ-ron, từ đó cập nhật trọng số một cách hiệu quả.
- **Khả năng học phi tuyến:** Thuật toán này có khả năng tối ưu hóa các mạng nơ-ron sâu, giúp nó học các mối quan hệ phi tuyến phức tạp trong dữ liệu.
- **Tính tổng quát:** Lan truyền ngược có thể được áp dụng cho nhiều loại mạng nơ-ron khác nhau, bao gồm mạng nơ-ron truyền thống, mạng nơ-ron tích chập (CNN) và mạng nơ-ron hồi tiếp (RNN).
- **Kết quả tốt:** Khi được kết hợp với các kỹ thuật như điều chỉnh học (learning rate), tối ưu hóa như Adam, và regularization, thuật toán lan truyền ngược thường mang lại kết quả tốt trong các bài toán thực tế.

- Khuyết điểm:

- **Tốn tài nguyên:** Lan truyền ngược yêu cầu tính toán lớn, đặc biệt đối với mạng nơ-ron sâu với nhiều tham số, dẫn đến tốn kém về tài nguyên tính toán (CPU/GPU).
- **Vấn đề gradient vanishing/exploding:** Trong các mạng nơ-ron sâu, khi gradient được lan truyền ngược, chúng có thể trở nên rất nhỏ

(vanishing) hoặc rất lớn (exploding), làm cho việc tối ưu hóa gặp khó khăn.

- **Để bị overfitting:** Mạng nơ-ron có thể học quá mức các dữ liệu huấn luyện, dẫn đến hiệu suất kém khi gặp dữ liệu mới. Cần phải sử dụng các kỹ thuật như dropout hoặc regularization để giảm thiểu vấn đề này.
- **Cần dữ liệu lớn:** Thuật toán này thường cần một lượng lớn dữ liệu huấn luyện để đạt được hiệu suất tốt, đặc biệt đối với các bài toán phức tạp.
- **Chọn lựa kiến trúc:** Hiệu suất của thuật toán phụ thuộc vào việc lựa chọn cấu trúc mạng phù hợp. Việc tìm kiếm và tối ưu kiến trúc mạng có thể mất nhiều thời gian và công sức.

II.1.5 Các bước thực hiện

B1: Khởi tạo trọng số cho các nơ-ron

B2: Khởi tạo epoch và batchsize

B3: Lặp qua mỗi epoch:

B3.1: Lặp qua mẫu dữ liệu theo batch_size:

B3.1.2: Lan truyền tiến

B3.1.3: Đối với mỗi lớp nơ-ron tính toán giá trị đầu ra

$$Z_1 = \sum w * x + b$$

Và khởi tạo hàm kích hoạt

$$\hat{y} = g(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

B3.1.4: Cập nhật đầu vào cho hàm tiếp theo

$$Z_2 = W \cdot Z_1 + b$$

B3.2: Tính toán hàm lỗi

$$L = \frac{1}{2N} \sum (y - \hat{y})^2$$

B3.3: Truyền ngược

B3.3.1: Tính gradient của lỗi theo đầu ra của lớp:

$$\text{gradient_output} = \text{error} * \text{activation_derivative}(\text{predicted_output})$$

B3.3.2: Tính gradient của lỗi theo trọng số:

$$\text{gradient_weights} = \text{learning_rate} * \text{gradient_output} * \text{inputs}$$

B3.3.3: Cập nhật trọng số:

$$\text{weights} = \text{weights} + \text{gradient_weights}$$

B3.3.4: Cập nhật đầu vào cho lớp trước (để tính gradient cho lớp trước)

III. Tập dữ liệu

Tập dữ liệu gồm 10000 hình ảnh về chó và mèo. Trong đó có 8000 tấm ảnh để training và 2000 tấm ảnh để test. Tập dữ liệu được thu thập thông qua trang [Kaggle](#). Dưới đây là thông tin về dữ liệu

IV. Phương pháp tiếp cận

Dữ liệu được thu thập tại Kaggle sau đó tiến thành cắt nhỏ dữ liệu thành 2000 ảnh train và 400 ảnh test từ 2 folder training_set và test_set (Quá trình này sẽ được trình bày ở bên dưới). Tiếp đến sẽ áp dụng thuật toán lan truyền ngược và sử dụng hàm softmax để xây dựng mô hình. Sau khi mô hình được xây dựng sẽ tiến hành đánh giá thông qua độ đo accuracy.

IV.1. Thông tin về dữ liệu

Đọc dữ liệu từ các folder:

```
from google.colab import drive
drive.mount('/content/drive')
import os
import shutil
import pandas as pd
import matplotlib.pyplot as plt
import cv2
from google.colab.patches import cv2_imshow
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
import keras
import tensorflow as tf

def count_files_in_folder(folder_path):
    try:
        # Lấy danh sách tất cả các file và thư mục trong folder
        items = os.listdir(folder_path)

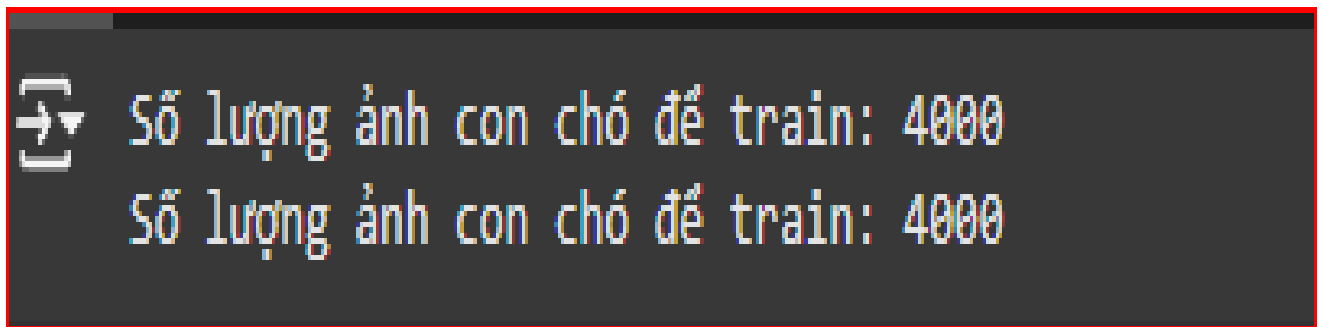
        # Đếm số lượng file
        file_count = sum(1 for item in items if
os.path.isfile(os.path.join(folder_path, item)))

        return file_count
    except Exception as e:
        print(f"Đã xảy ra lỗi: {e}")
        return None

# Đường dẫn đến folder bạn muốn kiểm tra
folder_path =
"/content/drive/MyDrive/MachineLearning/Dog_Cat/training_set/cats"
file_count = count_files_in_folder(folder_path)
```

```
if file_count is not None:
    print(f"Số lượng file trong folder: {file_count}")
```

- Kết quả



Hình 1 kết quả 1

IV.2. Cắt dữ liệu

IV.2.1 Cắt dữ liệu train

- Cắt dữ liệu từ 8000 hình ảnh con chó và mèo thành 1000 hình ảnh cho con chó và 1000 hình ảnh cho con mèo

```
def create_and_copy_images(source_folder, destination_folder,
limit=1000):
    # Tạo thư mục đích nếu chưa tồn tại
    os.makedirs(destination_folder, exist_ok=True)

    # Lấy danh sách các file trong thư mục nguồn
    files = [f for f in os.listdir(source_folder) if
os.path.isfile(os.path.join(source_folder, f))]

    # Chỉ lấy số lượng file giới hạn
    for file in files[:limit]:
        source_path = os.path.join(source_folder, file)
        destination_path = os.path.join(destination_folder, file)

        # Sao chép file từ thư mục nguồn sang thư mục đích
        shutil.copy(source_path, destination_path)

    # Đường dẫn đến các folder
    training_set_folder =
"/content/drive/MyDrive/MachineLearning/Dog_Cat/training_set"
    training_folder =
"/content/drive/MyDrive/MachineLearning/Dog_Cat/Training"

    # Tạo folder training với các folder con
    cats_destination_folder = os.path.join(training_folder , "cats")
    dogs_destination_folder = os.path.join(training_folder , "dogs")

    # Tạo folder training nếu chưa tồn tại
    os.makedirs(training_folder , exist_ok=True)
```

```
# Sao chép 200 bức ảnh từ folder cats
create_and_copy_images(os.path.join(training_set_folder, "cats"),
cats_destination_folder, limit=1000)

# Sao chép 200 bức ảnh từ folder dogs
create_and_copy_images(os.path.join(training_set_folder, "dogs"),
dogs_destination_folder, limit=1000)

def list_files_in_folder(folder_path):
    # Lấy danh sách các file trong thư mục
    return [f for f in os.listdir(folder_path) if
os.path.isfile(os.path.join(folder_path, f))]

# Đường dẫn đến các thư mục con
cats_destination_folder =
"/content/drive/MyDrive/MachineLearning/Dog_Cat/Training/cats"
dogs_destination_folder =
"/content/drive/MyDrive/MachineLearning/Dog_Cat/Training/dogs"

# Hiển thị danh sách các file trong thư mục "cats"
cats_files = list_files_in_folder(cats_destination_folder)
print("Số lượng ảnh của con mèo sau khi cắt", len(cats_files))

# Hiển thị danh sách các file trong thư mục "dogs"
dogs_files = list_files_in_folder(dogs_destination_folder)
print("Số lượng ảnh của con chó sau khi cắt", len(dogs_files))
```

- Hiển thị kết quả sau khi cắt:

```
→ Số lượng ảnh của con mèo sau khi cắt 1000
   Số lượng ảnh của con chó sau khi cắt 1000
```

Hình 2. Kết quả 2

IV.2.2 Cắt dữ liệu test

- Cắt dữ liệu từ 2000 hình ảnh chó và mèo thành 400 hình ảnh chó và mèo để test

```
def create_and_copy_images(source_folder, destination_folder,
limit=200):
    # Tạo thư mục đích nếu chưa tồn tại
    os.makedirs(destination_folder, exist_ok=True)

    # Lấy danh sách các file trong thư mục nguồn
    files = [f for f in os.listdir(source_folder) if
os.path.isfile(os.path.join(source_folder, f))]

    # Chỉ lấy số lượng file giới hạn
    for file in files[:limit]:
        source_path = os.path.join(source_folder, file)
```



```
destination_path = os.path.join(destination_folder, file)

# Sao chép file từ thư mục nguồn sang thư mục đích
shutil.copy(source_path, destination_path)

# Đường dẫn đến các folder
training_set_folder =
"""//content/drive/MyDrive/MachineLearning/Dog_Cat/test_set"""
training_folder = "/content/drive/MyDrive/MachineLearning/Dog_Cat/Test"

# Tạo folder training với các folder con
cats_destination_folder = os.path.join(training_folder, "cats")
dogs_destination_folder = os.path.join(training_folder, "dogs")

# Tạo folder training nếu chưa tồn tại
os.makedirs(training_folder, exist_ok=True)

# Sao chép 200 bức ảnh từ folder cats
create_and_copy_images(os.path.join(training_set_folder, "cats"),
cats_destination_folder, limit=200)

# Sao chép 200 bức ảnh từ folder dogs
create_and_copy_images(os.path.join(training_set_folder, "dogs"),
dogs_destination_folder, limit=200)


def list_files_in_folder(folder_path):
    # Lấy danh sách các file trong thư mục
    return [f for f in os.listdir(folder_path) if
os.path.isfile(os.path.join(folder_path, f))]

# Đường dẫn đến các thư mục con
cats_test = "/content/drive/MyDrive/MachineLearning/Dog_Cat/Test/cats"
dogs_test = "/content/drive/MyDrive/MachineLearning/Dog_Cat/Test/dogs"

# Hiển thị danh sách các file trong thư mục "cats"
cats_test_file = list_files_in_folder(cats_test)
print("Số lượng ảnh của con mèo sau khi cắt", len(cats_test_file))

# Hiển thị danh sách các file trong thư mục "dogs"
dogs_test_file = list_files_in_folder(dogs_test)
print("Số lượng ảnh của con chó sau khi cắt", len(dogs_test_file))
```

- Kết quả

 Số lượng ảnh của con mèo sau khi cắt 200
Số lượng ảnh của con chó sau khi cắt 200

Hình 3 kết quả 3

IV.3. Lấy tên folder của hình ảnh

- Lấy tên folder để làm label cho tập dữ liệu

```
base_path = '/content/drive/MyDrive/MachineLearning/Dog_Cat/Training'
types = os.listdir(base_path)
```

```
print(types)
```

- Kết quả

➡ ['cats', 'dogs']

Hình 4 kết quả 4

IV.4. Tạo file csv cho dữ liệu train

- Tạo file csv cho folder hình ảnh train

```
import os
import pandas as pd

# Đường dẫn đến thư mục chứa ảnh
base_path = '/content/drive/MyDrive/MachineLearning/Dog_Cat/Training'
types = os.listdir(base_path)

# Danh sách để lưu thông tin ảnh
dt = []

# Duyệt qua từng loại ảnh (cats và dogs)
for item in types:
    # Lấy tất cả file ảnh trong thư mục
    all_type = os.listdir(os.path.join(base_path, item))
    for img in all_type:
        # Tạo đường dẫn đầy đủ cho ảnh
        img_path = os.path.join(base_path, item, img)

        # Xác định label: 0 cho cats và 1 cho dogs
        label = 0 if item == 'cats' else 1

        # Thêm tuple (img_path, label, item) vào danh sách
        dt.append((img_path, label, item))

# Tạo DataFrame từ danh sách dt
df = pd.DataFrame(dt, columns=['image_path', 'label', 'type'])

# Đường dẫn tới file CSV mà bạn muốn lưu
csv_file_path =
'/content/drive/MyDrive/MachineLearning/Dog_Cat/image_labels.csv'

# Ghi DataFrame vào file CSV
df.to_csv(csv_file_path, index=False)


print("File CSV đã được tạo thành công tại:", csv_file_path)
```

- Đọc file csv

```
df =
```

```
pd.read_csv('/content/drive/MyDrive/MachineLearning/Dog_Cat/image_labels.csv')
```

- Kết quả:



	image_path	label	type
0	/content/drive/MyDrive/MachineLearning/Dog_Cat...	0	cats
1	/content/drive/MyDrive/MachineLearning/Dog_Cat...	0	cats
2	/content/drive/MyDrive/MachineLearning/Dog_Cat...	0	cats
3	/content/drive/MyDrive/MachineLearning/Dog_Cat...	0	cats
4	/content/drive/MyDrive/MachineLearning/Dog_Cat...	0	cats
...
1995	/content/drive/MyDrive/MachineLearning/Dog_Cat...	1	dogs
1996	/content/drive/MyDrive/MachineLearning/Dog_Cat...	1	dogs
1997	/content/drive/MyDrive/MachineLearning/Dog_Cat...	1	dogs
1998	/content/drive/MyDrive/MachineLearning/Dog_Cat...	1	dogs
1999	/content/drive/MyDrive/MachineLearning/Dog_Cat...	1	dogs

2000 rows x 3 columns

Hình 5 kết quả 5

IV.5. Trực quan hóa dữ liệu

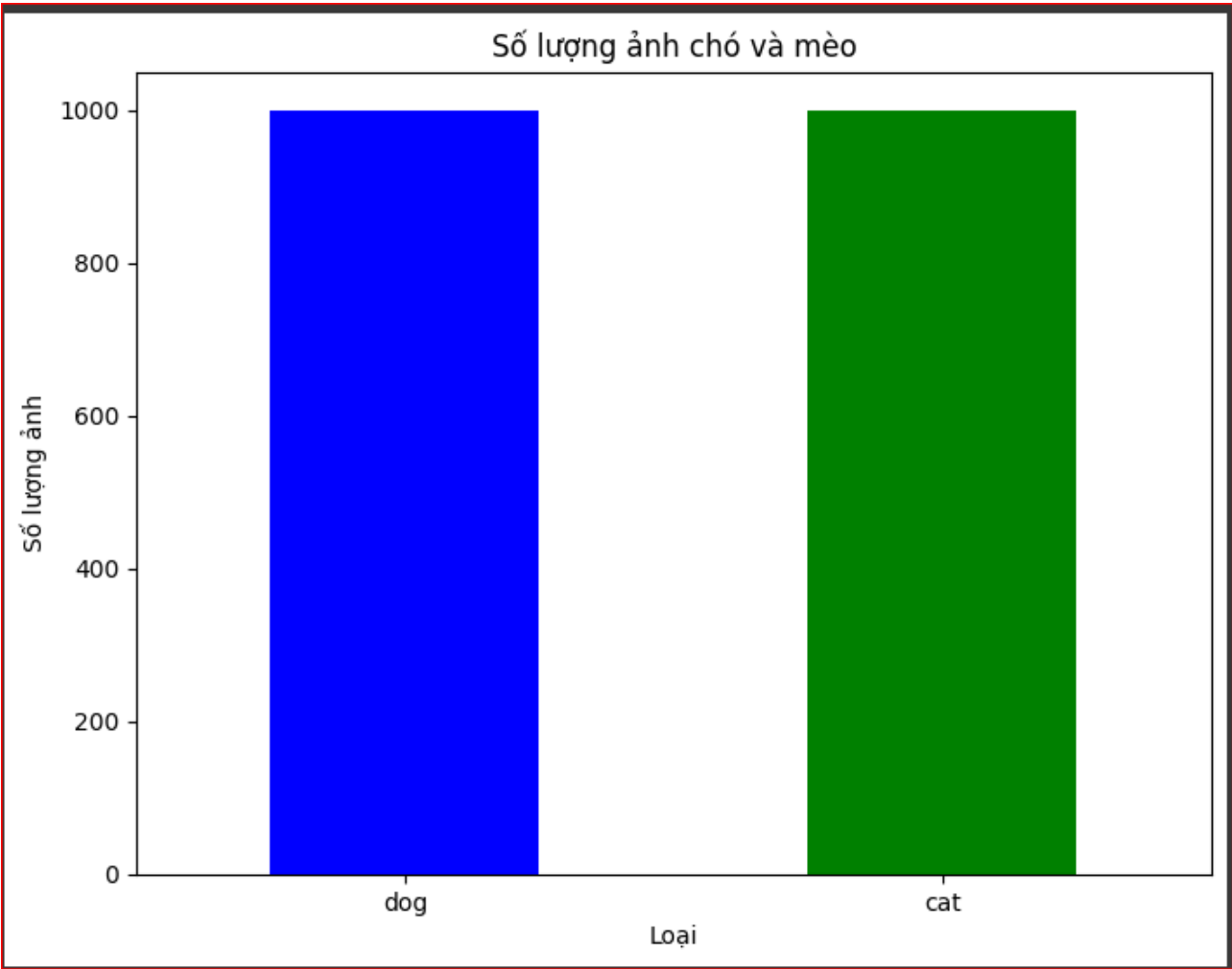
Trực quan dữ liệu theo biểu đồ bar

```
# Đổi nhãn từ số sang tên loại
label_map = {0: 'dog', 1: 'cat'}
df['Type'] = df['label'].map(label_map)

# Đếm số lượng ảnh cho mỗi loại
type_counts = df['Type'].value_counts()

# Trực quan hóa kết quả bằng biểu đồ thanh
plt.figure(figsize=(8, 6))
type_counts.plot(kind='bar', color=['blue', 'green'])
plt.title("Số lượng ảnh chó và mèo")
plt.xlabel("Loại")
plt.ylabel("Số lượng ảnh")
plt.xticks(rotation=0)
plt.show()
```

- Kết quả:



IV.6. Cắt hình ảnh về chung một kích thước

```
# Hàm resize_images_in_folder nhận vào thư mục gốc và kích thước mong muốn
def resize_images_in_folder(root_folder, imsize=64):
    images = []
    labels = []
    # Lấy tất cả thư mục con trong thư mục gốc
    types = os.listdir(root_folder)

    for folder in types:
        folder_path = os.path.join(root_folder, folder)
        # Kiểm tra nếu folder_path là một thư mục
        if os.path.isdir(folder_path):
            filenames = os.listdir(folder_path)
            for f in filenames:
                img_path = os.path.join(folder_path, f)
                img = cv2.imread(img_path)
                if img is not None:
                    img = cv2.resize(img, (imsize, imsize))
                    images.append(img)
                    labels.append(folder)
            else:
                print(f"Failed to load image: {img_path}")

    return images, labels

# Sử dụng hàm resize_images_in_folder với đường dẫn thư mục
resize_image_train =
'/content/drive/MyDrive/MachineLearning/Dog_Cat/Training'
x_train,y_train = resize_images_in_folder(resize_image_train)

resize_image_test = '/content/drive/MyDrive/MachineLearning/Dog_Cat/Test'
x_test,y_test = resize_images_in_folder(resize_image_test)
```

IV.7. Chuẩn hóa hình ảnh thành các mảng

```
# Hàm resize_images_in_folder nhận vào thư mục gốc và kích thước mong muốn
def resize_images_in_folder(root_folder, imsize=64):
    images = []
    labels = []
    # Lấy tất cả thư mục con trong thư mục gốc
    types = os.listdir(root_folder)

    for folder in types:
        folder_path = os.path.join(root_folder, folder)
        # Kiểm tra nếu folder_path là một thư mục
        if os.path.isdir(folder_path):
            filenames = os.listdir(folder_path)
            for f in filenames:
                img_path = os.path.join(folder_path, f)
```

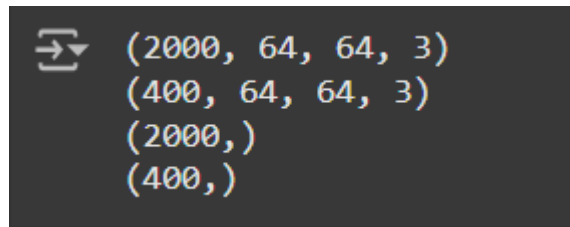
```
img = cv2.imread(img_path)
if img is not None:
    img = cv2.resize(img, (imsized, imsize))
    images.append(img)
    labels.append(folder)
else:
    print(f"Failed to load image: {img_path}")

return images, labels

# Sử dụng hàm resize_images_in_folder với đường dẫn thư mục
resize_image_train =
'/content/drive/MyDrive/MachineLearning/Dog_Cat/Training'
x_train,y_train = resize_images_in_folder(resize_image_train)

resize_image_test = '/content/drive/MyDrive/MachineLearning/Dog_Cat/Test'
x_test,y_test = resize_images_in_folder(resize_image_test)
```

- Kết quả:



```
(2000, 64, 64, 3)
(400, 64, 64, 3)
(2000,)
(400,)
```

Hình 6 kết quả 6

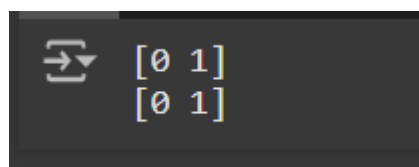
IV.8. Mã hóa nhãn

```
# Khởi tạo LabelEncoder
y_labelencoder = LabelEncoder()

# Mã hóa nhãn cho y_train và y_test
y_train = y_labelencoder.fit_transform(y_train)
y_test= y_labelencoder.transform(y_test)

# Kiểm tra nhãn sau khi mã hóa
print(np.unique(y_train))
print(np.unique(y_test))
```

- Kết quả:



```
[0 1]
[0 1]
```

Hình 7 kết quả 7

IV.9. Xây dựng mô hình

- Xây dựng mô hình bằng tensorflow

```
model = keras.models.Sequential([
    # Flatten layer to reshape input
    keras.layers.Flatten(input_shape=(64, 64, 3)),
    # First Dense Layer with BatchNorm and Dropout
    keras.layers.Dense(512, activation=tf.nn.relu),
    # Second Dense Layer with BatchNorm and Dropout
    keras.layers.Dense(256, activation=tf.nn.relu),
    # Third Dense Layer with BatchNorm and Dropout
    keras.layers.Dense(128, activation=tf.nn.relu),
    # Output Layer
    keras.layers.Dense(2, activation='softmax')
])
```

- Tóm tắt mô hình:

```
model.summary()
```

- Kết quả:

Model: "sequential_6"

Layer (type)	Output Shape	Param #
flatten_6 (Flatten)	(None, 12288)	0
dense_24 (Dense)	(None, 512)	6,291,968
dense_25 (Dense)	(None, 256)	131,328
dense_26 (Dense)	(None, 128)	32,896
dense_27 (Dense)	(None, 2)	258

Total params: 6,456,450 (24.63 MB)
 Trainable params: 6,456,450 (24.63 MB)
 Non-trainable params: 0 (0.00 B)

Hình 8 kết quả 8

- Biên dịch mô hình:

```
model.compile(optimizer=keras.optimizers.Adam(learning_rate=0.000001),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

- Huấn luyện mô hình:

```
history = model.fit(x_train, y_train, epochs=100)
```

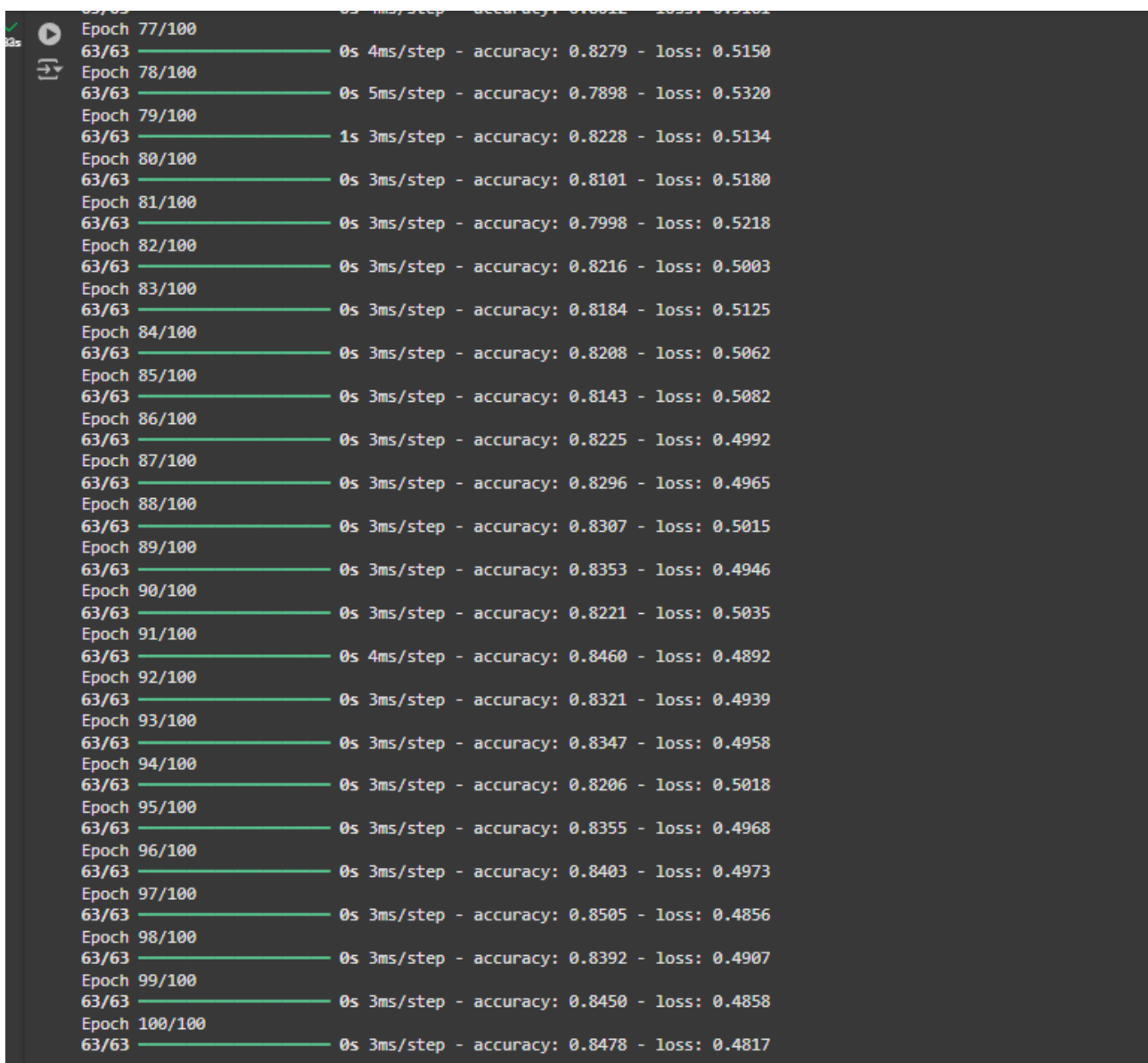
- Kết quả:

```

Epoch 1/100
63/63 ————— 4s 45ms/step - accuracy: 0.4941 - loss: 0.7141
Epoch 2/100
63/63 ————— 0s 3ms/step - accuracy: 0.5479 - loss: 0.6893
Epoch 3/100
63/63 ————— 1s 7ms/step - accuracy: 0.5564 - loss: 0.6847
Epoch 4/100
63/63 ————— 0s 3ms/step - accuracy: 0.5876 - loss: 0.6698
Epoch 5/100
63/63 ————— 0s 3ms/step - accuracy: 0.5927 - loss: 0.6676
Epoch 6/100
63/63 ————— 0s 3ms/step - accuracy: 0.6221 - loss: 0.6632
Epoch 7/100
63/63 ————— 0s 3ms/step - accuracy: 0.6127 - loss: 0.6610
Epoch 8/100
63/63 ————— 0s 3ms/step - accuracy: 0.6099 - loss: 0.6638
Epoch 9/100
63/63 ————— 0s 3ms/step - accuracy: 0.6270 - loss: 0.6580
Epoch 10/100
63/63 ————— 0s 3ms/step - accuracy: 0.6416 - loss: 0.6462
Epoch 11/100
63/63 ————— 0s 3ms/step - accuracy: 0.6393 - loss: 0.6437
Epoch 12/100
63/63 ————— 0s 3ms/step - accuracy: 0.6563 - loss: 0.6443
Epoch 13/100
63/63 ————— 0s 3ms/step - accuracy: 0.6495 - loss: 0.6355
Epoch 14/100
63/63 ————— 0s 3ms/step - accuracy: 0.6252 - loss: 0.6461
Epoch 15/100
63/63 ————— 0s 3ms/step - accuracy: 0.6709 - loss: 0.6306
Epoch 16/100
63/63 ————— 0s 3ms/step - accuracy: 0.6538 - loss: 0.6369
Epoch 17/100
63/63 ————— 0s 3ms/step - accuracy: 0.6767 - loss: 0.6280
Epoch 18/100
63/63 ————— 0s 3ms/step - accuracy: 0.6692 - loss: 0.6305
Epoch 19/100
63/63 ————— 0s 4ms/step - accuracy: 0.6796 - loss: 0.6246
Epoch 20/100
63/63 ————— 0s 4ms/step - accuracy: 0.6750 - loss: 0.6246
Epoch 21/100
63/63 ————— 0s 4ms/step - accuracy: 0.6727 - loss: 0.6242
Epoch 22/100
63/63 ————— 0s 4ms/step - accuracy: 0.6903 - loss: 0.6221
Epoch 23/100
63/63 ————— 0s 4ms/step - accuracy: 0.6708 - loss: 0.6241
Epoch 24/100
63/63 ————— 0s 4ms/step - accuracy: 0.6867 - loss: 0.6195
Epoch 25/100
63/63 ————— 0s 4ms/step - accuracy: 0.7053 - loss: 0.6084
Epoch 26/100
63/63 ————— 1s 4ms/step - accuracy: 0.7074 - loss: 0.6051
Epoch 27/100
63/63 ————— 0s 5ms/step - accuracy: 0.7067 - loss: 0.6070
Epoch 28/100
63/63 ————— 0s 5ms/step - accuracy: 0.6976 - loss: 0.6078

```

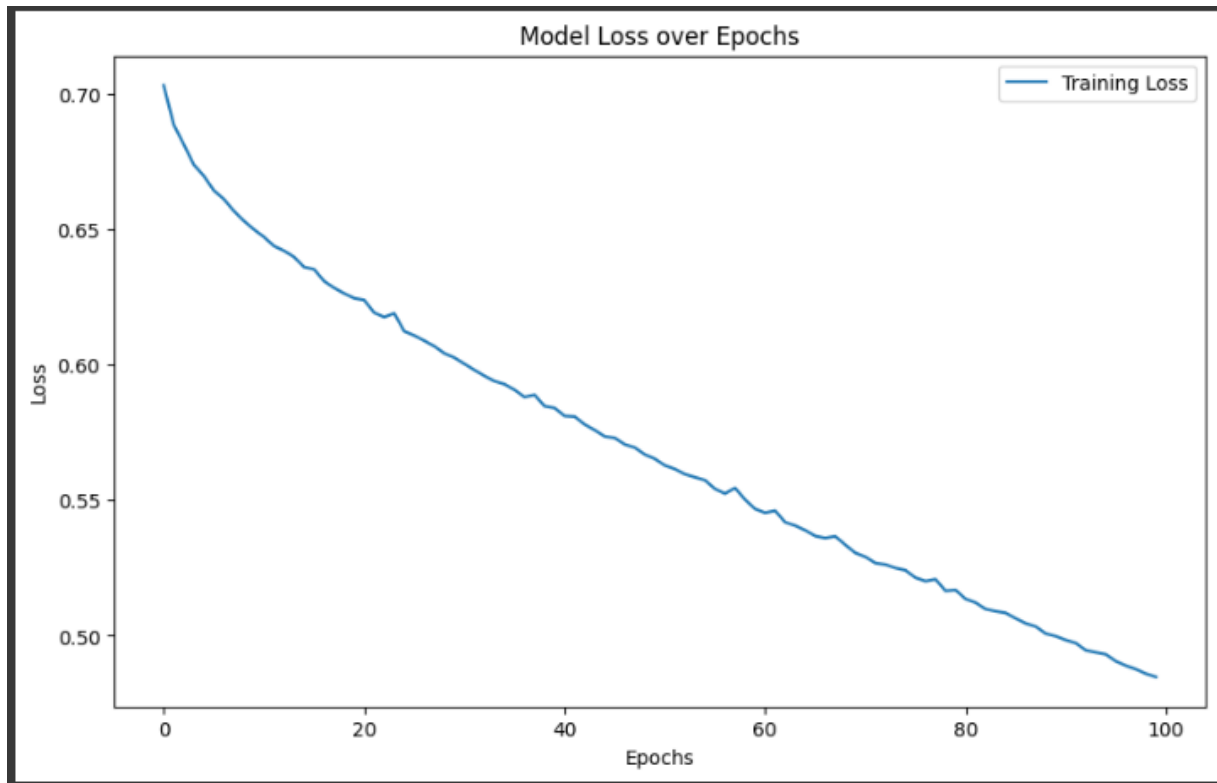

Epoch 29/100	63/63	1s 4ms/step	- accuracy: 0.7027	- loss: 0.6089
Epoch 30/100	63/63	0s 3ms/step	- accuracy: 0.7115	- loss: 0.6058
Epoch 31/100	63/63	0s 3ms/step	- accuracy: 0.7227	- loss: 0.5978
Epoch 32/100	63/63	0s 3ms/step	- accuracy: 0.7204	- loss: 0.5957
Epoch 33/100	63/63	0s 4ms/step	- accuracy: 0.7353	- loss: 0.5932
Epoch 34/100	63/63	0s 3ms/step	- accuracy: 0.7286	- loss: 0.5958
Epoch 35/100	63/63	0s 3ms/step	- accuracy: 0.7241	- loss: 0.5850
Epoch 36/100	63/63	0s 3ms/step	- accuracy: 0.7205	- loss: 0.5950
Epoch 37/100	63/63	0s 3ms/step	- accuracy: 0.7347	- loss: 0.5929
Epoch 38/100	63/63	0s 3ms/step	- accuracy: 0.7248	- loss: 0.5843
Epoch 39/100	63/63	0s 3ms/step	- accuracy: 0.7385	- loss: 0.5836
Epoch 40/100	63/63	0s 3ms/step	- accuracy: 0.7489	- loss: 0.5776
Epoch 41/100	63/63	0s 3ms/step	- accuracy: 0.7299	- loss: 0.5838
Epoch 42/100	63/63	0s 3ms/step	- accuracy: 0.7430	- loss: 0.5791
Epoch 43/100	63/63	0s 3ms/step	- accuracy: 0.7345	- loss: 0.5837
Epoch 44/100	63/63	0s 3ms/step	- accuracy: 0.7496	- loss: 0.5715
Epoch 45/100	63/63	0s 3ms/step	- accuracy: 0.7559	- loss: 0.5644
Epoch 46/100	63/63	0s 3ms/step	- accuracy: 0.7477	- loss: 0.5730
Epoch 47/100	63/63	0s 3ms/step	- accuracy: 0.7389	- loss: 0.5744
Epoch 48/100	63/63	0s 3ms/step	- accuracy: 0.7463	- loss: 0.5708
Epoch 49/100	63/63	0s 3ms/step	- accuracy: 0.7443	- loss: 0.5667
Epoch 50/100	63/63	0s 3ms/step	- accuracy: 0.7516	- loss: 0.5609
Epoch 51/100	63/63	0s 3ms/step	- accuracy: 0.7497	- loss: 0.5712
Epoch 52/100	63/63	0s 3ms/step	- accuracy: 0.7767	- loss: 0.5549
Epoch 53/100	63/63	0s 3ms/step	- accuracy: 0.7652	- loss: 0.5619
Epoch 54/100	63/63	0s 3ms/step	- accuracy: 0.7576	- loss: 0.5631
Epoch 55/100	63/63	0s 3ms/step	- accuracy: 0.7630	- loss: 0.5548
Epoch 56/100	63/63	0s 3ms/step	- accuracy: 0.7497	- loss: 0.5595
Epoch 57/100	63/63	0s 3ms/step	- accuracy: 0.7663	- loss: 0.5487
Epoch 58/100	63/63	0s 3ms/step	- accuracy: 0.7621	- loss: 0.5557
Epoch 59/100	63/63	0s 3ms/step	- accuracy: 0.7784	- loss: 0.5439
Epoch 60/100	63/63	0s 3ms/step	- accuracy: 0.7835	- loss: 0.5460
Epoch 61/100	63/63	0s 3ms/step	- accuracy: 0.7873	- loss: 0.5428
Epoch 62/100	63/63	0s 3ms/step	- accuracy: 0.7867	- loss: 0.5400
Epoch 63/100	63/63	0s 3ms/step	- accuracy: 0.7907	- loss: 0.5416
Epoch 64/100	63/63	0s 3ms/step	- accuracy: 0.7879	- loss: 0.5390
Epoch 65/100	63/63	0s 3ms/step	- accuracy: 0.7997	- loss: 0.5364
Epoch 66/100	63/63	0s 3ms/step	- accuracy: 0.7969	- loss: 0.5356
Epoch 67/100	63/63	0s 4ms/step	- accuracy: 0.7979	- loss: 0.5320
Epoch 68/100	63/63	0s 4ms/step	- accuracy: 0.7906	- loss: 0.5327
Epoch 69/100	63/63	0s 4ms/step	- accuracy: 0.7980	- loss: 0.5261
Epoch 70/100	63/63	0s 4ms/step	- accuracy: 0.8089	- loss: 0.5259
Epoch 71/100	63/63	0s 4ms/step	- accuracy: 0.8069	- loss: 0.5314
Epoch 72/100	63/63	0s 4ms/step	- accuracy: 0.8106	- loss: 0.5252
Epoch 73/100	63/63	0s 4ms/step	- accuracy: 0.8036	- loss: 0.5268
Epoch 74/100	63/63	0s 4ms/step	- accuracy: 0.7885	- loss: 0.5335
Epoch 75/100	63/63	0s 4ms/step	- accuracy: 0.8203	- loss: 0.5171
Epoch 76/100	63/63	0s 4ms/step	- accuracy: 0.8012	- loss: 0.5181



Hình 9 kết quả 9

- Trực quan hàm loss:
- `plt.figure(figsize=(10, 6))`
- `plt.plot(history.history['loss'], label='Training Loss')`
- `plt.title('Model Loss over Epochs')`
- `plt.xlabel('Epochs')`
- `plt.ylabel('Loss')`
- `plt.legend()`
- `plt.show()`

- Kết quả:



Hình 10 kết quả 10

- Đánh giá mô hình bằng accuracy score, classification report, confusion matrix
- ```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
from sklearn.metrics import accuracy_score

predictions = model.predict(x_test)
predicted_classes = np.argmax(predictions, axis=1)

Đảm bảo class_labels là danh sách các chuỗi
class_labels = list(map(str, y_labelencoder.classes_))

accuracy = accuracy_score(y_test, predicted_classes)
print(f"Accuracy on the test set: {accuracy * 100:.2f}%")

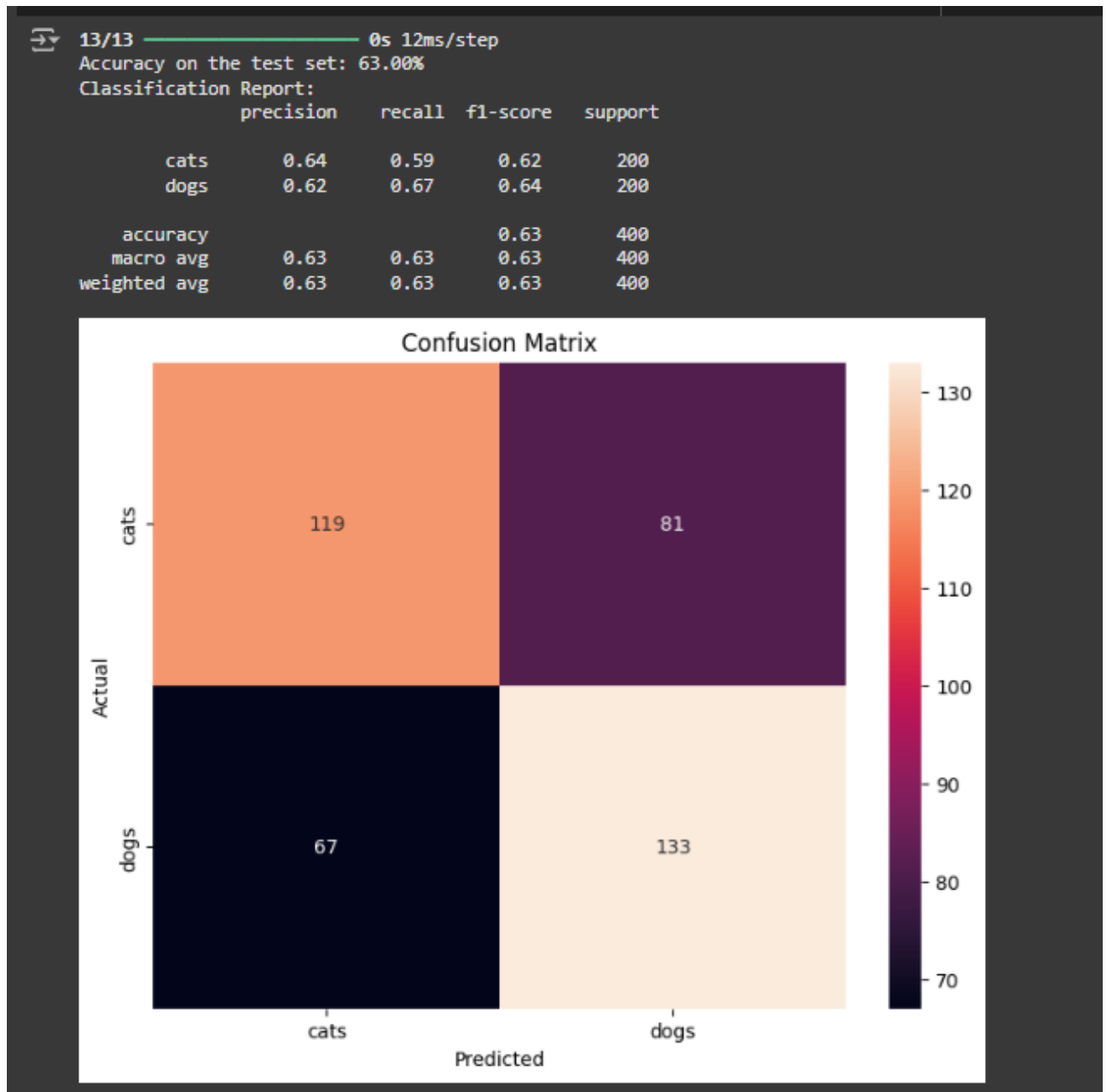
Sử dụng class_labels trong classification_report và heatmap
class_report = classification_report(y_test, predicted_classes,
target_names=class_labels)
print("Classification Report:\n", class_report)

conf_matrix = confusion_matrix(y_test, predicted_classes)

Vẽ heatmap cho ma trận nhầm lẫn
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d',
xticklabels=class_labels, yticklabels=class_labels)
plt.title('Confusion Matrix')
plt.ylabel('Actual')
```

- `plt.xlabel('Predicted')`
- `plt.show()`

- Kết quả:




Hình 11 kết quả 11

## V. Xây dựng demo

- Lưu file model:
- ```
model.save('/content/drive/MyDrive/MachineLearning/WebPage.h5')
```
- Giao diện chương trình:

Phân biệt chó và mèo

Tải hình ảnh lên


 Drag and drop file here
Limit 200MB per file

Browse files

Hình 12. Giao diện chương trình

Phân biệt chó và mèo ⇄

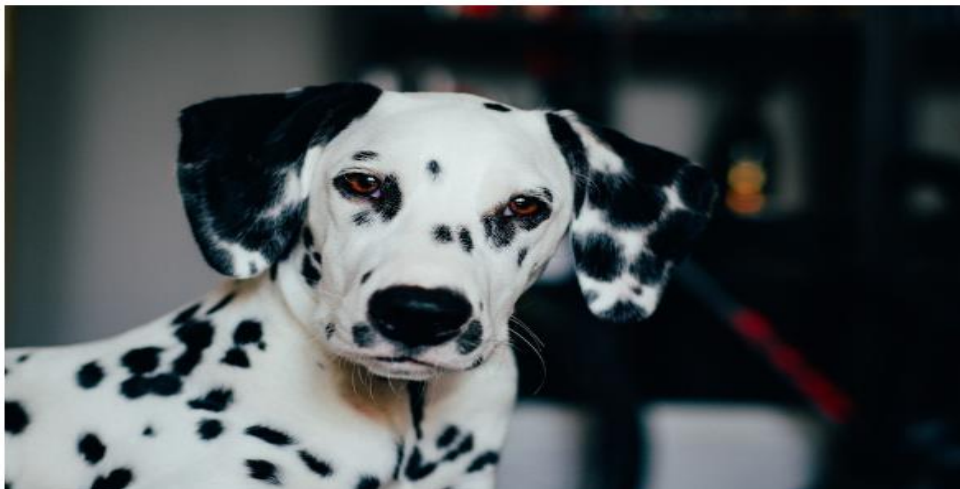
Tải hình ảnh lên

 Drag and drop file here
Limit 200MB per file

Browse files

 dalmatian12.jpg 72.0KB 

The use_column_width parameter has been deprecated and will be removed in a future release. Please utilize the use_container_width parameter instead.



Uploaded Image

Dự đoán

Kết quả dự đoán: dog

Hình 13. Kết quả

VI. Kết luận

Mô hình nghiên cứu và xây dựng phân biệt chó và mèo bằng mạng lan truyền ngược đã mang lại cái nhìn sâu sắc hơn về khả năng ứng dụng của trí tuệ nhân tạo trong lĩnh vực nhận diện hình ảnh. Mô hình được triển khai dựa trên nền tảng mạng nơ-ron nhân tạo (Artificial Neural Network - ANN) với thuật toán lan truyền ngược (Backpropagation), một phương pháp phổ biến trong việc tối ưu hóa trọng số mạng để giảm thiểu sai số dự đoán. Quá trình triển khai đã bao gồm các bước quan trọng như tiền xử lý dữ liệu, thiết kế kiến trúc mạng phù hợp và tinh chỉnh các tham số như số lượng lớp ẩn, số nút trong mỗi lớp, và hàm kích hoạt, đảm bảo mô hình có khả năng học được đặc trưng từ tập dữ liệu huấn luyện.

Kết quả thu được từ mô hình cho thấy độ chính xác đạt được là 60.00%, đây là một bước đầu đáng ghi nhận trong việc xây dựng hệ thống phân loại hình ảnh đơn giản. Ngoài ra, ma trận nhầm lẫn (Confusion Matrix) được xây dựng đã giúp đánh giá chi tiết hơn hiệu suất của mô hình bằng cách chỉ ra tỷ lệ dự đoán đúng và sai trong từng nhãn lớp. Dựa vào ma trận này, nhóm nghiên cứu có thể xác định được các nhược điểm cụ thể như tỷ lệ lỗi cao ở lớp nào hoặc nhãn nào dễ bị nhầm lẫn nhất, từ đó có cơ sở để cải thiện mô hình trong tương lai.

Tuy nhiên, mô hình vẫn tồn tại nhiều hạn chế cần được khắc phục để nâng cao hiệu suất. Độ chính xác hiện tại chưa cao, điều này có thể do các yếu tố như chưa tối ưu hóa các tham số quan trọng bao gồm learning rate, số lượng epoch, và kích thước hình ảnh đầu vào. Hơn nữa, tập dữ liệu huấn luyện hiện tại còn nhỏ và chưa đủ đa dạng, dẫn đến khả năng mô hình bị overfitting hoặc không đủ khả năng tổng quát hóa khi làm việc với dữ liệu kiểm thử. Ngoài ra, việc chỉ sử dụng mạng lan truyền ngược mà không so sánh với các mô hình tiên tiến hơn, chẳng hạn như các mạng học sâu (Deep Learning) như CNN (Convolutional Neural Network), RNN (Recurrent Neural Network), hoặc các mô hình dựa trên học chuyển giao (Transfer Learning), cũng là một hạn chế lớn của nghiên cứu này.

Trong tương lai, nghiên cứu cần được mở rộng để khắc phục các hạn chế trên. Việc cải thiện mô hình có thể bắt đầu bằng cách tinh chỉnh cẩn thận

các tham số, sử dụng tập dữ liệu huấn luyện lớn hơn và đa dạng hơn để cải thiện khả năng học của mô hình. Đồng thời, nghiên cứu cần thử nghiệm và so sánh hiệu suất của mạng lan truyền ngược với các mô hình học sâu hiện đại để đánh giá rõ ràng hơn về hiệu quả của từng phương pháp. Những hướng phát triển này không chỉ giúp nâng cao độ chính xác của mô hình mà còn mở ra tiềm năng ứng dụng rộng rãi hơn trong các bài toán phân loại hình ảnh thực tế.

- [1] Rumelhart, Hinton, and Williams, “Nature,” 1986. [Trực tuyến]. Available: <https://www.nature.com/articles/323533a0>.
- [2] Diederik P. Kingma, Diederik P. Kingma, “A METHOD FOR STOCHASTIC OPTIMIZATION,” trong *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION*, 2015, p. 15.
- [3] On the difficulty of training Recurrent Neural Networks, “On the difficulty of training Recurrent Neural Networks,” trong *On the difficulty of training Recurrent Neural Networks*, 2013.
- [4] A. Géron, “Neural Networks and Deep Learning,” trong *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, O'Reilly Media, Inc., 2019, pp. 363-411.
- [5] N. T. Tuấn, “Deep Learning cơ bản,” 2019. [Trực tuyến]. Available: <https://nttuan8.com/bai-4-backpropagation/>.
- [6] d. thuan, “VILBO,” 24 12 2018. [Trực tuyến]. Available: <https://viblo.asia/p/tim-hieu-ve-thuat-toan-lan-truyen-nguoc-phan-1-GrLZDv8O5k0>.
- [7] S. Nguyen, “YouTube,” 21 4 2019. [Trực tuyến]. Available: <https://www.youtube.com/watch?v=8TBrOj5iAkU&list=PLZEIt444jqBPoqtW2ARJp9ICnF3c7vJC&index=30>.
- [8] S. Nguyen, “YouTube,” 21 4 2019. [Trực tuyến]. Available: <https://www.youtube.com/watch?v=i3ujiwfxGtE&list=PLZEIt444jqBPoqtW2ARJp9ICnF3c7vJC&index=31>.
- [9] S. Nguyen, “YouTube,” 22 4 2019. [Trực tuyến]. Available: <https://www.youtube.com/watch?v=NJWldD-oz0&list=PLZEIt444jqBPoqtW2ARJp9ICnF3c7vJC&index=32>.
- [10] S. Nguyen, “YouTube,” 22 4 2019. [Trực tuyến]. Available: <https://www.youtube.com/watch?v=vZO6mIsCqJ8&list=PLZEIt444jqBPoqtW2ARJp9ICnF3c7vJC&index=33>.
- [11] Son Nguyen, “YouTube,” [Trực tuyến]. Available: <https://www.youtube.com/watch?v=jFA3zCqNZyU&list=PLZEIt444jqBPoqtW2ARJp9ICnF3c7vJC&index=34>.
- [12] S. Nguyen, “YouTube,” 23 4 2019. [Trực tuyến]. Available: <https://www.youtube.com/watch?v=CZJdNoxg5ro&list=PLZEIt444jqBPoqtW2ARJp9ICnF3c7vJC&index=35>.
- [13] S. Nguyen, “YouTube,” 24 4 2019. [Trực tuyến]. Available: <https://www.youtube.com/watch?v=MXE4dWOw9JE&list=PLZEIt444jqBPoqtW2ARJp9ICnF3c7vJC&index=36>.
- [14] S. Nguyen, “YouTube,” 25 4 2019. [Trực tuyến]. Available: <https://www.youtube.com/watch?v=MXE4dWOw9JE&list=PLZEIt444jqBPoqtW2ARJp9ICnF3c7vJC&index=36>.