

✓ Practical Image Processing and Natural Language Processing

Image Classification using Convolutional Neural Networks (CNN)

Student Name:

(Salam Nader Khouli)

Submission Date:

28/6/2026

```
from google.colab import files

uploaded = files.upload()
```

```
from google.colab import files

uploaded = files.upload()
```

Choose Files archive.zip
archive.zip(application/x-zip-compressed) - 825979578 bytes, last modified: 28/06/2026 - 100% done
Saving archive.zip to archive.zip

```
import zipfile

with zipfile.ZipFile('archive.zip', 'r') as zip_ref:
    zip_ref.extractall('dataset')

print("Done!")
```

Done!

```
import os

print(os.listdir("dataset"))
```

['readme[1].txt', 'MSR-LA - 3467.docx', 'PetImages']

```
import os

print(os.listdir("dataset/PetImages"))
```

['Dog', 'Cat']

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import os
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=20,
    zoom_range=0.2,
    horizontal_flip=True
)

train_generator = train_datagen.flow_from_directory(
    "dataset/PetImages",
    target_size=(150, 150),
    batch_size=32,
    class_mode="binary",
    subset="training"
)

validation_generator = train_datagen.flow_from_directory(
    "dataset/PetImages",
    target_size=(150, 150),
```

```

    batch_size=32,
    class_mode="binary",
    subset="validation"
)

```

Found 20000 images belonging to 2 classes.
Found 4998 images belonging to 2 classes.

```

from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=(150,150,3)),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(128, (3,3), activation='relu'),
    layers.MaxPooling2D(2,2),

    layers.Flatten(),
    layers.Dense(512, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

model.compile(
    optimizer='adam',
    loss='binary_crossentropy',
    metrics=['accuracy']
)

model.summary()

```

/usr/local/lib/python3.12/dist-packages/keras/src/layers/convolutional/base_conv.py:113: UserWarning: Do not pass an `input\_shape` argument to `Conv2D` or `Conv3D` layers. It will be inferred from the input data.
super().__init__(activity_regularizer=activity_regularizer, **kwargs)

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)	0
flatten (Flatten)	(None, 36992)	0
dense (Dense)	(None, 512)	18,940,416
dense_1 (Dense)	(None, 1)	513

Total params: 19,034,177 (72.61 MB)
Trainable params: 19,034,177 (72.61 MB)
Non-trainable params: 0 (0.00 B)

```

import os
from PIL import Image

dataset_dir = "dataset/PetImages"

deleted = 0

for folder in ["Cat", "Dog"]:
    folder_path = os.path.join(dataset_dir, folder)

    for filename in os.listdir(folder_path):
        file_path = os.path.join(folder_path, filename)

        try:
            img = Image.open(file_path)
            img.verify()
        except:
            os.remove(file_path)
            deleted += 1

print(f"Deleted {deleted} corrupted images.")

```

```
/usr/local/lib/python3.12/dist-packages/PIL/TiffImagePlugin.py:950: UserWarning: Truncated File Read
warnings.warn(str(msg))
Deleted 4 corrupted images.
```

```
history = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=10
)
```

```
Epoch 1/10
625/625 ————— 1215s 2s/step - accuracy: 0.6287 - loss: 0.6447 - val_accuracy: 0.6753 - val_loss: 0.5962
Epoch 2/10
625/625 ————— 1218s 2s/step - accuracy: 0.6959 - loss: 0.5768 - val_accuracy: 0.7451 - val_loss: 0.5191
Epoch 3/10
625/625 ————— 1208s 2s/step - accuracy: 0.7477 - loss: 0.5189 - val_accuracy: 0.7773 - val_loss: 0.4715
Epoch 4/10
625/625 ————— 1204s 2s/step - accuracy: 0.7717 - loss: 0.4702 - val_accuracy: 0.7591 - val_loss: 0.4973
Epoch 5/10
625/625 ————— 1206s 2s/step - accuracy: 0.7915 - loss: 0.4415 - val_accuracy: 0.7791 - val_loss: 0.4667
Epoch 6/10
625/625 ————— 1189s 2s/step - accuracy: 0.8105 - loss: 0.4147 - val_accuracy: 0.8095 - val_loss: 0.4135
Epoch 7/10
625/625 ————— 1191s 2s/step - accuracy: 0.8248 - loss: 0.3873 - val_accuracy: 0.8277 - val_loss: 0.3795
Epoch 8/10
625/625 ————— 1196s 2s/step - accuracy: 0.8367 - loss: 0.3706 - val_accuracy: 0.8317 - val_loss: 0.3651
Epoch 9/10
625/625 ————— 1244s 2s/step - accuracy: 0.8431 - loss: 0.3558 - val_accuracy: 0.8485 - val_loss: 0.3494
Epoch 10/10
625/625 ————— 1202s 2s/step - accuracy: 0.8517 - loss: 0.3385 - val_accuracy: 0.8557 - val_loss: 0.3422
```

```
test_loss, test_accuracy = model.evaluate(validation_generator)
```

```
print("Test Loss:", test_loss)
print("Test Accuracy:", test_accuracy)
```

```
157/157 ————— 94s 595ms/step - accuracy: 0.8509 - loss: 0.3470
Test Loss: 0.34695518016815186
Test Accuracy: 0.8509403467178345
```

```
import matplotlib.pyplot as plt
```

```
plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1,2,2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

