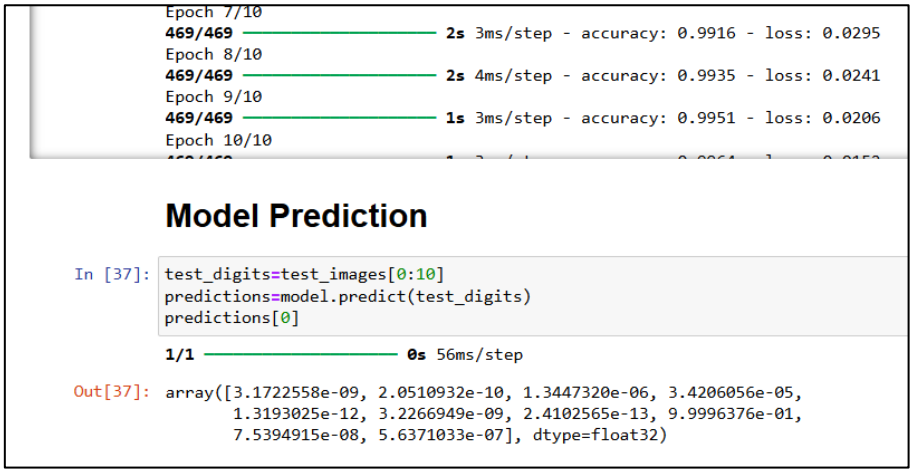
Assignment-1

1. **Create a neural network with one dense layer for MNIST dataset. Modify the network to achieve better accuracy by increasing the number of hidden layers and number of neurons.**

   With one dense layer:

```
from tensorflow import keras
from tensorflow.keras import layers
model=keras.Sequential([
    layers.Dense(256,activation='relu'),
    layers.Dense(10,activation='softmax')
])
```

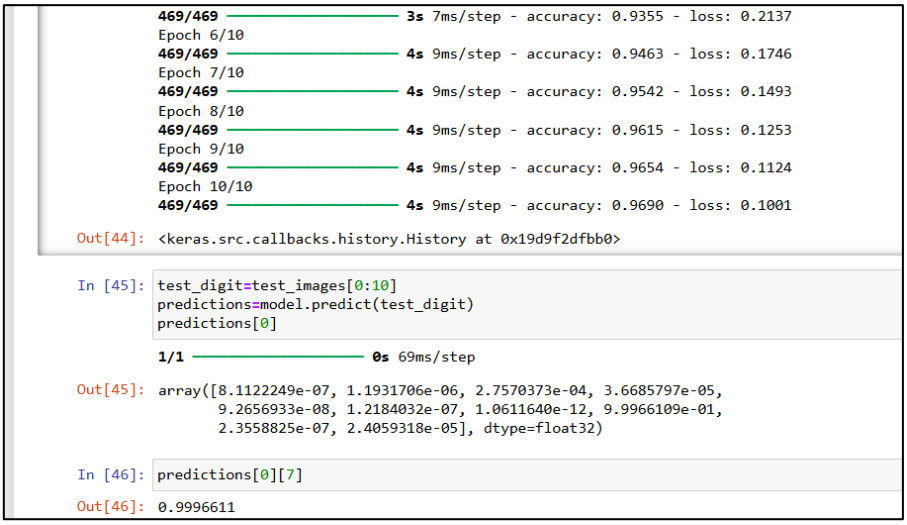
Output:



After modifying:

```
from tensorflow import keras
from tensorflow.keras import layers
model=keras.Sequential([
    layers.Dense(256,activation='relu'),
    layers.Dense(512,activation='relu'),
    layers.Dense(512,activation='relu'),
    layers.Dense(10,activation='softmax')
])
```

Output:



2. Explain the maths of a MLP with loss function as Cross entropy (negative log likelihood) for softmax classifier. {Take hint from class notes for regression MLP}

A **Multi-layer Perceptron** is a type of neural network designed for supervised learning tasks such as regression and classification. It consists of an input layer, one or more hidden layers for feature extraction and an output layer that provides predictions and classified output. It also contains parameters weights and biases that are learned during training. Mathematics for MLP includes Forward Propagation, Activation Functions, Back Propagation and Loss Functions.

1. Forward Pass in an MLP

A Multilayer Perceptron (MLP) consists of multiple layers of neurons. Each layer transforms its input using weights, biases, and activation functions. Let's assume a network with:
- Input: $x \in \mathbb{R}^d$ (where d is the input dimensionality).
- Hidden Layers: Each hidden layer applies a linear transformation followed by a non-linear activation function (e.g., ReLU).
- Output Layer: A final linear transformation followed by the softmax function.

Hidden Layer Transformation

For a hidden layer h, the output is:

$h = f(Whx + bh)$.

Output Layer Transformation

The output layer computes logits z:

$z = W_o h + b_o$

Softmax Function

The softmax function converts logits into probabilities:

$\hat{y}_i = exp(z_i) / \Sigma_j exp(z_j)$

Where $\hat{y}_i$ is the predicted probability for class i.

2. Cross-Entropy Loss Function

The cross-entropy loss quantifies the difference between the predicted probability distribution and the true distribution. For a single example, the loss is:

$$L = -\Sigma_i \, y_i \, log(\hat{y}_i)$$

Where:
- y_i is the true label in one-hot encoding (1 for the correct class, 0 otherwise).
- $\hat{y}_i$ is the predicted probability for class i.

For a dataset with n samples:

$$L\_total = (1/n) \, \Sigma_k \, L^{(k)}$$

Where $L^{(k)}$ is the loss for the k^{th} sample.

3. Gradient Computation (Backpropagation)

Gradient of Softmax Function

The derivative of the softmax function with respect to the logits z_i is:

$$\partial \hat{y}_i / \partial z_j = \hat{y}_i \, (\delta_{ij} - \hat{y}_j)$$

Where δ_{ij} is the Kronecker delta ($\delta_{ij} = 1$ if i = j, else 0).

Gradient of Cross-Entropy Loss

For the loss function:

$$\partial L / \partial z_i = \hat{y}_i - y_i$$

Backpropagation Through Layers

1. Output Layer:

$$\partial L / \partial W_o = (\partial L / \partial z) \, h^T, \; \partial L / \partial b_o = \partial L / \partial z$$

2. Hidden Layers:

$$\partial L / \partial Wh = (\partial L / \partial h) \, x^T, \; \partial L / \partial bh = \partial L / \partial h$$

Where $\partial L / \partial h$ depends on the activation function.

3. What is Vanishing gradient and Exploding gradient problem in neural networks.

Vanishing Gradient Problem

The vanishing gradient problem occurs during the training of deep neural networks when gradients of the loss function with respect to earlier layers become extremely small. This makes it difficult for the weights of these layers to update effectively, leading to slow or stalled learning.

Mathematically:

- During backpropagation, gradients are calculated using the chain rule:
  $\partial L/\partial W_1 = (\partial L/\partial a_n)\,(\partial a_n/\partial a_{n-1}) \ldots (\partial a_2/\partial a_1)\,(\partial a_1/\partial W_1)$.
- If the activation functions have derivatives less than 1 (e.g., sigmoid or tanh), repeated multiplications can shrink the gradients exponentially as they propagate backward through layers
- As a result, the weights in earlier layers receive minimal updates.

Consequences:

- Slow convergence or no convergence during training.
- Poor performance of the model due to insufficient learning in early layers.

Exploding Gradient Problem

The exploding gradient problem occurs when gradients during backpropagation grow exponentially as they propagate backward through the layers. This results in extremely large updates to weights, which can destabilize the model.

Mathematically:

- During backpropagation, if activation functions or weights have large values, repeated multiplications can cause gradients to increase exponentially as they propagate backward through layers.
- This can lead to very large weight updates and numerical instability.

Import libraries, load and transform data

In [1]:
```
!pip install -U -q evaluate transformers datasets>=2.14.5 accelerate>=0.27 mlflow
```

In [2]:
```
import warnings
warnings.filterwarnings("ignore")

import gc
import numpy as np
import pandas as pd
import itertools
from collections import Counter
import matplotlib.pyplot as plt
from sklearn.metrics import  accuracy_score, roc_auc_score, confusion_matrix,clas


from imblearn.over_sampling import RandomOverSampler
import accelerate
import evaluate
from datasets import Dataset, Image, ClassLabel
from transformers import (
    TrainingArguments,
    Trainer,
    ViTImageProcessor,
    ViTForImageClassification,
    DefaultDataCollator
)
import torch
from torch.utils.data import DataLoader
from torchvision.transforms import CenterCrop,Compose,Normalize,RandomRotation,Ra
```

In [3]:
```
from PIL import ImageFile


ImageFile.LOAD_TRUNCATED_IMAGES = True
```

In [4]:

```
image_dict = {}


from pathlib import Path
from tqdm import tqdm
import os

file_names = []
labels = []


for file in sorted((Path('/kaggle/input/cifake-real-and-ai-generated-synthetic-im
    label = str(file).split('/')[-2]
    labels.append(label)
    file_names.append(str(file))


print(len(file_names), len(labels))


df = pd.DataFrame.from_dict({"image": file_names, "label": labels})
print(df.shape)
```

```
120000 120000
(120000, 2)
```

In [5]:

```
df.head()
```

Out[5]:

	image	label
0	/kaggle/input/cifake-real-and-ai-generated-syn...	FAKE
1	/kaggle/input/cifake-real-and-ai-generated-syn...	FAKE
2	/kaggle/input/cifake-real-and-ai-generated-syn...	FAKE
3	/kaggle/input/cifake-real-and-ai-generated-syn...	FAKE
4	/kaggle/input/cifake-real-and-ai-generated-syn...	FAKE

In [6]:

```
df['label'].unique()
```

Out[6]:

```
array(['FAKE', 'REAL'], dtype=object)
```

In [7]:
```
y = df[['label']]
df = df.drop(['label'], axis=1)
ros = RandomOverSampler(random_state=83)
df, y_resampled = ros.fit_resample(df, y)
del y
df['label'] = y_resampled
del y_resampled
gc.collect()
print(df.shape)
```

```
(120000, 2)
```

In [8]:
```
dataset = Dataset.from_pandas(df).cast_column("image", Image())
```

In [9]:
```
dataset[0]["image"]
```

Out[9]:

In [10]:
```
labels_subset = labels[:5]
print(labels_subset)
```

```
['FAKE', 'FAKE', 'FAKE', 'FAKE', 'FAKE']
```

In [11]:
```
labels_list = ['REAL', 'FAKE']
label2id, id2label = dict(), dict()
for i, label in enumerate(labels_list):
    label2id[label] = i
    id2label[i] = label

print("Mapping of IDs to Labels:", id2label, '\n')
print("Mapping of Labels to IDs:", label2id)
```

```
Mapping of IDs to Labels: {0: 'REAL', 1: 'FAKE'}

Mapping of Labels to IDs: {'REAL': 0, 'FAKE': 1}
```

In [18]:
```
!pip uninstall -y ipywidgets
```

```
Found existing installation: ipywidgets 8.1.5
Uninstalling ipywidgets-8.1.5:
  Successfully uninstalled ipywidgets-8.1.5
```

In [19]:
```
!pip install ipywidgets
```

```
Collecting ipywidgets
  Downloading ipywidgets-8.1.5-py3-none-any.whl (139 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 139.8/139.8 kB 6.0 MB/s eta 0:00:0
0
Requirement already satisfied: comm>=0.1.3 in /opt/conda/lib/python3.10/site-pa
ckages (from ipywidgets) (0.1.3)
Requirement already satisfied: ipython>=6.1.0 in /opt/conda/lib/python3.10/site
-packages (from ipywidgets) (8.14.0)
Requirement already satisfied: traitlets>=4.3.1 in /opt/conda/lib/python3.10/si
te-packages (from ipywidgets) (5.9.0)
Requirement already satisfied: widgetsnbextension~=4.0.12 in /opt/conda/lib/pyt
hon3.10/site-packages (from ipywidgets) (4.0.13)
Requirement already satisfied: jupyterlab-widgets~=3.0.12 in /opt/conda/lib/pyt
hon3.10/site-packages (from ipywidgets) (3.0.13)
Requirement already satisfied: backcall in /opt/conda/lib/python3.10/site-packa
ges (from ipython>=6.1.0->ipywidgets) (0.2.0)
Requirement already satisfied: decorator in /opt/conda/lib/python3.10/site-pack
ages (from ipython>=6.1.0->ipywidgets) (5.1.1)
Requirement already satisfied: jedi>=0.16 in /opt/conda/lib/python3.10/site-pac
kages (from ipython>=6.1.0->ipywidgets) (0.18.2)
Requirement already satisfied: matplotlib-inline in /opt/conda/lib/python3.10/s
ite-packages (from ipython>=6.1.0->ipywidgets) (0.1.6)
Requirement already satisfied: pickleshare in /opt/conda/lib/python3.10/site-pa
ckages (from ipython>=6.1.0->ipywidgets) (0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.37,<3.1.0,>=3.0.30 in /opt/c
onda/lib/python3.10/site-packages (from ipython>=6.1.0->ipywidgets) (3.0.38)
Requirement already satisfied: pygments>=2.4.0 in /opt/conda/lib/python3.10/sit
e-packages (from ipython>=6.1.0->ipywidgets) (2.15.1)
Requirement already satisfied: stack-data in /opt/conda/lib/python3.10/site-pac
kages (from ipython>=6.1.0->ipywidgets) (0.6.2)
Requirement already satisfied: pexpect>4.3 in /opt/conda/lib/python3.10/site-pa
ckages (from ipython>=6.1.0->ipywidgets) (4.8.0)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in /opt/conda/lib/python3.1
0/site-packages (from jedi>=0.16->ipython>=6.1.0->ipywidgets) (0.8.3)
Requirement already satisfied: ptyprocess>=0.5 in /opt/conda/lib/python3.10/sit
e-packages (from pexpect>4.3->ipython>=6.1.0->ipywidgets) (0.7.0)
Requirement already satisfied: wcwidth in /opt/conda/lib/python3.10/site-packag
es (from prompt-toolkit!=3.0.37,<3.1.0,>=3.0.30->ipython>=6.1.0->ipywidgets)
(0.2.6)
Requirement already satisfied: executing>=1.2.0 in /opt/conda/lib/python3.10/si
te-packages (from stack-data->ipython>=6.1.0->ipywidgets) (1.2.0)
Requirement already satisfied: asttokens>=2.1.0 in /opt/conda/lib/python3.10/si
te-packages (from stack-data->ipython>=6.1.0->ipywidgets) (2.2.1)
Requirement already satisfied: pure-eval in /opt/conda/lib/python3.10/site-pack
ages (from stack-data->ipython>=6.1.0->ipywidgets) (0.2.2)
Requirement already satisfied: six in /opt/conda/lib/python3.10/site-packages
(from asttokens>=2.1.0->stack-data->ipython>=6.1.0->ipywidgets) (1.16.0)
Installing collected packages: ipywidgets
Successfully installed ipywidgets-8.1.5
```

In [21]:
```
!pip list | grep ipywidgets
```

```
ipywidgets                            8.1.5
```

In [22]:
```
!jupyter nbextension enable --py widgetsnbextension --sys-prefix
```

```
Config option `kernel_spec_manager_class` not recognized by `EnableNBExtensionA
pp`.
Enabling notebook extension jupyter-js-widgets/extension...
      - Validating: OK
```

In [23]:
```
from datasets import ClassLabel

# Define ClassLabels
ClassLabels = ClassLabel(num_classes=len(labels_list), names=labels_list)

# Function to map label names to IDs
def map_label2id(example):
    example['label'] = ClassLabels.str2int(example['label'])
    return example

# Apply mapping
dataset = dataset.map(map_label2id, batched=True)

# Cast the label column
dataset = dataset.cast_column('label', ClassLabels)

# Apply train-test split only to the 'train' split if it exists
if 'train' in dataset:
    dataset_split = dataset['train'].train_test_split(test_size=0.4, shuffle=True
    dataset['train'] = dataset_split['train']
    dataset['test'] = dataset_split['test']
else:
    dataset = dataset.train_test_split(test_size=0.4, shuffle=True, stratify_by_c

# Assign train and test datasets
train_data = dataset['train']
test_data = dataset['test']
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [24]:
```
model_str = "dima806/ai_vs_real_image_detection"
processor = ViTImageProcessor.from_pretrained(model_str)
image_mean, image_std = processor.image_mean, processor.image_std
size = processor.size["height"]
print("Size: ", size)
normalize = Normalize(mean=image_mean, std=image_std)
_train_transforms = Compose(
    [
        Resize((size, size)),
        RandomRotation(90),
        RandomAdjustSharpness(2),
        ToTensor(),
        normalize
    ]
)


_val_transforms = Compose(
    [
        Resize((size, size)),
        ToTensor(),
        normalize
    ]
)

def train_transforms(examples):
    examples['pixel_values'] = [_train_transforms(image.convert("RGB")) for image
    return examples


def val_transforms(examples):
    examples['pixel_values'] = [_val_transforms(image.convert("RGB")) for image i
    return examples
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

```
Size:  224
```

In [25]:
```
train_data.set_transform(train_transforms)
test_data.set_transform(val_transforms)
```

In [26]:
```
def collate_fn(examples):
    pixel_values = torch.stack([example["pixel_values"] for example in examples])
    labels = torch.tensor([example['label'] for example in examples])
    return {"pixel_values": pixel_values, "labels": labels}
```

Load, train, and evaluate model

In [27]:
```
model = ViTForImageClassification.from_pretrained(model_str, num_labels=len(label
model.config.id2label = id2label
model.config.label2id = label2id
print(model.num_parameters(only_trainable=True) / 1e6)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

```
85.800194
```

In [29]:
```
accuracy = evaluate.load("accuracy")
def compute_metrics(eval_pred):

    predictions = eval_pred.predictions
    label_ids = eval_pred.label_ids
    predicted_labels = predictions.argmax(axis=1)
    acc_score = accuracy.compute(predictions=predicted_labels, references=label_i
    return {
        "accuracy": acc_score
    }
```

In [30]:
```
metric_name = "accuracy"
model_name = "ai_vs_real_image_detection"
num_train_epochs = 2
args = TrainingArguments(
    output_dir=model_name,
    logging_dir='./logs',
    evaluation_strategy="epoch",
    learning_rate=1e-6,
    per_device_train_batch_size=64,
    per_device_eval_batch_size=32,
    num_train_epochs=num_train_epochs,
    weight_decay=0.02,
    warmup_steps=50,
    remove_unused_columns=False,
    save_strategy='epoch',
    load_best_model_at_end=True,
    save_total_limit=1,
    report_to="none"
)
```

In [31]:
```
trainer = Trainer(
    model,
    args,
    train_dataset=train_data,
    eval_dataset=test_data,
    data_collator=collate_fn,
    compute_metrics=compute_metrics,
    tokenizer=processor,
)
```

In [32]:
```
trainer.evaluate()
```

[324/324 15:11]

Out[32]:
```
{'eval_loss': 0.043403297662734985,
 'eval_model_preparation_time': 0.0035,
 'eval_accuracy': 0.9858217592592593,
 'eval_runtime': 141.4132,
 'eval_samples_per_second': 73.317,
 'eval_steps_per_second': 2.291}
```

In [33]:
```
trainer.train()
```

[486/486 15:22, Epoch 2/2]

Epoch	Training Loss	Validation Loss	Model Preparation Time	Accuracy
1	No log	0.043228	0.003500	0.986400
2	No log	0.042479	0.003500	0.985436

Out[33]:
```
TrainOutput(global_step=486, training_loss=0.034261570055298354, metrics={'trai
n_runtime': 925.3218, 'train_samples_per_second': 33.614, 'train_steps_per_seco
nd': 0.525, 'total_flos': 2.4103108449722696e+18, 'train_loss': 0.0342615700552
98354, 'epoch': 2.0})
```

In [34]:
```
trainer.evaluate()
```

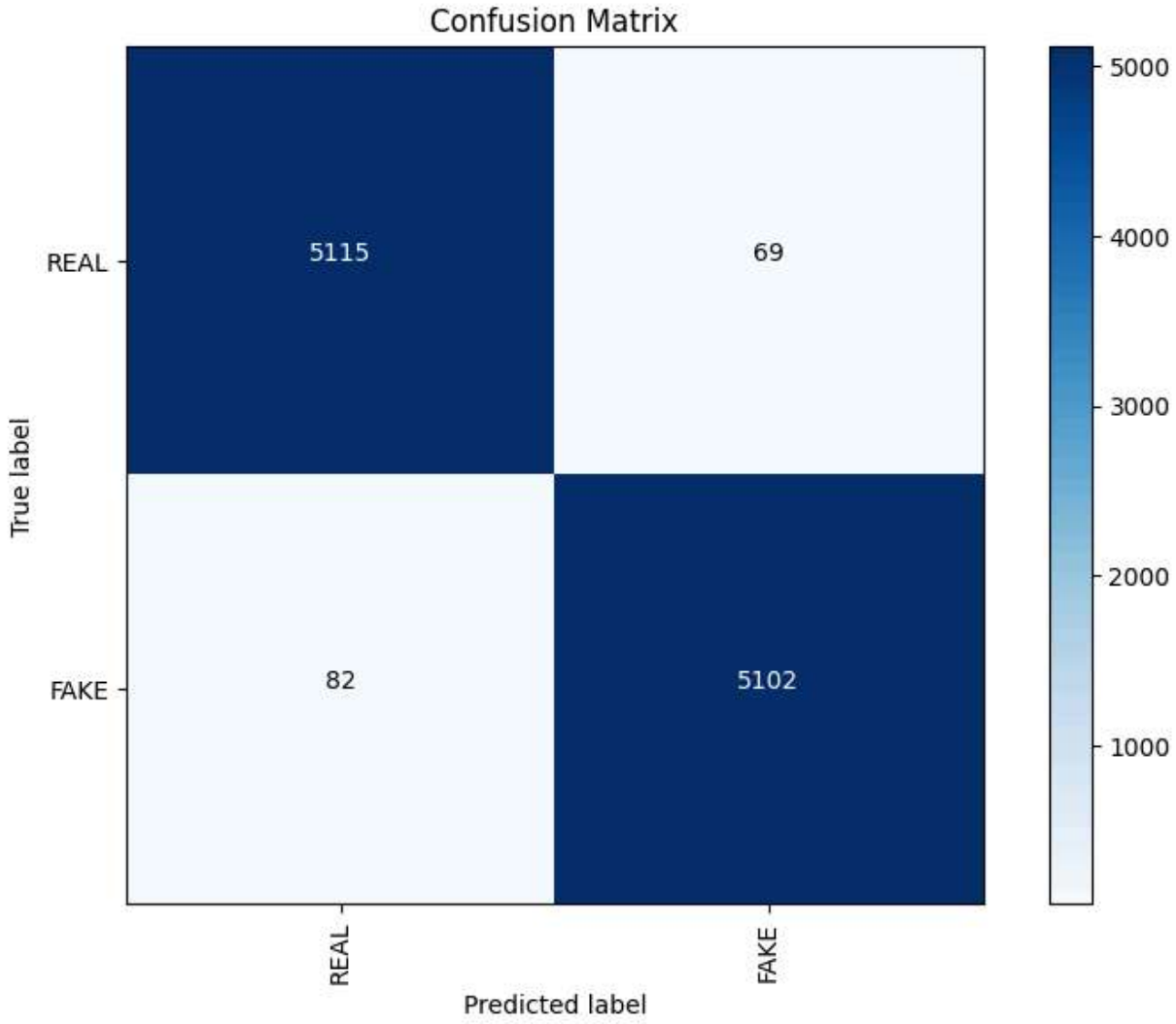
Out[34]:
```
{'eval_loss': 0.04247914254665375,
 'eval_model_preparation_time': 0.0035,
 'eval_accuracy': 0.9854359567901234,
 'eval_runtime': 91.5588,
 'eval_samples_per_second': 113.239,
 'eval_steps_per_second': 3.539,
 'epoch': 2.0}
```

In [35]:
```
outputs = trainer.predict(test_data)
print(outputs.metrics)
```

```
{'test_loss': 0.04247914254665375, 'test_model_preparation_time': 0.0035, 'test
_accuracy': 0.9854359567901234, 'test_runtime': 84.3297, 'test_samples_per_seco
nd': 122.946, 'test_steps_per_second': 3.842}
```

In [36]:
```
y_true = outputs.label_ids
y_pred = outputs.predictions.argmax(1)


def plot_confusion_matrix(cm, classes, title='Confusion Matrix', cmap=plt.cm.Blue

    plt.figure(figsize=figsize)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)

    fmt = '.0f'
    thresh = cm.max() / 2.0
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt), horizontalalignment="center", color


    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
    plt.show()
accuracy = accuracy_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred, average='macro')

print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1:.4f}")
if len(labels_list) <= 150:
    cm = confusion_matrix(y_true, y_pred)
    plot_confusion_matrix(cm, labels_list, figsize=(8, 6))
print()
print("Classification report:")
print()
print(classification_report(y_true, y_pred, target_names=labels_list, digits=4))
```

```
Accuracy: 0.9854
F1 Score: 0.9854
```




```
Classification report:

              precision    recall  f1-score   support

        REAL     0.9842    0.9867    0.9855      5184
        FAKE     0.9867    0.9842    0.9854      5184

    accuracy                         0.9854     10368
   macro avg     0.9854    0.9854    0.9854     10368
weighted avg     0.9854    0.9854    0.9854     10368
```

In [37]:

```
trainer.save_model()
```

In [38]:
```
from transformers import pipeline

pipe = pipeline('image-classification', model=model_name, device=0)
```

```
Using a slow image processor as `use_fast` is unset and a slow processor was sa
ved with this model. `use_fast=True` will be the default behavior in v4.48, eve
n if the model was saved with a slow processor. This will result in minor diffe
rences in outputs. You'll still be able to use a slow processor with `use_fast=
False`.
Device set to use cuda:0
```

In [39]:
```
image = test_data[1]["image"]

.
image
```

Out[39]:

In [40]:
```
pipe(image)
```

Out[40]:
```
[{'label': 'REAL', 'score': 0.9987921118736267},
 {'label': 'FAKE', 'score': 0.0012079098960384727}]
```

In [41]:
```
id2label[test_data[1]["label"]]
```

Out[41]:
```
'REAL'
```

In [ ]:

In [1]:
```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/dock
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list a

import os
for dirname, _, filenames in os.walk('/kaggle/input/clothes-dataset'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets
# You can also write temporary files to /kaggle/temp/, but they won't be saved ou
```

```
07d7154538.jpg
/kaggle/input/clothes-dataset/Clothes_Dataset/Gaun/kemeja_dress_putih_oversiz
e_1697091330_37c86162_progressive_thumbnail.jpg
/kaggle/input/clothes-dataset/Clothes_Dataset/Gaun/e59186b6-abfa-4f6a-9e7b-2c
7e2cf3fd22.jpg
/kaggle/input/clothes-dataset/Clothes_Dataset/Gaun/3193786c-22f1-4ea9-9e84-8e
e47c138a8d.jpg
/kaggle/input/clothes-dataset/Clothes_Dataset/Gaun/dress_pita_depan_169710304
6_c7cc20e1_progressive_thumbnail.jpg
/kaggle/input/clothes-dataset/Clothes_Dataset/Gaun/hm_dress_1697109054_8b47e0
cb_progressive_thumbnail.jpg
/kaggle/input/clothes-dataset/Clothes_Dataset/Gaun/fc5051e5-59aa-44e8-88dc-15
1981e61dcc.jpg
/kaggle/input/clothes-dataset/Clothes_Dataset/Gaun/dress_cantik_bahan_jersey_
prem_1697106977_53491f82_progressive_thumbnail.jpg
/kaggle/input/clothes-dataset/Clothes_Dataset/Gaun/love_bonito_womans_midi_dr
ess_1697087445_309021c7_progressive_thumbnail.jpg
/kaggle/input/clothes-dataset/Clothes_Dataset/Gaun/f68947a8-6f6b-4461-ab13-81
c285908509.jpg
```

In [2]:
```
!pip install evaluate
import evaluate # for evaluating functions
print(evaluate.__version__)
```

```
Collecting evaluate
  Downloading evaluate-0.4.3-py3-none-any.whl.metadata (9.2 kB)
Requirement already satisfied: datasets>=2.0.0 in /usr/local/lib/python3.10/d
ist-packages (from evaluate) (3.2.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-
packages (from evaluate) (1.26.4)
Requirement already satisfied: dill in /usr/local/lib/python3.10/dist-package
s (from evaluate) (0.3.8)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packa
ges (from evaluate) (2.2.3)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/
dist-packages (from evaluate) (2.32.3)
Requirement already satisfied: tqdm>=4.62.1 in /usr/local/lib/python3.10/dist
-packages (from evaluate) (4.67.1)
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packa
ges (from evaluate) (3.5.0)
Requirement already satisfied: multiprocess in /usr/local/lib/python3.10/dist
-packages (from evaluate) (0.70.16)
Requirement already satisfied: fsspec>=2021.05.0 in /usr/local/lib/python3.1
```

In [3]:
```
import gc # garbage collection
import numpy as np
import pandas as pd
import itertools # for iterators and looping
from collections import Counter
```

In [4]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import (accuracy_score,
                             roc_auc_score,
                             confusion_matrix,
                             classification_report,
                             f1_score)
```

In [5]:
```
from imblearn.over_sampling import RandomOverSampler
import accelerate
import evaluate
from transformers import (TrainingArguments,
                          Trainer,
                          ViTImageProcessor,
                          ViTForImageClassification,
                          DefaultDataCollator
)
```

In [6]:
```
import torch
from torch.utils.data import DataLoader
from torchvision.transforms import (CenterCrop,Compose,Normalize,RandomRotation,
```

In [7]:
```
from PIL import ImageFile
ImageFile.LOAD_TRUNCATED_IMAGES = True
```

In [8]:
```
image_dict = {}

from pathlib import Path
from tqdm import tqdm
import os

file_names = []
labels = []
```

In [9]:
```
label_dict = {
    "Blazer": "Blazer",
    "Celana_Panjang": "Long Pants",
    "Celana_Pendek": "Shorts",
    "Gaun": "Dresses",
    "Hoodie": "Hoodie",
    "Jaket": "Jacket",
    "Jaket_Denim": "Denim Jacket",
    "Jaket_Olahraga": "Sports Jacket",
    "Jeans": "Jeans",
    "Kaos": "T-shirt",
    "Kemeja": "Shirt",
    "Mantel": "Coat",
    "Polo": "Polo",
    "Rok": "Skirt",
    "Sweter": "Sweater",
}
```

In [10]:
```
for file in tqdm(sorted((Path('/kaggle/input/clothes-dataset/Clothes_Dataset/').g
    label = str(file).split('/')[-2]  # Extract the label from the file path
    labels.append(label_dict[label])  # Add the label to the list
    file_names.append(str(file))  # Add the file path to the list


print(len(file_names), len(labels))

df = pd.DataFrame.from_dict({"image": file_names, "label": labels})
print(df.shape)
```

```
100%|██████████| 7500/7500 [00:00<00:00, 416757.59it/s]

7500 7500
(7500, 2)
```

In [11]:
```
df.head()
```

Out[11]:

	image	label
0	/kaggle/input/clothes-dataset/Clothes_Dataset/...	Blazer
1	/kaggle/input/clothes-dataset/Clothes_Dataset/...	Blazer
2	/kaggle/input/clothes-dataset/Clothes_Dataset/...	Blazer
3	/kaggle/input/clothes-dataset/Clothes_Dataset/...	Blazer
4	/kaggle/input/clothes-dataset/Clothes_Dataset/...	Blazer

In [12]:
```
df["label"].value_counts()
```

Out[12]:
```
label
Blazer           500
Long Pants       500
Shorts           500
Dresses          500
Hoodie           500
Jacket           500
Denim Jacket     500
Sports Jacket    500
Jeans            500
T-shirt          500
Shirt            500
Coat             500
Polo             500
Skirt            500
Sweater          500
Name: count, dtype: int64
```

In [13]:
```
from datasets import Dataset
from datasets import Image
dataset = Dataset.from_pandas(df).cast_column("image",Image())
```

In [14]:
```
dataset[50]["image"]
```

Out[14]:

In [15]:
```
labels_subset = labels[:5]
print(labels_subset)
```

```
['Blazer', 'Blazer', 'Blazer', 'Blazer', 'Blazer']
```

In [16]:
```
labels_list = sorted(list(set(labels)))
```

In [17]:
```
label2id, id2label = dict(),dict()

for i,label in enumerate(labels_list):
    label2id[label] = i
    id2label[i] = label

print(id2label)
print(label2id)
```

```
{0: 'Blazer', 1: 'Coat', 2: 'Denim Jacket', 3: 'Dresses', 4: 'Hoodie', 5: 'Jack
et', 6: 'Jeans', 7: 'Long Pants', 8: 'Polo', 9: 'Shirt', 10: 'Shorts', 11: 'Ski
rt', 12: 'Sports Jacket', 13: 'Sweater', 14: 'T-shirt'}
{'Blazer': 0, 'Coat': 1, 'Denim Jacket': 2, 'Dresses': 3, 'Hoodie': 4, 'Jacke
t': 5, 'Jeans': 6, 'Long Pants': 7, 'Polo': 8, 'Shirt': 9, 'Shorts': 10, 'Skir
t': 11, 'Sports Jacket': 12, 'Sweater': 13, 'T-shirt': 14}
```

In [18]:
```
import datasets
from datasets import ClassLabel
ClassLabels = ClassLabel(num_classes = len(labels_list),names = labels_list)
def map_label2id(example):
    example['label'] = ClassLabels.str2int(example['label'])
    return example
```

In [19]:
```
dataset = dataset.map(map_label2id, batched=True)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [20]:
```
dataset = dataset.cast_column('label', ClassLabels)


dataset = dataset.train_test_split(test_size=0.4, shuffle=True, stratify_by_colu

train_data = dataset['train']

test_data = dataset['test']
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [21]:
```
model_str = 'google/vit-base-patch16-224-in21k'
```

In [22]:
```
processor = ViTImageProcessor.from_pretrained(model_str)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [23]:
```
image_mean, image_std = processor.image_mean, processor.image_std
```

In [24]:
```
size = processor.size["height"]
print("Size: ", size)
```

```
Size:  224
```

In [25]:
```
normalize = Normalize(mean=image_mean, std=image_std)
```

In [26]:
```
_train_transformers = Compose([
    Resize((size,size)),
    RandomRotation(90),
    RandomAdjustSharpness(2),
    RandomHorizontalFlip(0.5),
    ToTensor(),
    normalize
])
```

In [27]:
```
_val_transforms = Compose(
    [
        Resize((size,size)),
        ToTensor(),
        normalize
    ]
)
```

In [28]:
```
def train_transforms(examples):
    examples['pixel_values'] = [_train_transformers(image.convert("RGB")) for ima
    return examples
```

In [29]:
```
def val_transforms(examples):
    examples['pixel_values'] = [_val_transforms(image.convert("RGB")) for image i
    return examples
```

In [30]:
```
train_data.set_transform(train_transforms)
```

In [31]:
```
test_data.set_transform(val_transforms)
```

In [32]:
```
def collate_fn(examples):
    pixel_values = torch.stack([example["pixel_values"] for example in examples])
    labels = torch.tensor([example['label'] for example in examples])
    return {"pixel_values": pixel_values, "labels": labels}
```

In [33]:
```
model = ViTForImageClassification.from_pretrained(model_str, num_labels=len(label
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

```
Some weights of ViTForImageClassification were not initialized from the model c
heckpoint at google/vit-base-patch16-224-in21k and are newly initialized: ['cla
ssifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

In [34]:
```
model.config.id2label = id2label
model.config.label2id = label2id
```

In [35]:
```
print(model.num_parameters(only_trainable=True) / 1e6)
```

```
85.810191
```

In [36]:
```
accuracy = evaluate.load("accuracy")
def compute_metrics(eval_pred):
    predictions = eval_pred.predictions
    label_ids = eval_pred.label_ids
    predicted_labels = predictions.argmax(axis=1)
    acc_score = accuracy.compute(predictions=predicted_labels, references=label_i
    return {
        "accuracy": acc_score
    }
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [37]:
```
metric_name = "accuracy"
model_name = "clothes_image_detection"
num_train_epochs = 10
args = TrainingArguments(
    output_dir=model_name,
    logging_dir='./logs',
    evaluation_strategy="epoch",
    learning_rate=3e-6,
    per_device_train_batch_size=32,
    per_device_eval_batch_size=8,
    num_train_epochs=num_train_epochs,
    weight_decay=0.02,# it is use to prevent overfitting
    warmup_steps=50,
    remove_unused_columns=False,
    save_strategy='epoch',
    load_best_model_at_end=True,
    save_total_limit=1,
    report_to="none"
)
```

```
/usr/local/lib/python3.10/dist-packages/transformers/training_args.py:1575: Fut
ureWarning: `evaluation_strategy` is deprecated and will be removed in version
4.46 of 🤗 Transformers. Use `eval_strategy` instead
  warnings.warn(
```

In [38]:
```
trainer = Trainer(
    model,
    args,
    train_dataset=train_data,
    eval_dataset=test_data,
    data_collator=collate_fn,
    compute_metrics=compute_metrics,
    tokenizer=processor,
)
```

```
<ipython-input-38-9c13e1b762e1>:1: FutureWarning: `tokenizer` is deprecated and
will be removed in version 5.0.0 for `Trainer.__init__`. Use `processing_class`
instead.
  trainer = Trainer(
```

In [39]:
```
trainer.evaluate()
```

```
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: Use
rWarning: Was asked to gather along dimension 0, but all input tensors were sca
lars; will instead unsqueeze and return a vector.
  warnings.warn(
```

[188/188 05:08]

Out[39]:
```
{'eval_loss': 2.714015245437622,
 'eval_accuracy': 0.07233333333333333,
 'eval_runtime': 76.9229,
 'eval_samples_per_second': 39.0,
 'eval_steps_per_second': 2.444}
```

In [40]:
```
trainer.train()
```

```
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: Use
rWarning: Was asked to gather along dimension 0, but all input tensors were sca
lars; will instead unsqueeze and return a vector.
  warnings.warn(
```

[710/710 35:46, Epoch 10/10]

Epoch	Training Loss	Validation Loss	Accuracy
1	No log	2.688771	0.112667
2	No log	2.649584	0.200333
3	No log	2.607557	0.311000
4	No log	2.563295	0.397667
5	No log	2.523514	0.456667
6	No log	2.490482	0.509667
7	No log	2.464117	0.543333
8	2.574500	2.445119	0.562333
9	2.574500	2.433961	0.577667
10	2.574500	2.430238	0.584000

```
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: UserWarning: Was asked to gather along dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: UserWarning: Was asked to gather along dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: UserWarning: Was asked to gather along dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: UserWarning: Was asked to gather along dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: UserWarning: Was asked to gather along dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: UserWarning: Was asked to gather along dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: UserWarning: Was asked to gather along dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: UserWarning: Was asked to gather along dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: UserWarning: Was asked to gather along dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: UserWarning: Was asked to gather along dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
  warnings.warn(
```

Out[40]:

```
TrainOutput(global_step=710, training_loss=2.5268360406580106, metrics={'train_runtime': 2150.1722, 'train_samples_per_second': 20.929, 'train_steps_per_second': 0.33, 'total_flos': 3.48754583632896e+18, 'train_loss': 2.5268360406580106, 'epoch': 10.0})
```

In [41]:
```
trainer.evaluate()
```

```
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: Use
rWarning: Was asked to gather along dimension 0, but all input tensors were sca
lars; will instead unsqueeze and return a vector.
  warnings.warn(
```

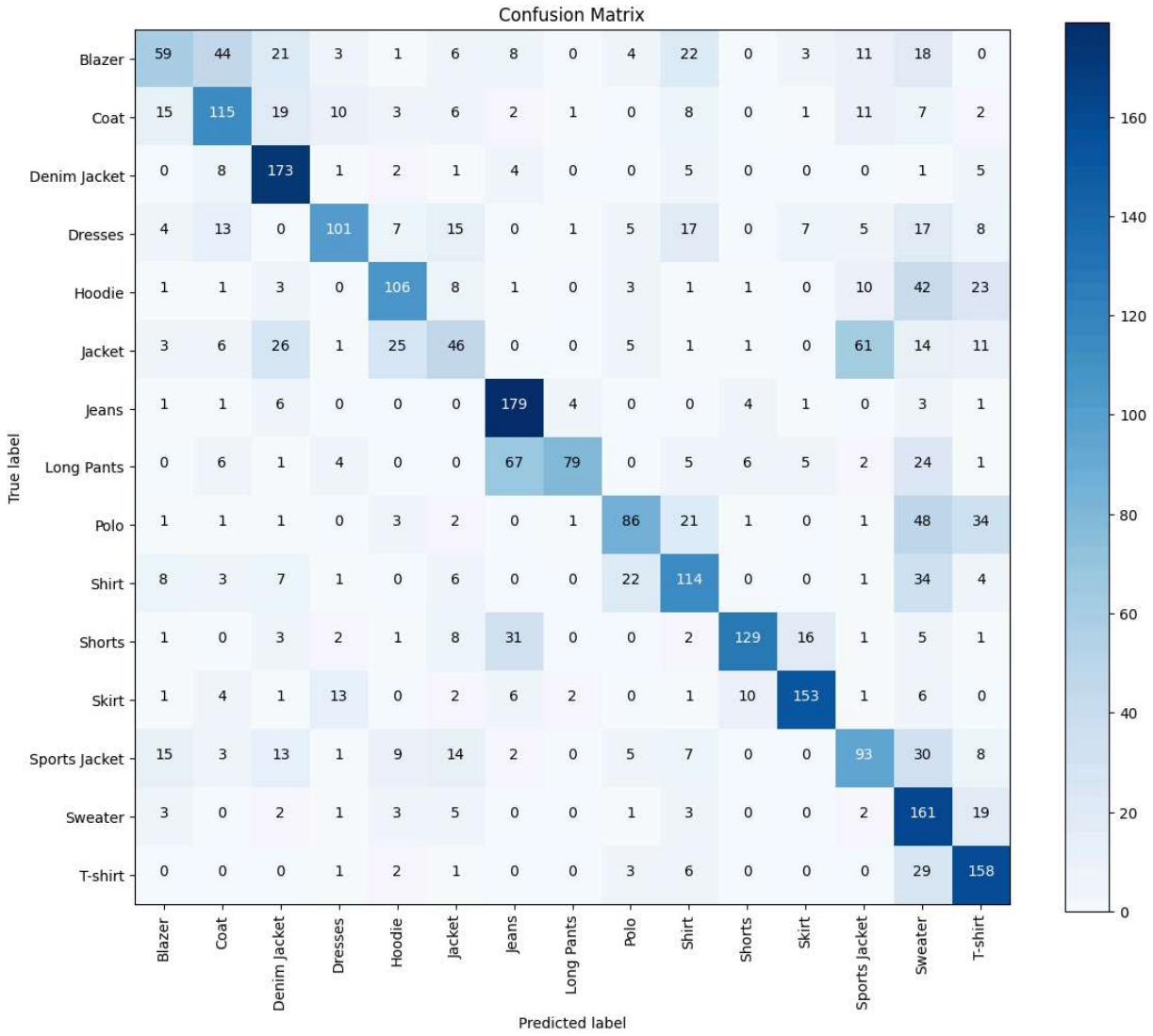
Out[41]:
```
{'eval_loss': 2.4302384853363037,
 'eval_accuracy': 0.584,
 'eval_runtime': 61.3954,
 'eval_samples_per_second': 48.864,
 'eval_steps_per_second': 3.062,
 'epoch': 10.0}
```

In [42]:
```
outputs = trainer.predict(test_data)
print(outputs.metrics)
```

```
/usr/local/lib/python3.10/dist-packages/torch/nn/parallel/_functions.py:71: Use
rWarning: Was asked to gather along dimension 0, but all input tensors were sca
lars; will instead unsqueeze and return a vector.
  warnings.warn(

{'test_loss': 2.4302384853363037, 'test_accuracy': 0.584, 'test_runtime': 61.60
22, 'test_samples_per_second': 48.7, 'test_steps_per_second': 3.052}
```

In [43]:
```
y_true = outputs.label_ids
y_pred = outputs.predictions.argmax(1)
def plot_confusion_matrix(cm, classes, title='Confusion Matrix', cmap=plt.cm.Blue
    """
    This function plots a confusion matrix.

    Parameters:
        cm (array-like): Confusion matrix as returned by sklearn.metrics.confusic
        classes (list): List of class names, e.g., ['Class 0', 'Class 1'].
        title (str): Title for the plot.
        cmap (matplotlib colormap): Colormap for the plot.
    """
    plt.figure(figsize=figsize)
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=90)
    plt.yticks(tick_marks, classes)
    fmt = '.0f'
    thresh = cm.max() / 2.0
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt), horizontalalignment="center", color
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
    plt.show()

accuracy = accuracy_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred, average='macro')

print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1:.4f}")

if len(labels_list) <= 150:
    cm = confusion_matrix(y_true, y_pred)
    plot_confusion_matrix(cm, labels_list, figsize=(12, 10))

print()
print("Classification report:")
print()
print(classification_report(y_true, y_pred, target_names=labels_list, digits=4))
```

```
Accuracy: 0.5840
F1 Score: 0.5756
```


Confusion Matrix
True label
Predicted label
Blazer
Coat
Denim Jacket
Dresses
Hoodie
Jacket
Jeans
Long Pants
Polo
Shirt
Shorts
Skirt
Sports Jacket
Sweater
T-shirt
59 44 21 3 1 6 8 0 4 22 0 3 11 18 0
15 115 19 10 3 6 2 1 0 8 0 1 11 7 2
0 8 173 1 2 1 4 0 0 5 0 0 0 1 5
4 13 0 101 7 15 0 1 5 17 0 7 5 17 8
1 1 3 0 106 8 1 0 3 1 1 0 10 42 23
3 6 26 1 25 46 0 0 5 1 1 0 61 14 11
1 1 6 0 0 0 179 4 0 0 4 1 0 3 1
0 6 1 4 0 0 67 79 0 5 6 5 2 24 1
1 1 1 0 3 2 0 1 86 21 1 0 1 48 34
8 3 7 1 0 6 0 0 22 114 0 0 1 34 4
1 0 3 2 1 8 31 0 0 2 129 16 1 5 1
1 4 1 13 0 2 6 2 0 1 10 153 1 6 0
15 3 13 1 9 14 2 0 5 7 0 0 93 30 8
3 0 2 1 3 5 0 0 1 3 0 0 2 161 19
0 0 0 1 2 1 0 0 3 6 0 0 0 29 158
160
140
120
100
80
60
40
20
0

```
Classification report:

              precision    recall  f1-score   support

      Blazer     0.5268    0.2950    0.3782       200
        Coat     0.5610    0.5750    0.5679       200
 Denim Jacket     0.6268    0.8650    0.7269       200
     Dresses     0.7266    0.5050    0.5959       200
      Hoodie     0.6543    0.5300    0.5856       200
      Jacket     0.3833    0.2300    0.2875       200
       Jeans     0.5967    0.8950    0.7160       200
   Long Pants     0.8977    0.3950    0.5486       200
        Polo     0.6418    0.4300    0.5150       200
       Shirt     0.5352    0.5700    0.5521       200
      Shorts     0.8487    0.6450    0.7330       200
       Skirt     0.8226    0.7650    0.7927       200
Sports Jacket     0.4673    0.4650    0.4662       200
     Sweater     0.3667    0.8050    0.5039       200
     T-shirt     0.5745    0.7900    0.6653       200

    accuracy                         0.5840      3000
   macro avg     0.6153    0.5840    0.5756      3000
weighted avg     0.6153    0.5840    0.5756      3000
```

In [44]:
```
trainer.save_model()
```

In [45]:
```
from transformers import pipeline
pipe = pipeline('image-classification', model=model_name, device=0)
```

```
Device set to use cuda:0
```

In [46]:
```
image = test_data[1]["image"]
image
```

Out[46]:

In [47]:
```
pipe(image)
```

Out[47]:
```
[{'label': 'Denim Jacket', 'score': 0.6254792809486389},
 {'label': 'Shirt', 'score': 0.5571456551551819},
 {'label': 'Sports Jacket', 'score': 0.5241682529449463},
 {'label': 'Polo', 'score': 0.5233001112937927},
 {'label': 'Jeans', 'score': 0.5214868187904358}]
```

In [48]:
```
id2label[test_data[1]["label"]]
```

Out[48]:
```
'Denim Jacket'
```

In [1]:
```
pip install seaborn
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packag
es (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in /usr/local/lib/python3.1
0/dist-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=0.25 in /usr/local/lib/python3.10/dist-p
ackages (from seaborn) (2.2.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in /usr/local/lib/python
3.10/dist-packages (from seaborn) (3.7.5)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/di
st-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-p
ackages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/d
ist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.55.3)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/d
ist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (24.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib!=3.6.1,>=3.1->seaborn) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/di
st-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.9.0.post0)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.10/dist-packag
es (from numpy!=1.24.0,>=1.17->seaborn) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python3.10/dist-pac
kages (from numpy!=1.24.0,>=1.17->seaborn) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.10/dist-pack
ages (from numpy!=1.24.0,>=1.17->seaborn) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.10/dist-packages
(from numpy!=1.24.0,>=1.17->seaborn) (2025.0.1)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.10/dist-package
s (from numpy!=1.24.0,>=1.17->seaborn) (2022.0.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python3.10/dist-pa
ckages (from numpy!=1.24.0,>=1.17->seaborn) (2.4.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-p
ackages (from pandas>=0.25->seaborn) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist
-packages (from pandas>=0.25->seaborn) (2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packa
ges (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.17.0)
Requirement already satisfied: intel-openmp>=2024 in /usr/local/lib/python3.10/
dist-packages (from mkl->numpy!=1.24.0,>=1.17->seaborn) (2024.2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python3.10/dist-pa
ckages (from mkl->numpy!=1.24.0,>=1.17->seaborn) (2022.0.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python3.10/dist-pa
ckages (from tbb==2022.*->mkl->numpy!=1.24.0,>=1.17->seaborn) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in /usr/local/lib/python3.10/
dist-packages (from mkl_umath->numpy!=1.24.0,>=1.17->seaborn) (2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in /usr/local/lib/p
ython3.10/dist-packages (from intel-openmp>=2024->mkl->numpy!=1.24.0,>=1.17->se
aborn) (2024.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
pip install split-folders
```

```
Collecting split-folders
  Downloading split_folders-0.5.1-py3-none-any.whl.metadata (6.2 kB)
Downloading split_folders-0.5.1-py3-none-any.whl (8.4 kB)
Installing collected packages: split-folders
Successfully installed split-folders-0.5.1
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
pip install numpy==1.23.4
```

```
Collecting numpy==1.23.4
  Downloading numpy-1.23.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_
64.whl.metadata (2.3 kB)
Downloading numpy-1.23.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (17.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.1/17.1 MB 92.7 MB/s eta 0:00:00:
00:0100:01
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.26.4
    Uninstalling numpy-1.26.4:
      Successfully uninstalled numpy-1.26.4
ERROR: pip's dependency resolver does not currently take into account all the p
ackages that are installed. This behaviour is the source of the following depen
dency conflicts.
albucore 0.0.19 requires numpy>=1.24.4, but you have numpy 1.23.4 which is inco
mpatible.
albumentations 1.4.20 requires numpy>=1.24.4, but you have numpy 1.23.4 which i
s incompatible.
bayesian-optimization 2.0.3 requires numpy>=1.25, but you have numpy 1.23.4 whi
ch is incompatible.
bigframes 1.29.0 requires numpy>=1.24.0, but you have numpy 1.23.4 which is inc
ompatible.
chex 0.1.88 requires numpy>=1.24.1, but you have numpy 1.23.4 which is incompat
ible.
featuretools 1.31.0 requires numpy>=1.25.0, but you have numpy 1.23.4 which is
incompatible.
jax 0.4.33 requires numpy>=1.24, but you have numpy 1.23.4 which is incompatibl
e.
jaxlib 0.4.33 requires numpy>=1.24, but you have numpy 1.23.4 which is incompat
ible.
langchain 0.3.12 requires async-timeout<5.0.0,>=4.0.0; python_version < "3.11",
but you have async-timeout 5.0.1 which is incompatible.
mizani 0.13.1 requires numpy>=1.23.5, but you have numpy 1.23.4 which is incomp
atible.
mkl-fft 1.3.8 requires numpy<1.27.0,>=1.26.4, but you have numpy 1.23.4 which i
s incompatible.
mkl-random 1.2.4 requires numpy<1.27.0,>=1.26.4, but you have numpy 1.23.4 whic
h is incompatible.
mkl-umath 0.1.1 requires numpy<1.27.0,>=1.26.4, but you have numpy 1.23.4 which
is incompatible.
mlxtend 0.23.3 requires scikit-learn>=1.3.1, but you have scikit-learn 1.2.2 wh
ich is incompatible.
pandas-gbq 0.25.0 requires google-api-core<3.0.0dev,>=2.10.2, but you have goog
le-api-core 1.34.1 which is incompatible.
pandas-stubs 2.2.2.240909 requires numpy>=1.23.5, but you have numpy 1.23.4 whi
ch is incompatible.
plotnine 0.14.4 requires matplotlib>=3.8.0, but you have matplotlib 3.7.5 which
is incompatible.
plotnine 0.14.4 requires numpy>=1.23.5, but you have numpy 1.23.4 which is inco
mpatible.
pyldavis 3.4.1 requires numpy>=1.24.2, but you have numpy 1.23.4 which is incom
patible.
pymc 5.19.1 requires numpy>=1.25.0, but you have numpy 1.23.4 which is incompat
ible.
scikit-image 0.25.0 requires numpy>=1.24, but you have numpy 1.23.4 which is in
compatible.
```

```
tensorflow 2.17.1 requires numpy<2.0.0,>=1.23.5; python_version <= "3.11", but
you have numpy 1.23.4 which is incompatible.
tensorflow-decision-forests 1.10.0 requires tensorflow==2.17.0, but you have te
nsorflow 2.17.1 which is incompatible.
woodwork 0.31.0 requires numpy>=1.25.0, but you have numpy 1.23.4 which is inco
mpatible.
xarray 2024.11.0 requires numpy>=1.24, but you have numpy 1.23.4 which is incom
patible.
Successfully installed numpy-1.23.4
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
import numpy as np
import os
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import backend as K
from tensorflow.keras import regularizers
from tensorflow.keras.layers import *
from tensorflow.keras.callbacks import *
from tensorflow.keras.optimizers import *
from tensorflow.keras.models import load_model, Model
from tensorflow.python.keras.utils import conv_utils
from tensorflow.keras import regularizers, constraints, initializers
import pandas as pd
from tensorflow.keras.layers import GlobalAveragePooling2D
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report

tf.random.set_seed(42)
np.random.seed(42)
```

In [5]:
```
train_path = ("/kaggle/input/brain-tumor-mri-dataset/Training")
val_path = ("/kaggle/input/brain-tumor-mri-dataset/Testing")
```

In [6]:
```
batch_size = 32
img_height = 128
img_width = 128
no_of_clases = 4
clsses_name = ['glioma', 'meningioma', 'notumor', 'pituitary']
input_shape = (img_height, img_width, 3)
```

In [7]:
```
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    shuffle=True,
    class_mode='categorical'
)

validation_generator = val_datagen.flow_from_directory(
    val_path,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    shuffle=False,  # No need to shuffle validation data
    class_mode='categorical'
)
```

```
Found 5712 images belonging to 4 classes.
Found 1311 images belonging to 4 classes.
```

In [8]:
```
print('train classes')
print(train_generator.class_indices)
print('validation classes')
print(validation_generator.class_indices)
```

```
train classes
{'glioma': 0, 'meningioma': 1, 'notumor': 2, 'pituitary': 3}
validation classes
{'glioma': 0, 'meningioma': 1, 'notumor': 2, 'pituitary': 3}
```

In [9]:
```
model = tf.keras.models.Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(no_of_clases, activation='softmax')  # Output layer
])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

In [10]:
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	P
conv2d (Conv2D)	(None, 126, 126, 32)	
max_pooling2d (MaxPooling2D)	(None, 63, 63, 32)	
conv2d_1 (Conv2D)	(None, 61, 61, 64)	
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 64)	
conv2d_2 (Conv2D)	(None, 28, 28, 128)	
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 128)	
flatten (Flatten)	(None, 25088)	
dense (Dense)	(None, 128)	3,2
dropout (Dropout)	(None, 128)	
dense_1 (Dense)	(None, 4)	

Total params: 3,305,156 (12.61 MB)

Trainable params: 3,305,156 (12.61 MB)

Non-trainable params: 0 (0.00 B)

In [11]:
```
epochs = 10
history = model.fit(
    train_generator,
    steps_per_epoch=len(train_generator),
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=len(validation_generator)
)
```

```
Epoch 1/10

/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dat
aset_adapter.py:122: UserWarning: Your `PyDataset` class should call `super()._
_init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_m
ultiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as t
hey will be ignored.
  self._warn_if_super_not_called()

179/179 ━━━━━━━━━━━━━━━━━━━━ 58s 289ms/step - accuracy: 0.4549 - loss: 1.1587 -
val_accuracy: 0.4630 - val_loss: 2.0576
Epoch 2/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 93us/step - accuracy: 0.0000e+00 - loss: 0.0000
e+00
Epoch 3/10

/usr/lib/python3.10/contextlib.py:153: UserWarning: Your input ran out of data;
interrupting training. Make sure that your dataset or generator can generate at
least `steps_per_epoch * epochs` batches. You may need to use the `.repeat()` f
unction when building your dataset.
  self.gen.throw(typ, value, traceback)

179/179 ━━━━━━━━━━━━━━━━━━━━ 32s 171ms/step - accuracy: 0.6867 - loss: 0.7860 -
val_accuracy: 0.7162 - val_loss: 0.6928
Epoch 4/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 63us/step - accuracy: 0.0000e+00 - loss: 0.0000
e+00
Epoch 5/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 32s 175ms/step - accuracy: 0.7363 - loss: 0.6905 -
val_accuracy: 0.5858 - val_loss: 1.2726
Epoch 6/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 62us/step - accuracy: 0.0000e+00 - loss: 0.0000
e+00
Epoch 7/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 32s 170ms/step - accuracy: 0.7663 - loss: 0.6190 -
val_accuracy: 0.7811 - val_loss: 0.5902
Epoch 8/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 67us/step - accuracy: 0.0000e+00 - loss: 0.0000
e+00
Epoch 9/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 33s 180ms/step - accuracy: 0.7718 - loss: 0.6066 -
val_accuracy: 0.5820 - val_loss: 1.2488
Epoch 10/10
179/179 ━━━━━━━━━━━━━━━━━━━━ 0s 48us/step - accuracy: 0.0000e+00 - loss: 0.0000
e+00
```

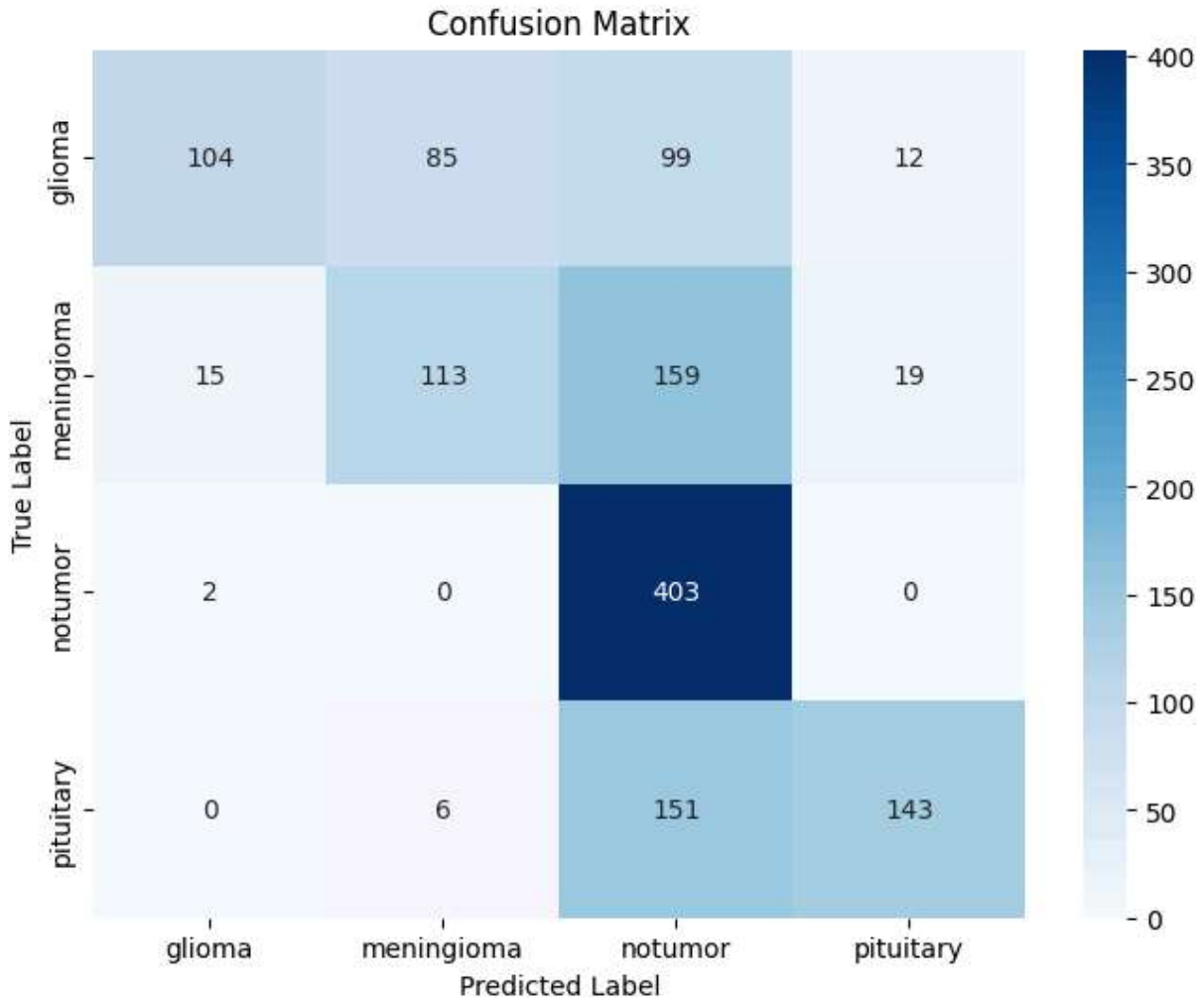
In [12]:
```
loss, accuracy = model.evaluate(validation_generator, steps=len(validation_genera
print('Validation accuracy:', accuracy)
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 3s 67ms/step - accuracy: 0.4610 - loss: 1.6857
Validation accuracy: 0.5819984674453735
```

In [13]:
```
# Classification Report and Confusion Matrix
y_pred = model.predict(validation_generator, steps=len(validation_generator))
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = validation_generator.classes
class_names = list(validation_generator.class_indices.keys())

print('\nClassification Report:\n', classification_report(y_true, y_pred_classes,
```

```
41/41 ━━━━━━━━━━━━━━━━━━━━ 3s 67ms/step

Classification Report:
               precision    recall  f1-score   support

      glioma       0.86      0.35      0.49       300
  meningioma       0.55      0.37      0.44       306
     notumor       0.50      1.00      0.66       405
   pituitary       0.82      0.48      0.60       300

    accuracy                           0.58      1311
   macro avg       0.68      0.55      0.55      1311
weighted avg       0.67      0.58      0.56      1311
```

In [14]:
```
cm = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, ytick
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



In [ ]:

In [1]:
```
pip install seaborn
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packag
es (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in /usr/local/lib/python3.1
0/dist-packages (from seaborn) (1.26.4)
Requirement already satisfied: pandas>=0.25 in /usr/local/lib/python3.10/dist-p
ackages (from seaborn) (2.2.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in /usr/local/lib/python
3.10/dist-packages (from seaborn) (3.7.5)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/di
st-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-p
ackages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/d
ist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.55.3)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/d
ist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (24.2)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib!=3.6.1,>=3.1->seaborn) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/di
st-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.1
0/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.9.0.post0)
Requirement already satisfied: mkl_fft in /usr/local/lib/python3.10/dist-packag
es (from numpy!=1.24.0,>=1.17->seaborn) (1.3.8)
Requirement already satisfied: mkl_random in /usr/local/lib/python3.10/dist-pac
kages (from numpy!=1.24.0,>=1.17->seaborn) (1.2.4)
Requirement already satisfied: mkl_umath in /usr/local/lib/python3.10/dist-pack
ages (from numpy!=1.24.0,>=1.17->seaborn) (0.1.1)
Requirement already satisfied: mkl in /usr/local/lib/python3.10/dist-packages
(from numpy!=1.24.0,>=1.17->seaborn) (2025.0.1)
Requirement already satisfied: tbb4py in /usr/local/lib/python3.10/dist-package
s (from numpy!=1.24.0,>=1.17->seaborn) (2022.0.0)
Requirement already satisfied: mkl-service in /usr/local/lib/python3.10/dist-pa
ckages (from numpy!=1.24.0,>=1.17->seaborn) (2.4.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-p
ackages (from pandas>=0.25->seaborn) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist
-packages (from pandas>=0.25->seaborn) (2025.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packa
ges (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.17.0)
Requirement already satisfied: intel-openmp>=2024 in /usr/local/lib/python3.10/
dist-packages (from mkl->numpy!=1.24.0,>=1.17->seaborn) (2024.2.0)
Requirement already satisfied: tbb==2022.* in /usr/local/lib/python3.10/dist-pa
ckages (from mkl->numpy!=1.24.0,>=1.17->seaborn) (2022.0.0)
Requirement already satisfied: tcmlib==1.* in /usr/local/lib/python3.10/dist-pa
ckages (from tbb==2022.*->mkl->numpy!=1.24.0,>=1.17->seaborn) (1.2.0)
Requirement already satisfied: intel-cmplr-lib-rt in /usr/local/lib/python3.10/
dist-packages (from mkl_umath->numpy!=1.24.0,>=1.17->seaborn) (2024.2.0)
Requirement already satisfied: intel-cmplr-lib-ur==2024.2.0 in /usr/local/lib/p
ython3.10/dist-packages (from intel-openmp>=2024->mkl->numpy!=1.24.0,>=1.17->se
aborn) (2024.2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [2]:
```
pip install numpy==1.23.4
```

```
Collecting numpy==1.23.4
  Downloading numpy-1.23.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_
64.whl.metadata (2.3 kB)
Downloading numpy-1.23.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (17.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.1/17.1 MB 75.8 MB/s eta 0:00:00:
00:0100:01
Installing collected packages: numpy
  Attempting uninstall: numpy
    Found existing installation: numpy 1.26.4
    Uninstalling numpy-1.26.4:
      Successfully uninstalled numpy-1.26.4
ERROR: pip's dependency resolver does not currently take into account all the p
ackages that are installed. This behaviour is the source of the following depen
dency conflicts.
albucore 0.0.19 requires numpy>=1.24.4, but you have numpy 1.23.4 which is inco
mpatible.
albumentations 1.4.20 requires numpy>=1.24.4, but you have numpy 1.23.4 which i
s incompatible.
bayesian-optimization 2.0.3 requires numpy>=1.25, but you have numpy 1.23.4 whi
ch is incompatible.
bigframes 1.29.0 requires numpy>=1.24.0, but you have numpy 1.23.4 which is inc
ompatible.
chex 0.1.88 requires numpy>=1.24.1, but you have numpy 1.23.4 which is incompat
ible.
featuretools 1.31.0 requires numpy>=1.25.0, but you have numpy 1.23.4 which is
incompatible.
jax 0.4.33 requires numpy>=1.24, but you have numpy 1.23.4 which is incompatibl
e.
jaxlib 0.4.33 requires numpy>=1.24, but you have numpy 1.23.4 which is incompat
ible.
langchain 0.3.12 requires async-timeout<5.0.0,>=4.0.0; python_version < "3.11",
but you have async-timeout 5.0.1 which is incompatible.
mizani 0.13.1 requires numpy>=1.23.5, but you have numpy 1.23.4 which is incomp
atible.
mkl-fft 1.3.8 requires numpy<1.27.0,>=1.26.4, but you have numpy 1.23.4 which i
s incompatible.
mkl-random 1.2.4 requires numpy<1.27.0,>=1.26.4, but you have numpy 1.23.4 whic
h is incompatible.
mkl-umath 0.1.1 requires numpy<1.27.0,>=1.26.4, but you have numpy 1.23.4 which
is incompatible.
mlxtend 0.23.3 requires scikit-learn>=1.3.1, but you have scikit-learn 1.2.2 wh
ich is incompatible.
pandas-gbq 0.25.0 requires google-api-core<3.0.0dev,>=2.10.2, but you have goog
le-api-core 1.34.1 which is incompatible.
pandas-stubs 2.2.2.240909 requires numpy>=1.23.5, but you have numpy 1.23.4 whi
ch is incompatible.
plotnine 0.14.4 requires matplotlib>=3.8.0, but you have matplotlib 3.7.5 which
is incompatible.
plotnine 0.14.4 requires numpy>=1.23.5, but you have numpy 1.23.4 which is inco
mpatible.
pyldavis 3.4.1 requires numpy>=1.24.2, but you have numpy 1.23.4 which is incom
patible.
pymc 5.19.1 requires numpy>=1.25.0, but you have numpy 1.23.4 which is incompat
ible.
scikit-image 0.25.0 requires numpy>=1.24, but you have numpy 1.23.4 which is in
compatible.
```

```
tensorflow 2.17.1 requires numpy<2.0.0,>=1.23.5; python_version <= "3.11", but
you have numpy 1.23.4 which is incompatible.
tensorflow-decision-forests 1.10.0 requires tensorflow==2.17.0, but you have te
nsorflow 2.17.1 which is incompatible.
woodwork 0.31.0 requires numpy>=1.25.0, but you have numpy 1.23.4 which is inco
mpatible.
xarray 2024.11.0 requires numpy>=1.24, but you have numpy 1.23.4 which is incom
patible.
Successfully installed numpy-1.23.4
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
import numpy as np
import os
import tensorflow as tf
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
print(tf.__version__)
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import backend as C
from tensorflow.keras import regularizers
from tensorflow.keras.layers import *
from tensorflow.keras.callbacks import *
from tensorflow.keras.optimizers import *
from tensorflow.keras.models import load_model,Model
from tensorflow.python.keras.utils import conv_utils
from tensorflow.keras import regularizers, constraints, initializers
import pandas as pd
from tensorflow.keras.layers import GlobalAveragePooling2D
#import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns
```

```
2.17.1
```

In [4]:
```
train_path=("/kaggle/input/cards-image-datasetclassification/train")
val_path=("/kaggle/input/cards-image-datasetclassification/valid")
test_path=("/kaggle/input/cards-image-datasetclassification/test")
batch_size = 32
img_height = 640
img_width = 640
no_of_clases = 53
classes_name = [
    'AS', '2S', '3S', '4S', '5S', '6S', '7S', '8S', '9S', '10S',
    'JS', 'QS', 'KS',  # Spades
    'AH', '2H', '3H', '4H', '5H', '6H', '7H', '8H', '9H', '10H',
    'JH', 'QH', 'KH',  # Hearts
    'AD', '2D', '3D', '4D', '5D', '6D', '7D', '8D', '9D', '10D',
    'JD', 'QD', 'KD',  # Diamonds
    'AC', '2C', '3C', '4C', '5C', '6C', '7C', '8C', '9C', '10C',
    'JC', 'QC', 'KC',  # Clubs
    'BACK'              # Card Back
]

input_shape = (img_height,img_width,1)

datagen = ImageDataGenerator(rescale = 1./255,featurewise_center=True,horizontal_

train_generator = datagen.flow_from_directory(train_path,target_size=(img_height,

validation_generator = datagen.flow_from_directory(val_path,target_size=(img_heig

print('train classes')
print(train_generator.class_indices)
print('train labels')
print(train_generator.labels)
print('validation clases')
print(validation_generator.class_indices)
print('validation labels')
print(validation_generator.labels)
```

```
Found 7624 images belonging to 53 classes.
Found 265 images belonging to 53 classes.
train classes
{'ace of clubs': 0, 'ace of diamonds': 1, 'ace of hearts': 2, 'ace of spades':
3, 'eight of clubs': 4, 'eight of diamonds': 5, 'eight of hearts': 6, 'eight of
spades': 7, 'five of clubs': 8, 'five of diamonds': 9, 'five of hearts': 10, 'f
ive of spades': 11, 'four of clubs': 12, 'four of diamonds': 13, 'four of heart
s': 14, 'four of spades': 15, 'jack of clubs': 16, 'jack of diamonds': 17, 'jac
k of hearts': 18, 'jack of spades': 19, 'joker': 20, 'king of clubs': 21, 'king
of diamonds': 22, 'king of hearts': 23, 'king of spades': 24, 'nine of clubs':
25, 'nine of diamonds': 26, 'nine of hearts': 27, 'nine of spades': 28, 'queen
of clubs': 29, 'queen of diamonds': 30, 'queen of hearts': 31, 'queen of spade
s': 32, 'seven of clubs': 33, 'seven of diamonds': 34, 'seven of hearts': 35,
'seven of spades': 36, 'six of clubs': 37, 'six of diamonds': 38, 'six of heart
s': 39, 'six of spades': 40, 'ten of clubs': 41, 'ten of diamonds': 42, 'ten of
hearts': 43, 'ten of spades': 44, 'three of clubs': 45, 'three of diamonds': 4
6, 'three of hearts': 47, 'three of spades': 48, 'two of clubs': 49, 'two of di
amonds': 50, 'two of hearts': 51, 'two of spades': 52}
train labels
[ 0  0  0 ... 52 52 52]
validation clases
{'ace of clubs': 0, 'ace of diamonds': 1, 'ace of hearts': 2, 'ace of spades':
3, 'eight of clubs': 4, 'eight of diamonds': 5, 'eight of hearts': 6, 'eight of
spades': 7, 'five of clubs': 8, 'five of diamonds': 9, 'five of hearts': 10, 'f
ive of spades': 11, 'four of clubs': 12, 'four of diamonds': 13, 'four of heart
s': 14, 'four of spades': 15, 'jack of clubs': 16, 'jack of diamonds': 17, 'jac
k of hearts': 18, 'jack of spades': 19, 'joker': 20, 'king of clubs': 21, 'king
of diamonds': 22, 'king of hearts': 23, 'king of spades': 24, 'nine of clubs':
25, 'nine of diamonds': 26, 'nine of hearts': 27, 'nine of spades': 28, 'queen
of clubs': 29, 'queen of diamonds': 30, 'queen of hearts': 31, 'queen of spade
s': 32, 'seven of clubs': 33, 'seven of diamonds': 34, 'seven of hearts': 35,
'seven of spades': 36, 'six of clubs': 37, 'six of diamonds': 38, 'six of heart
s': 39, 'six of spades': 40, 'ten of clubs': 41, 'ten of diamonds': 42, 'ten of
hearts': 43, 'ten of spades': 44, 'three of clubs': 45, 'three of diamonds': 4
6, 'three of hearts': 47, 'three of spades': 48, 'two of clubs': 49, 'two of di
amonds': 50, 'two of hearts': 51, 'two of spades': 52}
validation labels
[ 0  0  0  0  0  1  1  1  1  1  2  2  2  2  2  3  3  3  3  3  4  4  4  4
  4  5  5  5  5  5  6  6  6  6  6  7  7  7  7  7  8  8  8  8  8  9  9  9
  9  9 10 10 10 10 10 11 11 11 11 11 12 12 12 12 12 13 13 13 13 13 14 14
 14 14 14 15 15 15 15 15 16 16 16 16 16 17 17 17 17 17 18 18 18 18 18 19
 19 19 19 19 20 20 20 20 20 21 21 21 21 21 22 22 22 22 22 23 23 23 23 23
 24 24 24 24 24 25 25 25 25 25 26 26 26 26 26 27 27 27 27 27 28 28 28 28
 28 29 29 29 29 29 30 30 30 30 30 31 31 31 31 31 32 32 32 32 32 33 33 33
 33 33 34 34 34 34 34 35 35 35 35 35 36 36 36 36 36 37 37 37 37 37 38 38
 38 38 38 39 39 39 39 39 40 40 40 40 40 41 41 41 41 41 42 42 42 42 42 43
 43 43 43 43 44 44 44 44 44 45 45 45 45 45 46 46 46 46 46 47 47 47 47 47
 48 48 48 48 48 49 49 49 49 49 50 50 50 50 50 51 51 51 51 51 52 52 52 52
 52]
```

In [5]:
```
pretrained_model = tf.keras.applications.ResNet152V2(
                    input_shape=(640, 640, 3),
                    include_top=False,
                    weights='imagenet',
                    pooling='avg')

pretrained_model.trainable = False
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicati
ons/resnet/resnet152v2_weights_tf_dim_ordering_tf_kernels_notop.h5 (https://sto
rage.googleapis.com/tensorflow/keras-applications/resnet/resnet152v2_weights_tf
_dim_ordering_tf_kernels_notop.h5)
234545216/234545216 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

In [6]:
```
inputs = pretrained_model.input
outputs1 = tf.keras.layers.Dense(1024, activation='relu')(pretrained_model.output
dropout = tf.keras.layers.Dropout(0.5)(outputs1)
output2 = tf.keras.layers.Dense(1024,activation = 'relu')(dropout)
dropout2 = tf.keras.layers.Dropout(0.5)(output2)
outputs = tf.keras.layers.Dense(53,activation = 'softmax')(dropout2)
model = tf.keras.Model(inputs, outputs)
print(model.summary())
```

```
Model: "functional"
```

Layer (type)	Output Shape	Param #	Connec
input_layer (InputLayer)	(None, 640, 640, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 646, 646, 3)	0	input_
conv1_conv (Conv2D)	(None, 320, 320, 64)	9,472	conv1_
pool1_pad (ZeroPadding2D)	(None, 322, 322, 64)	0	conv1_
pool1_pool (MaxPooling2D)	(None, 160, 160, 64)	0	pool1_
conv2_block1_preact_bn (BatchNormalization)	(None, 160, 160, 64)	256	pool1_
conv2_block1_preact_relu	(None, 160, 160, 64)	0	conv2

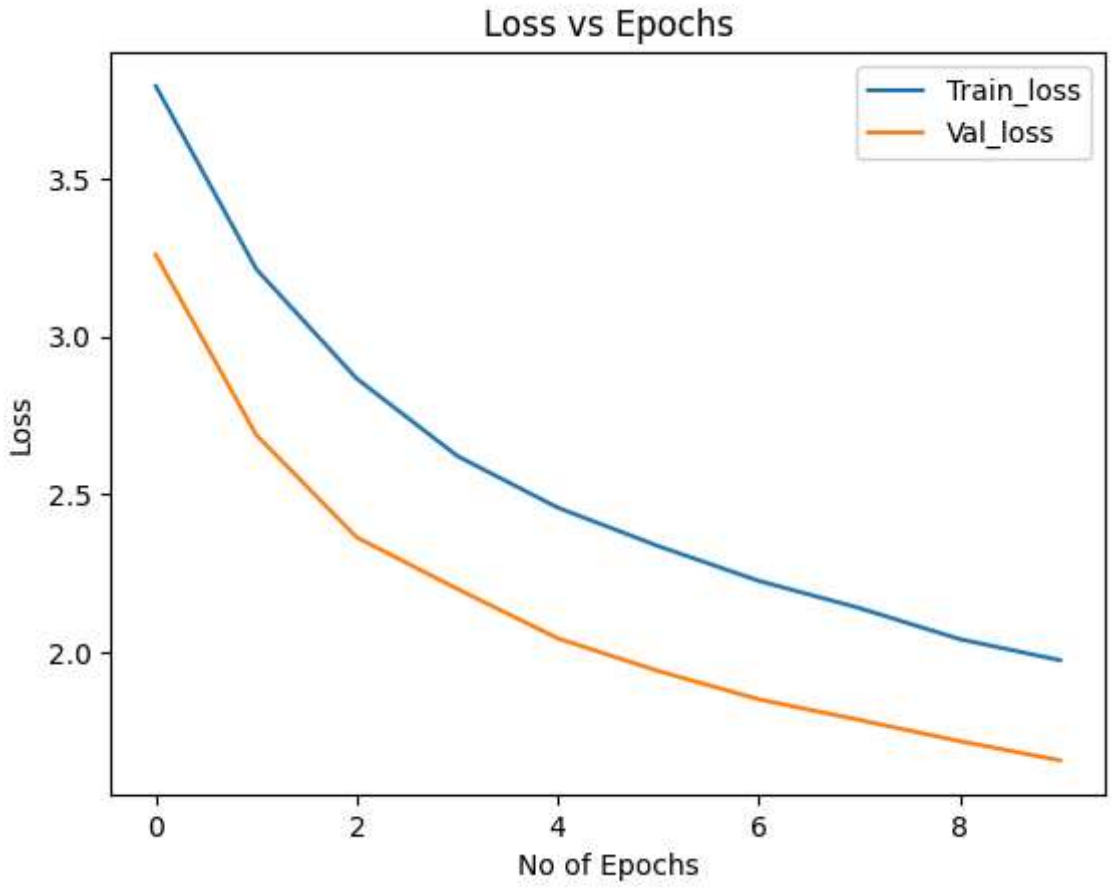
In [7]:
```
model.compile(optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, decay=1e-5),
```

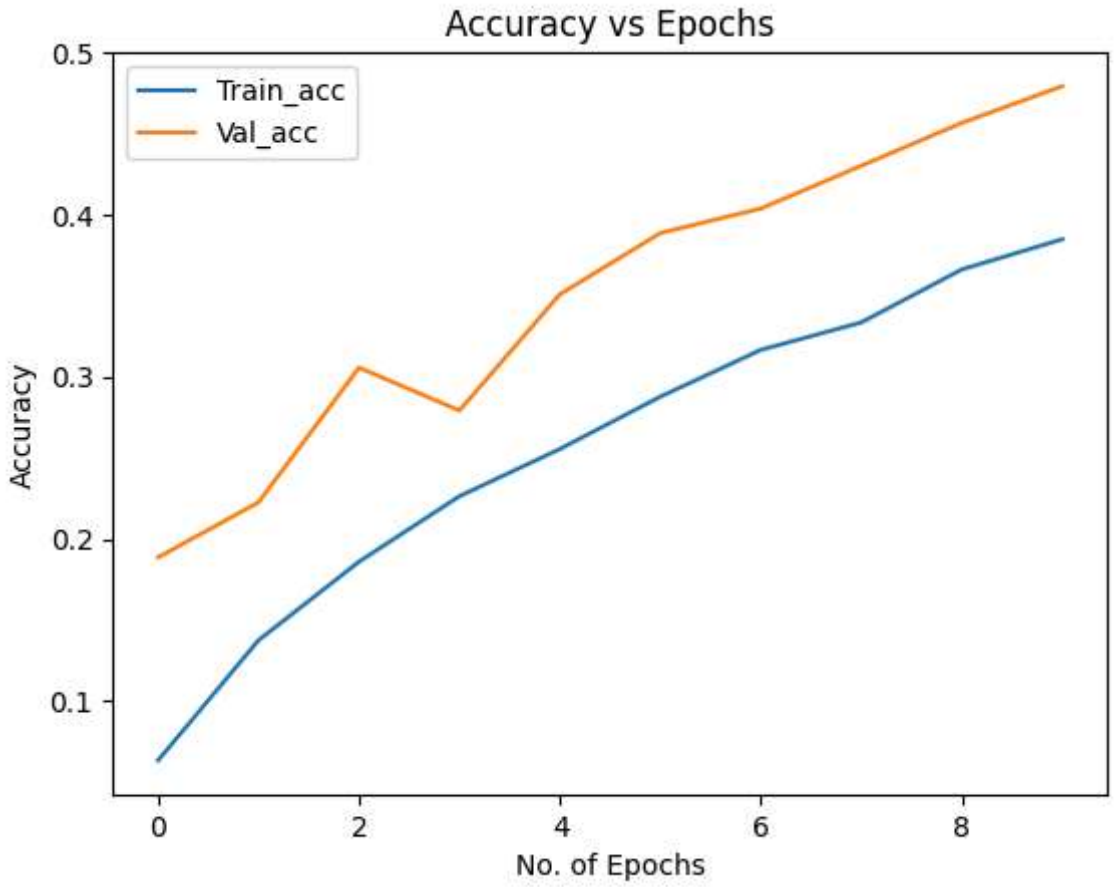
```
/usr/local/lib/python3.10/dist-packages/keras/src/optimizers/base_optimizer.py:
33: UserWarning: Argument `decay` is no longer supported and will be ignored.
  warnings.warn(
```

In [8]:
```
history = model.fit(train_generator,validation_data=validation_generator,epochs=1
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/legacy/preprocessing/image.p
y:1263: UserWarning: This ImageDataGenerator specifies `featurewise_center`, bu
t it hasn't been fit on any training data. Fit it first by calling `.fit(numpy_
data)`.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/keras/src/legacy/preprocessing/image.p
y:1273: UserWarning: This ImageDataGenerator specifies `featurewise_std_normali
zation`, but it hasn't been fit on any training data. Fit it first by calling
`.fit(numpy_data)`.
  warnings.warn(

Epoch 1/10

/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dat
aset_adapter.py:122: UserWarning: Your `PyDataset` class should call `super()._
_init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_m
ultiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as t
hey will be ignored.
  self._warn_if_super_not_called()

239/239 ━━━━━━━━━━━━━━━━━━━━ 467s 2s/step - accuracy: 0.0396 - loss: 3.9823 - v
al_accuracy: 0.1887 - val_loss: 3.2589
Epoch 2/10
239/239 ━━━━━━━━━━━━━━━━━━━━ 382s 2s/step - accuracy: 0.1206 - loss: 3.3266 - v
al_accuracy: 0.2226 - val_loss: 2.6875
Epoch 3/10
239/239 ━━━━━━━━━━━━━━━━━━━━ 382s 2s/step - accuracy: 0.1797 - loss: 2.9279 - v
al_accuracy: 0.3057 - val_loss: 2.3631
Epoch 4/10
239/239 ━━━━━━━━━━━━━━━━━━━━ 382s 2s/step - accuracy: 0.2289 - loss: 2.6432 - v
al_accuracy: 0.2792 - val_loss: 2.1997
Epoch 5/10
239/239 ━━━━━━━━━━━━━━━━━━━━ 382s 2s/step - accuracy: 0.2498 - loss: 2.5113 - v
al_accuracy: 0.3509 - val_loss: 2.0429
Epoch 6/10
239/239 ━━━━━━━━━━━━━━━━━━━━ 383s 2s/step - accuracy: 0.2767 - loss: 2.3602 - v
al_accuracy: 0.3887 - val_loss: 1.9393
Epoch 7/10
239/239 ━━━━━━━━━━━━━━━━━━━━ 382s 2s/step - accuracy: 0.3103 - loss: 2.2509 - v
al_accuracy: 0.4038 - val_loss: 1.8496
Epoch 8/10
239/239 ━━━━━━━━━━━━━━━━━━━━ 382s 2s/step - accuracy: 0.3399 - loss: 2.1333 - v
al_accuracy: 0.4302 - val_loss: 1.7842
Epoch 9/10
239/239 ━━━━━━━━━━━━━━━━━━━━ 381s 2s/step - accuracy: 0.3605 - loss: 2.0537 - v
al_accuracy: 0.4566 - val_loss: 1.7169
Epoch 10/10
239/239 ━━━━━━━━━━━━━━━━━━━━ 382s 2s/step - accuracy: 0.3826 - loss: 1.9712 - v
al_accuracy: 0.4792 - val_loss: 1.6559
```

In [9]:
```
plt.plot(history.history['loss'],label='Train_loss')
plt.plot(history.history['val_loss'],label='Val_loss')
plt.legend()
plt.xlabel('No of Epochs')
plt.ylabel('Loss')
plt.title('Loss vs Epochs')
plt.show()


plt.plot(history.history['accuracy'],label = 'Train_acc')
plt.plot(history.history['val_accuracy'],label = 'Val_acc')
plt.legend()
plt.xlabel('No. of Epochs')
plt.ylabel("Accuracy")
plt.title("Accuracy vs Epochs")
plt.show()
```





In [10]:
```
test_data = ImageDataGenerator(rescale=1./255)
test_generator = test_data.flow_from_directory(test_path, target_size=(640, 640),
```

```
Found 265 images belonging to 53 classes.
```

In [11]:
```
predictions = np.argmax(model.predict(test_generator), axis=1)
matrix = confusion_matrix(test_generator.labels, predictions)
report = classification_report(test_generator.labels, predictions, zero_division=
print("Classification Report:\n", report)
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 26s 2s/step
Classification Report:
```

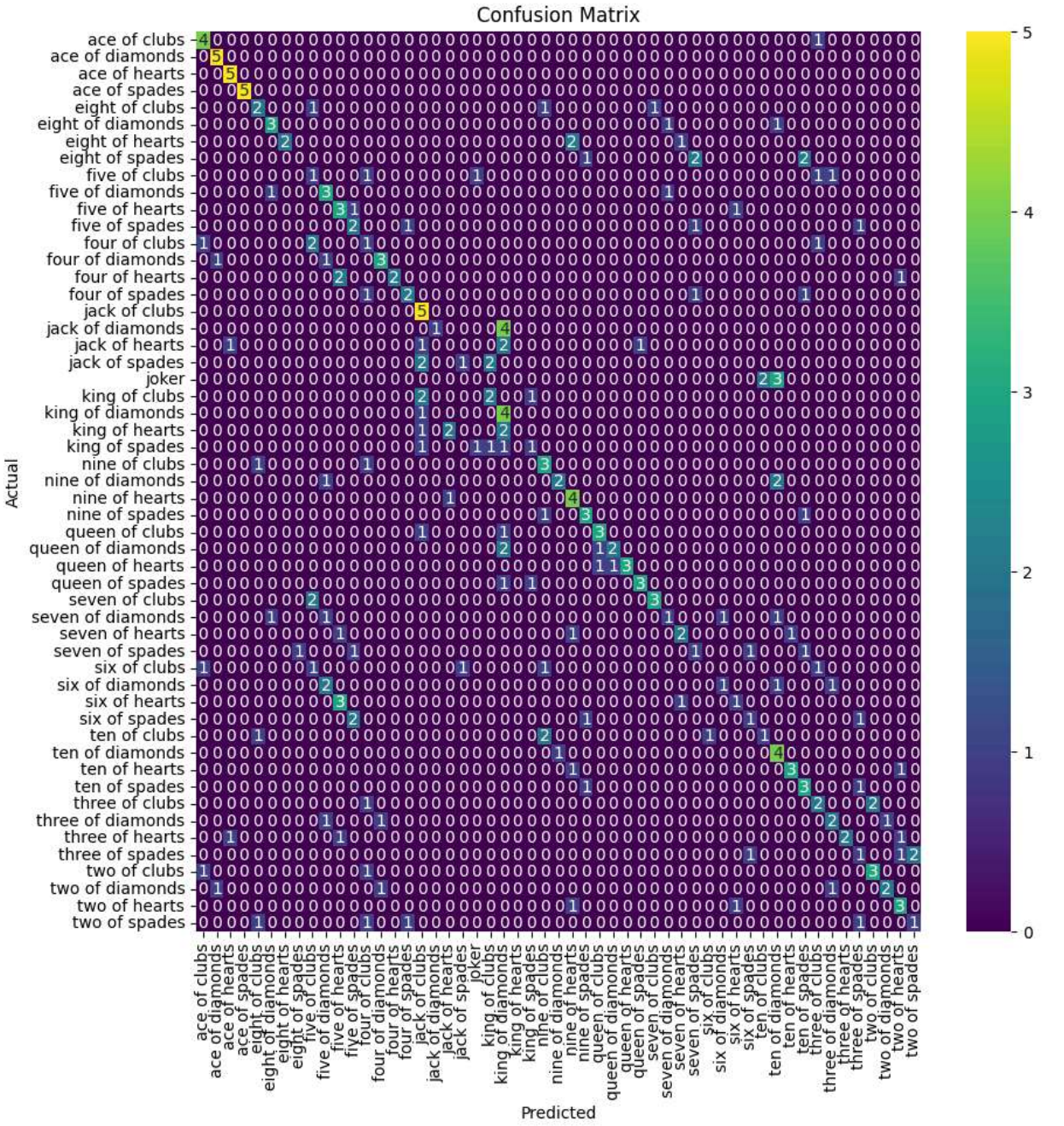
	precision	recall	f1-score	support
0	0.57	0.80	0.67	5
1	0.71	1.00	0.83	5
2	0.71	1.00	0.83	5
3	1.00	1.00	1.00	5
4	0.40	0.40	0.40	5
5	0.60	0.60	0.60	5
6	1.00	0.40	0.57	5
7	0.00	0.00	0.00	5
8	0.14	0.20	0.17	5
9	0.33	0.60	0.43	5
10	0.30	0.60	0.40	5
11	0.33	0.40	0.36	5
12	0.14	0.20	0.17	5
13	0.60	0.60	0.60	5
14	1.00	0.40	0.57	5
15	0.50	0.40	0.44	5
16	0.36	1.00	0.53	5
17	1.00	0.20	0.33	5
18	0.00	0.00	0.00	5
19	0.50	0.20	0.29	5
20	0.00	0.00	0.00	5
21	0.40	0.40	0.40	5
22	0.24	0.80	0.36	5
23	0.00	0.00	0.00	5
24	0.33	0.20	0.25	5
25	0.38	0.60	0.46	5
26	0.67	0.40	0.50	5
27	0.44	0.80	0.57	5
28	0.50	0.60	0.55	5
29	0.60	0.60	0.60	5
30	0.67	0.40	0.50	5
31	1.00	0.60	0.75	5
32	0.75	0.60	0.67	5
33	0.75	0.60	0.67	5
34	0.33	0.20	0.25	5
35	0.50	0.40	0.44	5
36	0.20	0.20	0.20	5
37	0.00	0.00	0.00	5
38	0.50	0.20	0.29	5
39	0.33	0.20	0.25	5
40	0.33	0.20	0.25	5
41	0.33	0.20	0.25	5
42	0.33	0.80	0.47	5
43	0.75	0.60	0.67	5
44	0.38	0.60	0.46	5
45	0.33	0.40	0.36	5
46	0.40	0.40	0.40	5
47	1.00	0.40	0.57	5
48	0.20	0.20	0.20	5
49	0.60	0.60	0.60	5
50	0.67	0.40	0.50	5
51	0.43	0.60	0.50	5
52	0.33	0.20	0.25	5

```
              accuracy                           0.44       265
             macro avg       0.47      0.44      0.42       265
          weighted avg       0.47      0.44      0.42       265
```

In [12]:
```
pred = model.predict(test_generator,verbose=1)
test_generator_indices = np.argmax(pred,axis=1)
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 13s 2s/step
```

In [14]:
```
fig = plt.figure(figsize=(10,10))
sns.heatmap(matrix, annot=True, cmap='viridis')
plt.xticks(ticks=np.arange(53) + 0.5, labels=test_generator.class_indices, rotati
plt.yticks(ticks=np.arange(53) + 0.5, labels=test_generator.class_indices, rotati
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
fig.savefig("Confusion Matrix",dpi=700)
```



In [ ]: