**Setting up the Environment**

```
!pip install tensorflow keras flask kaggle
```

```
Requirement already satisfied: tensorflow in
/usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: keras in
/usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: flask in
/usr/local/lib/python3.10/dist-packages (2.2.5)
Requirement already satisfied: kaggle in
/usr/local/lib/python3.10/dist-packages (1.6.14)
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.25.2)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.0)
```

```
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.16,>=2.15 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: Werkzeug>=2.2.2 in
/usr/local/lib/python3.10/dist-packages (from flask) (3.0.3)
Requirement already satisfied: Jinja2>=3.0 in
/usr/local/lib/python3.10/dist-packages (from flask) (3.1.4)
Requirement already satisfied: itsdangerous>=2.0 in
/usr/local/lib/python3.10/dist-packages (from flask) (2.2.0)
Requirement already satisfied: click>=8.0 in
/usr/local/lib/python3.10/dist-packages (from flask) (8.1.7)
Requirement already satisfied: certifi>=2023.7.22 in
/usr/local/lib/python3.10/dist-packages (from kaggle) (2024.6.2)
Requirement already satisfied: python-dateutil in
/usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in
/usr/local/lib/python3.10/dist-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-
packages (from kaggle) (4.66.4)
Requirement already satisfied: python-slugify in
/usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in
/usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in
/usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0-
>tensorflow) (0.43.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.10/dist-packages (from Jinja2>=3.0->flask)
(2.1.5)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (2.27.0)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (3.6)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow) (0.7.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->kaggle)
(3.3.2)
```

```
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.7)
Requirement already satisfied: webencodings in
/usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle)
(1.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.16,>=2.15->tensorflow) (5.3.3)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.16,>=2.15->tensorflow) (0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.16,>=2.15->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth-
oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (1.3.1)
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.6.0)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests-
oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5-
>tensorboard<2.16,>=2.15->tensorflow) (3.2.2)
```

```python
from google.colab import files

# This will prompt you to upload the kaggle.json file
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving kaggle.json to kaggle.json
```

```python
import os
import shutil

# Create the .kaggle directory if it doesn't exist
kaggle_dir = os.path.expanduser('~/.kaggle')
os.makedirs(kaggle_dir, exist_ok=True)

# Move the uploaded kaggle.json to the .kaggle directory
for filename in uploaded.keys():
    shutil.move(filename, os.path.join(kaggle_dir, filename))

# Set permissions for the kaggle.json file
os.chmod(os.path.join(kaggle_dir, 'kaggle.json'), 0o600)

!kaggle datasets download -d saailna/war-events-classification
```

```
Dataset URL: https://www.kaggle.com/datasets/saailna/war-events-
classification
License(s): MIT
war-events-classification.zip: Skipping, found more recently modified
local copy (use --force to force download)

!unzip /content/war-events-classification.zip

Archive:  /content/war-events-classification.zip
replace war_events/Combat/1.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename:

import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import img_to_array,
load_img

# Define paths
data_dir = '/content/war_events' # Ensure your images are in
subdirectories for each class

# Prepare dataset lists
image_paths = []
labels = []

# data_dir has subdirectories for each class
for class_name in os.listdir(data_dir):
    class_dir = os.path.join(data_dir, class_name)
    if os.path.isdir(class_dir):
        for img_name in os.listdir(class_dir):
            img_path = os.path.join(class_dir, img_name)
            image_paths.append(img_path)
            labels.append(class_name)

# Convert labels to numerical values
label_to_index = {label: idx for idx, label in enumerate(set(labels))}
labels = [label_to_index[label] for label in labels]

# Split the dataset
train_paths, val_paths, train_labels, val_labels =
train_test_split(image_paths, labels, test_size=0.2, stratify=labels)

# Data augmentation and preprocessing
def preprocess_image(img_path):
    img = load_img(img_path, target_size=(224, 224))
    img_array = img_to_array(img) / 255.0
    return img_array

def data_generator(paths, labels, batch_size, is_train):
    data_gen = ImageDataGenerator(
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest'
    ) if is_train else ImageDataGenerator()

    while True:
        for start in range(0, len(paths), batch_size):
            end = min(start + batch_size, len(paths))
            batch_paths = paths[start:end]
            batch_labels = labels[start:end]

            batch_images = np.array([preprocess_image(img_path) for
img_path in batch_paths])
            batch_labels = tf.keras.utils.to_categorical(batch_labels,
num_classes=len(label_to_index))

            if is_train:
                yield next(data_gen.flow(batch_images, batch_labels,
batch_size=batch_size))
            else:
                yield batch_images, batch_labels

batch_size = 32

train_gen = data_generator(train_paths, train_labels, batch_size,
is_train=True)
val_gen = data_generator(val_paths, val_labels, batch_size,
is_train=False)

# Load pre-trained ResNet50 model + higher level layers
base_model = ResNet50(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))

# Freeze convolutional layers
for layer in base_model.layers:
    layer.trainable = False

# Create a new model on top
model = Sequential([
    base_model,
    Flatten(),
    Dense(256, activation='relu'),
    Dense(len(label_to_index), activation='softmax')  # Adjust the
number of classes dynamically
])

model.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])

# Train the model
model.fit(
    train_gen,
    steps_per_epoch=len(train_paths) // batch_size,
    validation_data=val_gen,
    validation_steps=len(val_paths) // batch_size,
    epochs=20
)

# Save the model
model.save('war_lens_model_resnet50.h5')

Epoch 1/20
12/12 [==============================] - 139s 11s/step - loss: 15.9469
- accuracy: 0.2005 - val_loss: 4.5750 - val_accuracy: 0.1979
Epoch 2/20
12/12 [==============================] - 110s 9s/step - loss: 2.6478 -
accuracy: 0.3152 - val_loss: 1.6185 - val_accuracy: 0.4559
Epoch 3/20
12/12 [==============================] - 108s 9s/step - loss: 1.7286 -
accuracy: 0.3723 - val_loss: 1.4072 - val_accuracy: 0.3824
Epoch 4/20
12/12 [==============================] - 107s 9s/step - loss: 1.4118 -
accuracy: 0.3777 - val_loss: 1.3793 - val_accuracy: 0.4265
Epoch 5/20
12/12 [==============================] - 103s 9s/step - loss: 1.3041 -
accuracy: 0.4674 - val_loss: 1.2268 - val_accuracy: 0.4559
Epoch 6/20
12/12 [==============================] - 110s 9s/step - loss: 1.3147 -
accuracy: 0.4647 - val_loss: 1.4095 - val_accuracy: 0.3824
Epoch 7/20
12/12 [==============================] - 106s 9s/step - loss: 1.3526 -
accuracy: 0.4348 - val_loss: 1.4803 - val_accuracy: 0.3529
Epoch 8/20
12/12 [==============================] - 105s 9s/step - loss: 1.3010 -
accuracy: 0.4783 - val_loss: 1.2002 - val_accuracy: 0.4559
Epoch 9/20
12/12 [==============================] - 101s 8s/step - loss: 1.3379 -
accuracy: 0.4402 - val_loss: 1.1594 - val_accuracy: 0.5882
Epoch 10/20
12/12 [==============================] - ETA: 0s - loss: 1.2437 -
accuracy: 0.4810Epoch 11/20
```

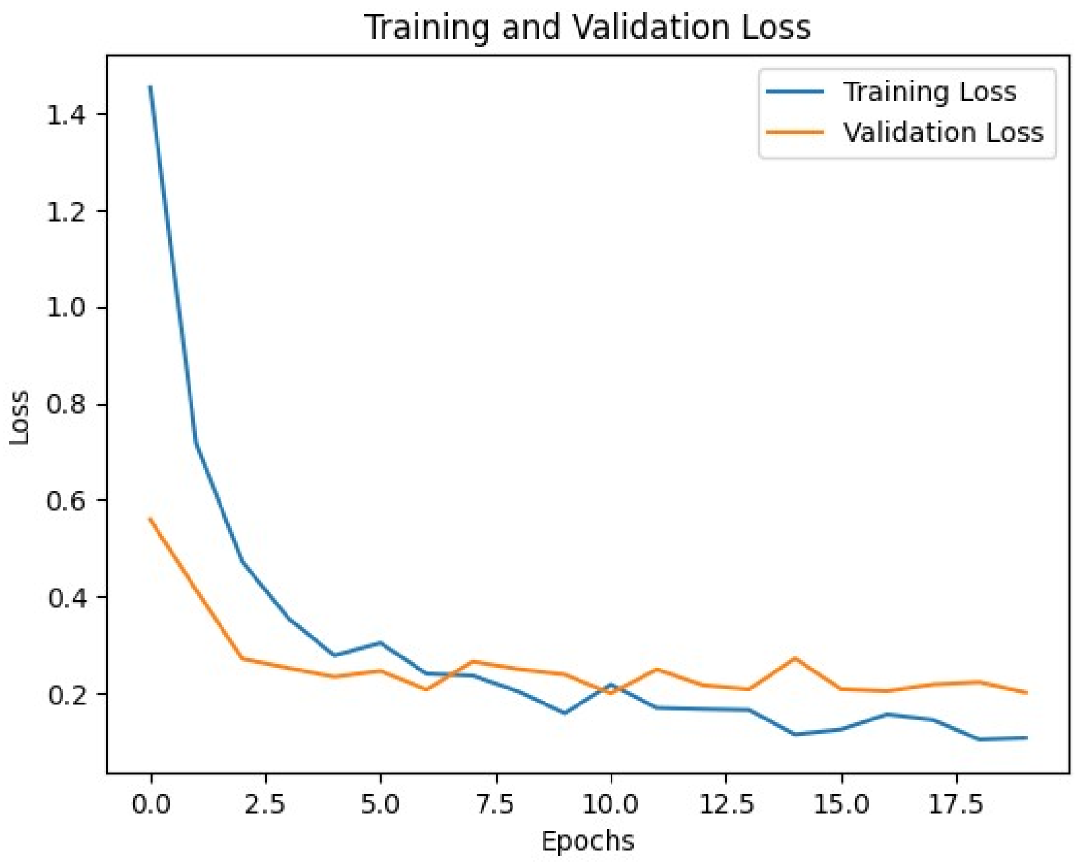
```
12/12 [==============================] - 99s 8s/step - loss: 1.2875 -
accuracy: 0.5054 - val_loss: 1.1221 - val_accuracy: 0.5147
Epoch 12/20
12/12 [==============================] - 98s 8s/step - loss: 1.2368 -
accuracy: 0.4538 - val_loss: 1.1342 - val_accuracy: 0.5588
Epoch 13/20
12/12 [==============================] - 102s 9s/step - loss: 1.2059 -
accuracy: 0.5109 - val_loss: 1.1484 - val_accuracy: 0.5588
Epoch 14/20
12/12 [==============================] - 107s 9s/step - loss: 1.1280 -
accuracy: 0.5547 - val_loss: 1.2937 - val_accuracy: 0.4853
Epoch 15/20
12/12 [==============================] - 103s 9s/step - loss: 1.2020 -
accuracy: 0.5217 - val_loss: 1.0901 - val_accuracy: 0.5882
Epoch 16/20
12/12 [==============================] - 106s 9s/step - loss: 1.1269 -
accuracy: 0.5625 - val_loss: 1.0758 - val_accuracy: 0.6029
Epoch 17/20
12/12 [==============================] - 104s 9s/step - loss: 1.1766 -
accuracy: 0.5217 - val_loss: 1.1638 - val_accuracy: 0.5294
Epoch 18/20
12/12 [==============================] - 100s 8s/step - loss: 1.2492 -
accuracy: 0.5272 - val_loss: 1.0006 - val_accuracy: 0.5735
Epoch 19/20
12/12 [==============================] - 100s 8s/step - loss: 1.1108 -
accuracy: 0.5625 - val_loss: 1.2997 - val_accuracy: 0.4706
Epoch 20/20
12/12 [==============================] - 99s 8s/step - loss: 1.0935 -
accuracy: 0.5897 - val_loss: 0.9670 - val_accuracy: 0.6324

/usr/local/lib/python3.10/dist-packages/keras/src/engine/
training.py:3103: UserWarning: You are saving your model as an HDF5
file via `model.save()`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(

import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout,
GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import img_to_array,
load_img
```

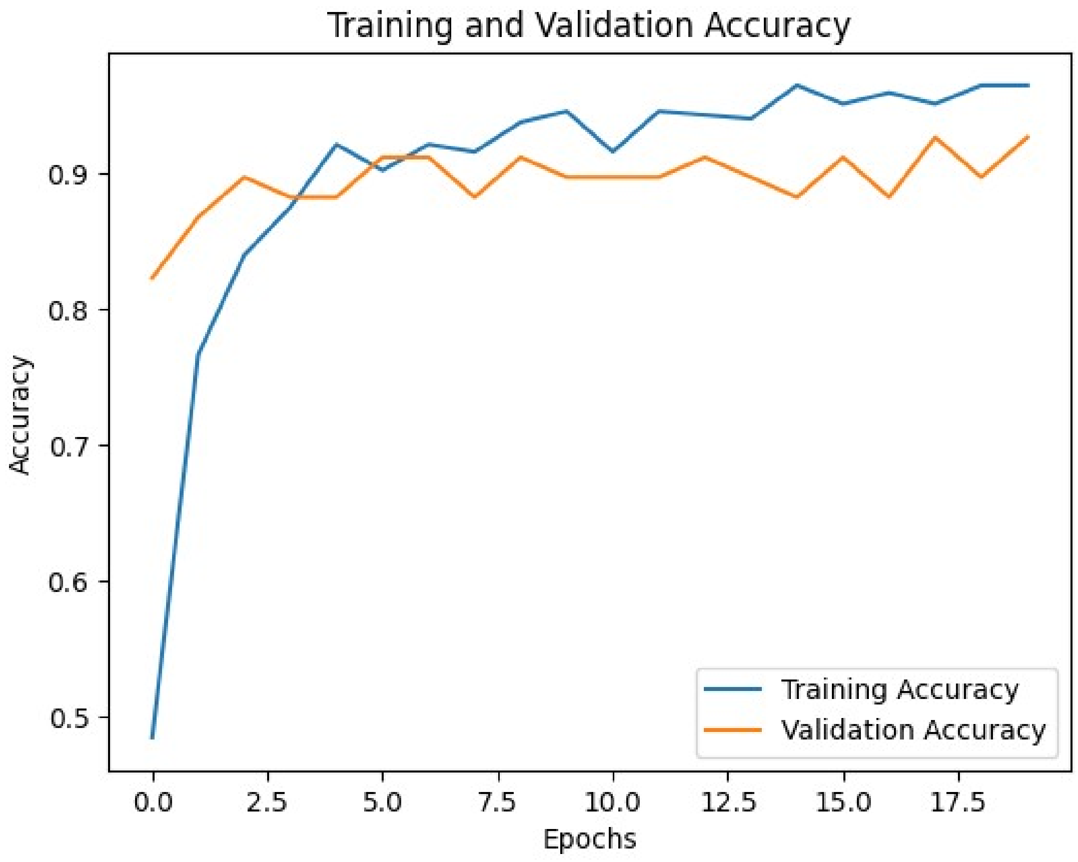
```python
# Define paths
data_dir = '/content/war_events'  # Ensure your images are in
subdirectories for each class

# Prepare dataset lists
image_paths = []
labels = []

# data_dir has subdirectories for each class
for class_name in os.listdir(data_dir):
    class_dir = os.path.join(data_dir, class_name)
    if os.path.isdir(class_dir):
        for img_name in os.listdir(class_dir):
            img_path = os.path.join(class_dir, img_name)
            image_paths.append(img_path)
            labels.append(class_name)

# Convert labels to numerical values
label_to_index = {label: idx for idx, label in enumerate(set(labels))}
labels = [label_to_index[label] for label in labels]

# Split the dataset
train_paths, val_paths, train_labels, val_labels =
train_test_split(image_paths, labels, test_size=0.2, stratify=labels)

# Data augmentation and preprocessing
def preprocess_image(img_path):
    img = load_img(img_path, target_size=(224, 224))
    img_array = img_to_array(img) / 255.0
    return img_array

def data_generator(paths, labels, batch_size, is_train):
    data_gen = ImageDataGenerator(
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest'
    ) if is_train else ImageDataGenerator(rescale=1./255)

    while True:
        for start in range(0, len(paths), batch_size):
            end = min(start + batch_size, len(paths))
            batch_paths = paths[start:end]
            batch_labels = labels[start:end]

            batch_images = np.array([preprocess_image(img_path) for
img_path in batch_paths])
```

```
            batch_labels = tf.keras.utils.to_categorical(batch_labels,
num_classes=len(label_to_index))

            if is_train:
                yield next(data_gen.flow(batch_images, batch_labels,
batch_size=batch_size))
            else:
                yield batch_images, batch_labels

batch_size = 32

train_gen = data_generator(train_paths, train_labels, batch_size,
is_train=True)
val_gen = data_generator(val_paths, val_labels, batch_size,
is_train=False)

# Load pre-trained MobileNetV2 model + higher level layers
mobilenet_model = MobileNetV2(input_shape=(224, 224, 3),
include_top=False, weights='imagenet')

# Freeze the pretrained layers
mobilenet_model.trainable = False

# Create a new model on top
model_new1 = Sequential([
    mobilenet_model,
    GlobalAveragePooling2D(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(label_to_index), activation='softmax')  # Adjust the
number of classes dynamically
])

# Compile the model
model_new1.compile(optimizer=Adam(), loss='categorical_crossentropy',
metrics=['accuracy'])

# Train the model
history_new1 = model_new1.fit(
    train_gen,
    steps_per_epoch=len(train_paths) // batch_size,
    validation_data=val_gen,
    validation_steps=len(val_paths) // batch_size,
    epochs=20
)

# Save the model
model_new1.save('war_lens_model_mobilenetv2.h5')

# Evaluate the model
```

```
val_loss, val_accuracy = model_new1.evaluate(val_gen,
steps=len(val_paths) // batch_size)
print(f'Validation accuracy: {val_accuracy}')

Epoch 1/20
12/12 [==============================] - 40s 3s/step - loss: 1.4525 -
accuracy: 0.4844 - val_loss: 0.5592 - val_accuracy: 0.8229
Epoch 2/20
12/12 [==============================] - 30s 3s/step - loss: 0.7163 -
accuracy: 0.7663 - val_loss: 0.4132 - val_accuracy: 0.8676
Epoch 3/20
12/12 [==============================] - 28s 2s/step - loss: 0.4721 -
accuracy: 0.8397 - val_loss: 0.2712 - val_accuracy: 0.8971
Epoch 4/20
12/12 [==============================] - 28s 2s/step - loss: 0.3550 -
accuracy: 0.8750 - val_loss: 0.2519 - val_accuracy: 0.8824
Epoch 5/20
12/12 [==============================] - 29s 2s/step - loss: 0.2780 -
accuracy: 0.9212 - val_loss: 0.2339 - val_accuracy: 0.8824
Epoch 6/20
12/12 [==============================] - 29s 2s/step - loss: 0.3039 -
accuracy: 0.9022 - val_loss: 0.2461 - val_accuracy: 0.9118
Epoch 7/20
12/12 [==============================] - 29s 3s/step - loss: 0.2404 -
accuracy: 0.9212 - val_loss: 0.2074 - val_accuracy: 0.9118
Epoch 8/20
12/12 [==============================] - 27s 2s/step - loss: 0.2364 -
accuracy: 0.9158 - val_loss: 0.2653 - val_accuracy: 0.8824
Epoch 9/20
12/12 [==============================] - 26s 2s/step - loss: 0.2033 -
accuracy: 0.9375 - val_loss: 0.2496 - val_accuracy: 0.9118
Epoch 10/20
12/12 [==============================] - 27s 2s/step - loss: 0.1587 -
accuracy: 0.9457 - val_loss: 0.2386 - val_accuracy: 0.8971
Epoch 11/20
12/12 [==============================] - 31s 3s/step - loss: 0.2181 -
accuracy: 0.9158 - val_loss: 0.1994 - val_accuracy: 0.8971
Epoch 12/20
12/12 [==============================] - 30s 2s/step - loss: 0.1696 -
accuracy: 0.9457 - val_loss: 0.2491 - val_accuracy: 0.8971
Epoch 13/20
12/12 [==============================] - 28s 2s/step - loss: 0.1672 -
accuracy: 0.9429 - val_loss: 0.2166 - val_accuracy: 0.9118
Epoch 14/20
12/12 [==============================] - 27s 2s/step - loss: 0.1654 -
accuracy: 0.9401 - val_loss: 0.2081 - val_accuracy: 0.8971
Epoch 15/20
12/12 [==============================] - 29s 2s/step - loss: 0.1147 -
accuracy: 0.9647 - val_loss: 0.2717 - val_accuracy: 0.8824
Epoch 16/20
```

```
12/12 [==============================] - 29s 2s/step - loss: 0.1250 -
accuracy: 0.9511 - val_loss: 0.2085 - val_accuracy: 0.9118
Epoch 17/20
12/12 [==============================] - 38s 3s/step - loss: 0.1561 -
accuracy: 0.9592 - val_loss: 0.2044 - val_accuracy: 0.8824
Epoch 18/20
12/12 [==============================] - 33s 3s/step - loss: 0.1446 -
accuracy: 0.9511 - val_loss: 0.2180 - val_accuracy: 0.9265
Epoch 19/20
12/12 [==============================] - 29s 3s/step - loss: 0.1043 -
accuracy: 0.9647 - val_loss: 0.2229 - val_accuracy: 0.8971
Epoch 20/20
12/12 [==============================] - 25s 2s/step - loss: 0.1076 -
accuracy: 0.9647 - val_loss: 0.2015 - val_accuracy: 0.9265

/usr/local/lib/python3.10/dist-packages/keras/src/engine/
training.py:3103: UserWarning: You are saving your model as an HDF5
file via `model.save()`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(

3/3 [==============================] - 3s 1s/step - loss: 0.2197 -
accuracy: 0.9265
Validation accuracy: 0.9264705777168274

import matplotlib.pyplot as plt

# Plot the training and validation loss
plt.plot(history_new1.history['loss'], label='Training Loss')
plt.plot(history_new1.history['val_loss'], label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Plot the training and validation accuracy
plt.plot(history_new1.history['accuracy'], label='Training Accuracy')
plt.plot(history_new1.history['val_accuracy'], label='Validation
Accuracy')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```


Training and Validation Loss
Training Loss
Validation Loss
Loss
Epochs
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


Training and Validation Accuracy
Accuracy
0.5
0.6
0.7
0.8
0.9
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epochs
Training Accuracy
Validation Accuracy