hw1

December 26, 2023

1 HOMEWORK 2 batch version

[ ]:
```
# %history -f output.txt
```

2 Libraries importation

[ ]:
```
import os
import cv2
import scipy
import numpy as np
import tensorflow as tf
from PIL import Image
from collections import defaultdict
from matplotlib import pyplot as plt
from sklearn.utils.class_weight import compute_class_weight
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten,␣
↪Dropout
from tensorflow.keras.metrics import Precision, Recall, BinaryAccuracy
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator

print('libraries imported')
```

```
2023-12-24 16:47:45.162513: I tensorflow/core/util/port.cc:113] oneDNN custom
operations are on. You may see slightly different numerical results due to
floating-point round-off errors from different computation orders. To turn them
off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2023-12-24 16:47:45.164394: I external/local_tsl/tsl/cuda/cudart_stub.cc:31]
Could not find cuda drivers on your machine, GPU will not be used.
2023-12-24 16:47:45.192789: E
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register
cuDNN factory: Attempting to register factory for plugin cuDNN when one has
already been registered
2023-12-24 16:47:45.192818: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
```

```
already been registered
2023-12-24 16:47:45.193617: E
external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to
register cuBLAS factory: Attempting to register factory for plugin cuBLAS when
one has already been registered
2023-12-24 16:47:45.198238: I external/local_tsl/tsl/cuda/cudart_stub.cc:31]
Could not find cuda drivers on your machine, GPU will not be used.
2023-12-24 16:47:45.198623: I tensorflow/core/platform/cpu_feature_guard.cc:182]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other
operations, rebuild TensorFlow with the appropriate compiler flags.
2023-12-24 16:47:45.820584: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not
find TensorRT
```

```
libraries imported
```

3 settings

[ ]:

```python
# Avoid OOM errors by setting GPU memory consumption growth
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
```

4 Data Collection

4.1 Data Augmentation

[ ]:

```python
datagen = ImageDataGenerator(
    rotation_range=0,
    width_shift_range=0,
    height_shift_range=0,
    shear_range=0,
    zoom_range=0,
    horizontal_flip=True,
    rescale=1./255
)
```

4.2 Data Load

[ ]:

```python
Train = datagen.flow_from_directory(
    'train',
    target_size=(256,256),
    batch_size=32,
    class_mode='categorical'
```

```
)

train_labels = []
train_labels = Train.classes
num_classes = Train.num_classes
train_labels_one_hot = tf.keras.utils.to_categorical(train_labels, num_classes=num_classes)
```

```
Found 6369 images belonging to 5 classes.
```

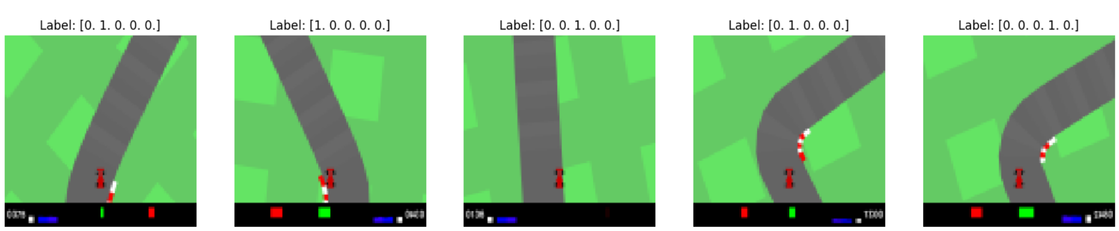
4.3 Visualization of the dataset

[ ]:

```
def visualize_dataset(dataset, num_samples=5):
    for images, labels in dataset:
        num_samples_batch = min(num_samples, len(images))
        fig, ax = plt.subplots(1, num_samples_batch, figsize=(20, 20))

        for i in range(num_samples_batch):
            ax[i].imshow((images[i] * 255).astype("uint8"))  # Remove the rescaling here
            ax[i].set_title(f"Label: {labels[i]}")
            ax[i].axis("off")

        plt.show()
        break


# num_samples = 5
# for images, labels in Train.take(1):
#     num_samples_batch = min(num_samples, len(images))
#     fig, ax = plt.subplots(1, num_samples_batch, figsize=(20, 20))
#     for i in range(num_samples):
#         ax[i].imshow(images[i].numpy().astype("uint8"))
#         ax[i].set_title(f"Label: {labels[i]}")
#         ax[i].axis("off")
#     plt.show()
```

[ ]:

```
visualize_dataset(Train)
```



5 Data Preprocessing

- Resize images to a common size (e.g., 96x96, as mentioned in the description).
- Normalize pixel values to a range between 0 and 1.
- Consider data augmentation techniques (e.g., rotation, flipping) to increase the diversity of your training set.

6 Model Selection:

6.1 Model Design

- Define your own CNN architecture. Start with a simple architecture and gradually increase complexity if needed.
- Experiment with different layer configurations, activation functions, and filter sizes.
- Consider incorporating dropout for regularization.

6.2 Approach 1

- Define the first approach with a specific architecture, optimizer, and regularization techniques.
- Choose appropriate hyperparameters (learning rate, batch size, etc.).
- Train the model on the training set and evaluate on the test set.
- Collect and analyze metrics such as accuracy, precision, recall, and F1 score.

[ ]:
```
model = Sequential()
```

[ ]:
```
optimizer = Adam(learning_rate=0.001)
```

[ ]:
```
# Layers
# (3,3) is the pixel selection, 1 is the translation of pixels
model.add(Conv2D(32, (3,3), activation='relu', input_shape=(256,256,3)))
model.add(MaxPooling2D())
model.add(Dropout(0.25))

model.add(Conv2D(64, (3,3), activation='relu'))
model.add(MaxPooling2D())
model.add(Dropout(0.25))

model.add(Conv2D(128, (3,3), activation='relu'))
model.add(MaxPooling2D())
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5)) # Dropout layer to reduce overfitting

num_classes = 5
model.add(Dense(num_classes, activation='softmax'))
```

[ ]:
```
model.compile(optimizer, loss=tf._losses.CategoricalCrossentropy(),
    metrics=['accuracy'])
```

[ ]:
```
model.summary()
```

```
Model: "sequential"

_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2  (None, 127, 127, 32)      0
 D)

 dropout (Dropout)           (None, 127, 127, 32)      0

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 max_pooling2d_1 (MaxPoolin  (None, 62, 62, 64)        0
 g2D)

 dropout_1 (Dropout)         (None, 62, 62, 64)        0

 conv2d_2 (Conv2D)           (None, 60, 60, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 30, 30, 128)       0
 g2D)

 dropout_2 (Dropout)         (None, 30, 30, 128)       0

 flatten (Flatten)           (None, 115200)            0

 dense (Dense)               (None, 512)               58982912

 dropout_3 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 5)                 2565

=================================================================
Total params: 59078725 (225.37 MB)
Trainable params: 59078725 (225.37 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

6.3 Approach 2

- Define the second approach with a different architecture, optimizer, or regularization techniques.

- Adjust hyperparameters independently of the first approach.
- Train the model on the training set and evaluate on the test set.
- Collect and analyze metrics as done for the first approach.

6.4 Hyperparameter Analysis

- Choose at least one hyperparameter (e.g., learning rate) and perform a systematic analysis.
- Train models with different values of the chosen hyperparameter.
- Compare and visualize the impact on metrics.
- Consider to apply an early stopping of the training in order to avoid overfitting (see slide 11 pag 55)
- Consider if to apply Dropout or parameter sharing

7 Model Training

[ ]:
```
logdir = 'logs'
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=logdir)
```

without validation data:

[ ]:
```
class_labels = np.unique(train_labels)
class_weights = compute_class_weight(class_weight='balanced', classes=np.
 ↪unique(train_labels), y=train_labels)
class_weights_dict = dict(zip(class_labels, class_weights))
```

[ ]:
```
hist = model.fit(Train, epochs=20, callbacks=[tensorboard_callback],␣
 ↪class_weight=class_weights_dict)
```

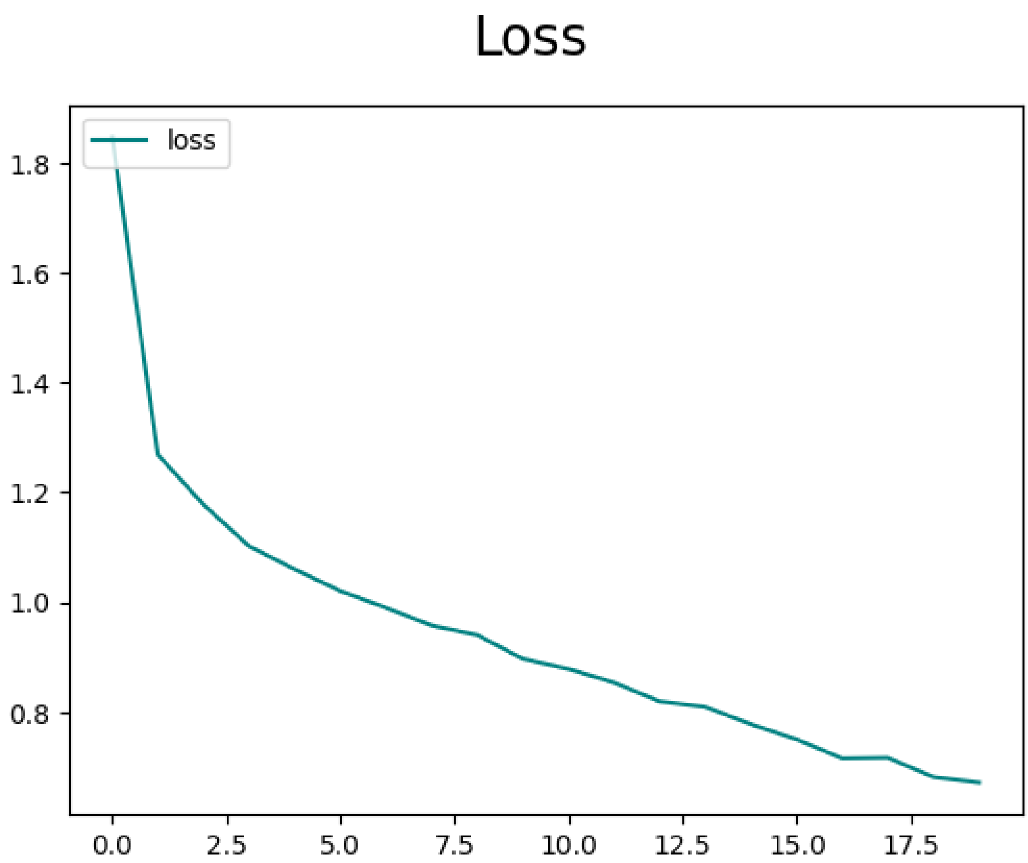
```
Epoch 1/20
200/200 [==============================] - 183s 912ms/step - loss: 1.8473 -
accuracy: 0.4318
Epoch 2/20
200/200 [==============================] - 184s 919ms/step - loss: 1.2687 -
accuracy: 0.5199
Epoch 3/20
200/200 [==============================] - 200s 998ms/step - loss: 1.1776 -
accuracy: 0.5635
Epoch 4/20
200/200 [==============================] - 181s 903ms/step - loss: 1.1017 -
accuracy: 0.5907
Epoch 5/20
200/200 [==============================] - 185s 925ms/step - loss: 1.0596 -
accuracy: 0.6119
Epoch 6/20
200/200 [==============================] - 178s 889ms/step - loss: 1.0201 -
accuracy: 0.6235
Epoch 7/20
200/200 [==============================] - 178s 891ms/step - loss: 0.9896 -
```

```
accuracy: 0.6360
Epoch 8/20
200/200 [==============================] - 178s 891ms/step - loss: 0.9574 -
accuracy: 0.6530
Epoch 9/20
200/200 [==============================] - 178s 888ms/step - loss: 0.9399 -
accuracy: 0.6568
Epoch 10/20
200/200 [==============================] - 193s 964ms/step - loss: 0.8966 -
accuracy: 0.6670
Epoch 11/20
200/200 [==============================] - 187s 933ms/step - loss: 0.8781 -
accuracy: 0.6791
Epoch 12/20
200/200 [==============================] - 187s 933ms/step - loss: 0.8533 -
accuracy: 0.6849
Epoch 13/20
200/200 [==============================] - 196s 982ms/step - loss: 0.8190 -
accuracy: 0.6981
Epoch 14/20
200/200 [==============================] - 186s 930ms/step - loss: 0.8089 -
accuracy: 0.6990
Epoch 15/20
200/200 [==============================] - 186s 929ms/step - loss: 0.7775 -
accuracy: 0.7185
Epoch 16/20
200/200 [==============================] - 185s 925ms/step - loss: 0.7498 -
accuracy: 0.7279
Epoch 17/20
200/200 [==============================] - 186s 927ms/step - loss: 0.7153 -
accuracy: 0.7408
Epoch 18/20
200/200 [==============================] - 185s 925ms/step - loss: 0.7164 -
accuracy: 0.7343
Epoch 19/20
200/200 [==============================] - 190s 949ms/step - loss: 0.6813 -
accuracy: 0.7494
Epoch 20/20
200/200 [==============================] - 197s 985ms/step - loss: 0.6713 -
accuracy: 0.7569
```

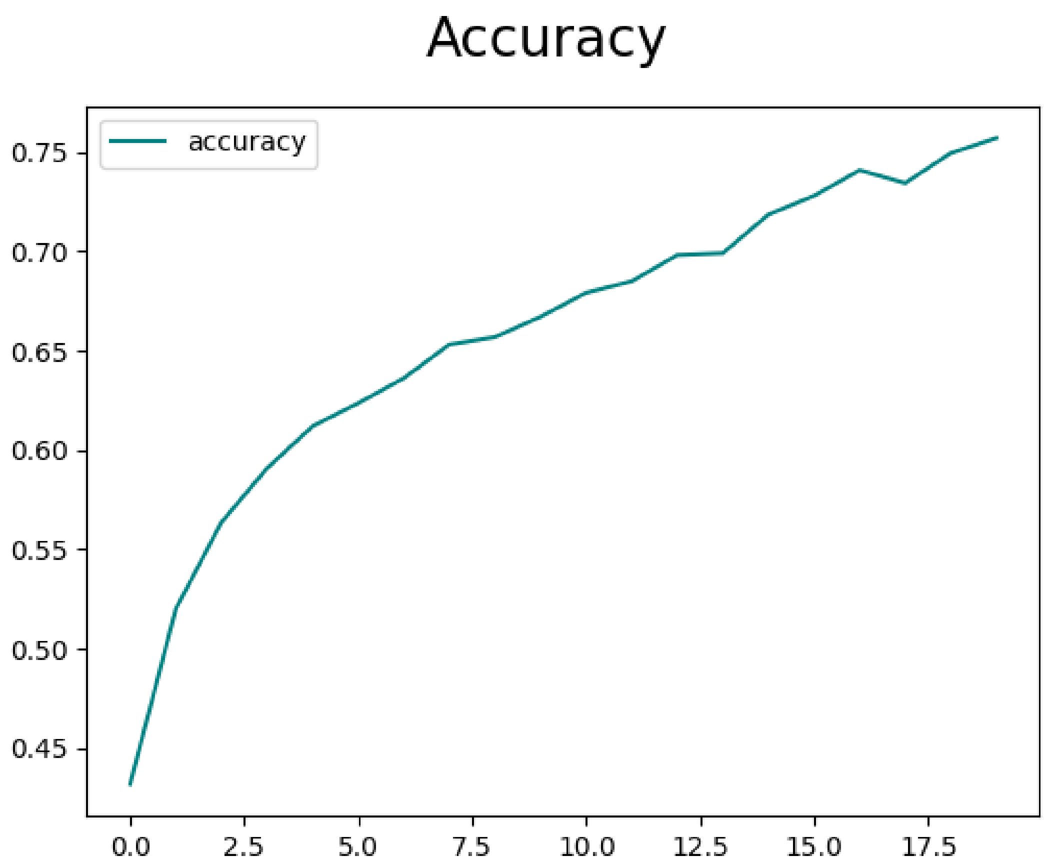
7.1 Plotting Model Performance

[ ]:
```
fig = plt.figure()
plt.plot(hist.history['loss'], color='teal', label='loss')
fig.suptitle('Loss', fontsize=20)
plt.legend(loc="upper left")
```

```
plt.show()
```




```
[ ]: fig = plt.figure()
     plt.plot(hist.history['accuracy'], color='teal', label='accuracy')
     fig.suptitle('Accuracy', fontsize=20)
     plt.legend(loc="upper left")
     plt.show()
```



8 Evaluate Performance

[ ]:
```
pre = Precision()
re = Recall()
acc = BinaryAccuracy()
```

9 Test

[ ]:
```
Test = tf.keras.utils.image_dataset_from_directory('test')
test_iterator = Test.as_numpy_iterator()

all_X_test = []
all_y_test = []

for test_batch in test_iterator:
    X_test_batch, y_test_batch = test_batch
    X_test_normalized_batch = X_test_batch/255.0
    yhat_batch = model.predict(X_test_normalized_batch)
```

```
    all_X_test.append(X_test_normalized_batch)
    all_y_test.append(y_test_batch)

X_test_normalized = np.concatenate(all_X_test)
y_test = np.concatenate(all_y_test)

test_iterator = Test.as_numpy_iterator()
Test_batch = next(test_iterator)
X_test, y_test = Test_batch
X_test_normalized = X_test/255.0
```

```
Found 2749 files belonging to 5 classes.
1/1 [==============================] - 0s 179ms/step
1/1 [==============================] - 0s 204ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 192ms/step
1/1 [==============================] - 0s 185ms/step
1/1 [==============================] - 0s 175ms/step
1/1 [==============================] - 0s 197ms/step
1/1 [==============================] - 0s 177ms/step
1/1 [==============================] - 0s 199ms/step
1/1 [==============================] - 0s 182ms/step
1/1 [==============================] - 0s 187ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 178ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 182ms/step
1/1 [==============================] - 0s 179ms/step
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 198ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 201ms/step
1/1 [==============================] - 0s 200ms/step
1/1 [==============================] - 0s 195ms/step
1/1 [==============================] - 0s 209ms/step
1/1 [==============================] - 0s 215ms/step
1/1 [==============================] - 0s 208ms/step
1/1 [==============================] - 0s 214ms/step
1/1 [==============================] - 0s 215ms/step
1/1 [==============================] - 0s 228ms/step
1/1 [==============================] - 0s 199ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 193ms/step
1/1 [==============================] - 0s 199ms/step
1/1 [==============================] - 0s 183ms/step
1/1 [==============================] - 0s 185ms/step
1/1 [==============================] - 0s 182ms/step
```

```
1/1 [==============================] - 0s 185ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 193ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 198ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 181ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 193ms/step
1/1 [==============================] - 0s 200ms/step
1/1 [==============================] - 0s 204ms/step
1/1 [==============================] - 0s 198ms/step
1/1 [==============================] - 0s 198ms/step
1/1 [==============================] - 0s 207ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 185ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 187ms/step
1/1 [==============================] - 0s 202ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 185ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 201ms/step
1/1 [==============================] - 0s 194ms/step
1/1 [==============================] - 0s 229ms/step
1/1 [==============================] - 0s 217ms/step
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 250ms/step
1/1 [==============================] - 0s 223ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 199ms/step
1/1 [==============================] - 0s 198ms/step
1/1 [==============================] - 0s 206ms/step
1/1 [==============================] - 0s 227ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 195ms/step
1/1 [==============================] - 0s 198ms/step
1/1 [==============================] - 0s 193ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 195ms/step
1/1 [==============================] - 0s 202ms/step
1/1 [==============================] - 0s 193ms/step
1/1 [==============================] - 0s 198ms/step
```

```
1/1 [==============================] - 0s 197ms/step
1/1 [==============================] - 0s 213ms/step
1/1 [==============================] - 0s 188ms/step
```

[ ]:
```
yhat = model.predict(X_test_normalized)
```

```
1/1 [==============================] - 0s 173ms/step
```

[ ]:
```
pre = Precision()
re = Recall()
acc = BinaryAccuracy()
```

[ ]:
```
for batch in Test.as_numpy_iterator():
    X, y = batch
    yhat = model.predict(X)
    y_one_hot = tf.keras.utils.to_categorical(y, num_classes=num_classes)
    pre.update_state(y_one_hot, yhat)
    re.update_state(y_one_hot, yhat)
    acc.update_state(y_one_hot, yhat)
```

```
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 180ms/step
1/1 [==============================] - 0s 184ms/step
1/1 [==============================] - 0s 182ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 180ms/step
1/1 [==============================] - 0s 178ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 177ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 185ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 194ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 184ms/step
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 187ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 192ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 182ms/step
1/1 [==============================] - 0s 185ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 185ms/step
```

```
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 199ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 199ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 182ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 184ms/step
1/1 [==============================] - 0s 179ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 198ms/step
1/1 [==============================] - 0s 192ms/step
1/1 [==============================] - 0s 198ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 187ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 185ms/step
1/1 [==============================] - 0s 197ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 194ms/step
1/1 [==============================] - 0s 186ms/step
1/1 [==============================] - 0s 201ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 193ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 187ms/step
1/1 [==============================] - 0s 199ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 189ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 190ms/step
1/1 [==============================] - 0s 202ms/step
1/1 [==============================] - 0s 187ms/step
1/1 [==============================] - 0s 195ms/step
```

```
1/1 [==============================] - 0s 191ms/step
1/1 [==============================] - 0s 187ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 188ms/step
1/1 [==============================] - 0s 192ms/step
1/1 [==============================] - 0s 193ms/step
1/1 [==============================] - 0s 229ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 196ms/step
1/1 [==============================] - 0s 179ms/step
```

[ ]:
```
print(f'Precision:{pre.result().numpy()}, Recall:{re.result().numpy()},␣
↪Accuracy:{acc.result().numpy()}')
```

```
Precision:0.567115306854248, Recall:0.567115306854248,
Accuracy:0.8268461227416992
```

9.1 Results Visualization

- Create visualizations (tables, charts, graphs) to present your results.
- Provide detailed commentary on each visualization, explaining trends or differences observed.

9.2 Fine-Tuning

9.3 Deployment

[ ]:
```

```

##valutazioni da fare poi: - regularization per ridurre l'overfitting? - il numero di images cambia da classe a classe (train) vedere se serve prenderne un numero uguale per ciascuna classe -