Data Analysis on the Relationship of Autism with Facial Expressions in Children

Introduction

In this notebook, we will explore the relationship between autism and facial expressions in children. We will analyze a dataset containing facial images of autistic and non-autistic children.

Objectives

- Investigate how facial expressions relate to autism in children.
- Perform visual and statistical analysis to better understand the data.
- Explore the possibility of using facial recognition for early autism detection.

Dataset

The dataset used contains facial images of children labeled as autistic and non-autistic. It was compiled for the purpose of researching the relationship between facial expressions and autism.

Notebook Structure

1. **Data loading and exploration**
   - Import libraries
   - Loading data.
   - Visualizing some samples.
2. **Exploratory data analysis**
   - Descriptive statistics.
3. **Modeling and autism detection**
   - Image preprocessing.
   - Training facial recognition models.
   - Evaluating autism detection capability.

Data Loading and Exploration

Import Libraries

First we are going to import the different libraries that will conform our project and will help us at the time of the creation of our neural network.

```
import numpy as np
import pandas as pd
import os
import cv2
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers, models
from keras.datasets import fashion_mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import SGD
```

```
from keras.utils import to_categorical
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import numpy as np
import numpy as np
from keras.layers import Input
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
import seaborn as sns
from sklearn.metrics import classification_report
import sys
from keras import optimizers, metrics, callbacks
```

```
2024-05-12 20:57:32.919105: 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
2024-05-12 20:57:32.919206: 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
2024-05-12 20:57:33.056500: 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
```

Loading Data

In this notebook, our objective is to load the dataset containing images of both autistic and non-autistic children. This dataset is organized into three distinct folders: `train`, `valid`, and `test`, each containing a portion of the dataset.

Rather than keeping the data separated in these folders, we will consolidate them into a single dataset. However, before combining the data, we will perform a shuffle of the images within each folder individually. This initial step ensures randomness in our dataset, which is crucial for effective training and evaluation of machine learning models.

```
train = []
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        full_path = os.path.join(dirname, filename)
        archive=os.path.join(dirname, filename)
        img=cv2.imread(archive)
        if img is not None:
            img = cv2.resize(img, (200, 200))
            relative_path = os.path.relpath(full_path,
        '/kaggle/input/autism-facial-image-dataset/Facial images
        dataset for Autism Detection/')
            train.append([img,relative_path,archive])
        else:
            print(f'Failed Load Image:{archive}')

train_data = []
valid_data = []
test_data = []
for item in train:
    full_path = item[0]
    label = item[1].split('/')[1]
    path=item[2]
    is_auth=0 if label == 'autistic' else 1
    train_data.append([full_path, label,path,is_auth])
np.random.shuffle(train_data)
print(f'There are {len(train_data)} images')
```

```
There are 2926 images
```

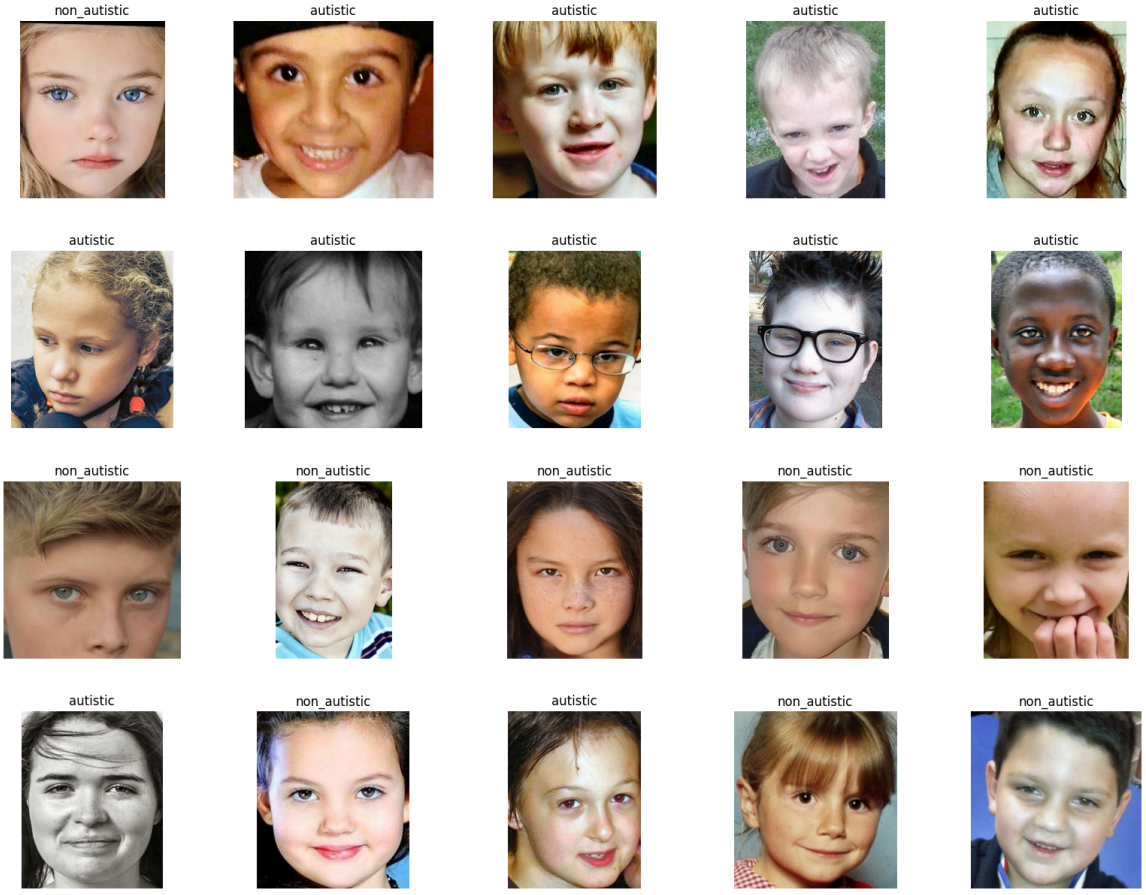
Data Visualization

Let's create a sample visualization using randomly generated data obtained earlier. We'll utilize Matplotlib to plot this data, showcasing its versatility and ease of use. Through this visualization, we can explore patterns, trends, or distributions within the list, demonstrating the power of data visualization in understanding and communicating insights.

```
def show_images(data):
    num_images = min(len(data), 20)
    fig, axes = plt.subplots(4, 5, figsize=(20, 15))
    for i in range(num_images):
        img_path = data[i][2]
```

```
            label = data[i][1]
            img = mpimg.imread(img_path)
            row = i // 5
            col = i % 5
            axes[row, col].imshow(img)
            axes[row, col].set_title(label)
            axes[row, col].axis('off')
        plt.subplots_adjust(wspace=0.2, hspace=0.3)
        plt.show()

show_images(train_data)
```



Exploratory data analysis

```
<h4 style="color:#FFFFFF; font-family: 'Times New Roman', Times,
serif;">Descriptive Stadistics</h4><div>
```

Upon examining the obtained list,we'll conduct an assessment to determine the count of children with and without autism. This analysis will provide insights into the distribution of autism diagnosis within the list.

Description of the partitions:

- This provides a summary of the 'Autism' column, including count, unique values, the most frequent value, and frequency of the most frequent value.

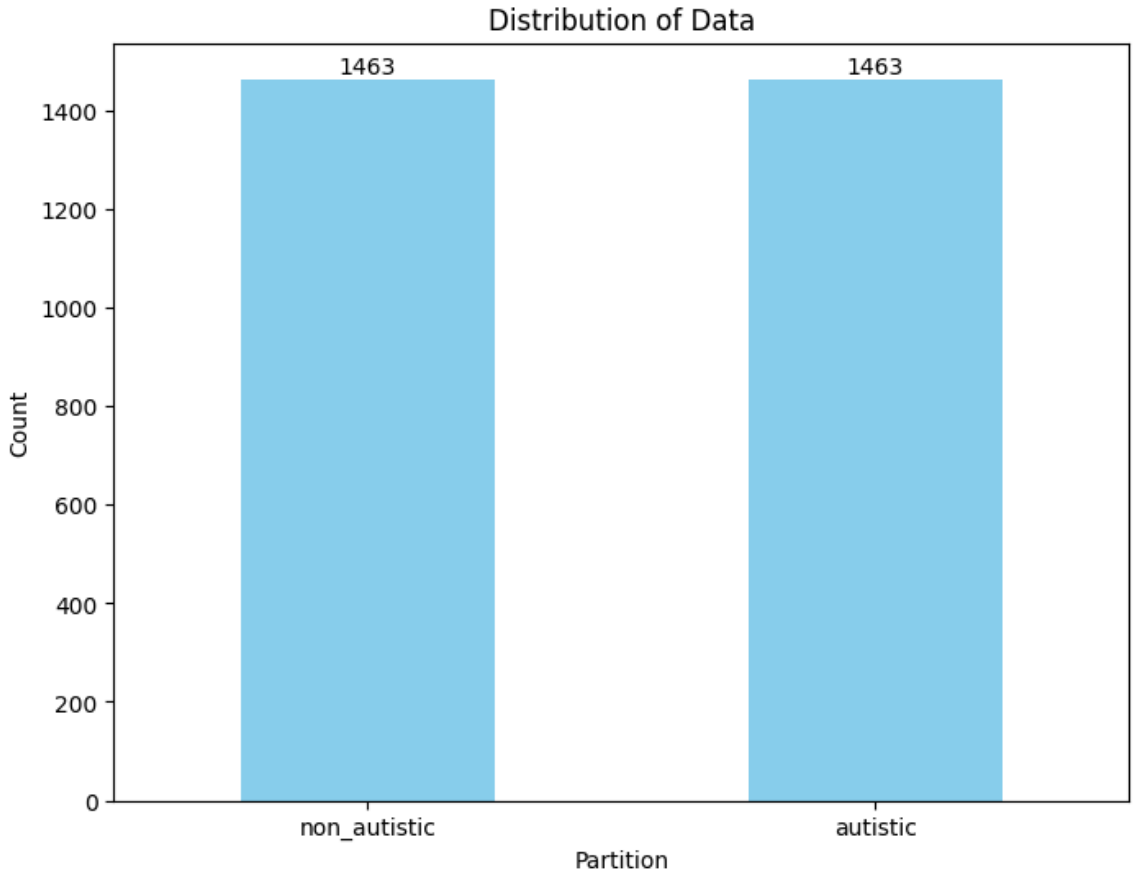
Value counts for each partition:

- This shows the count of each unique value in the 'Autism' column, providing insight into the distribution of data among different partitions.

```
df = pd.DataFrame(train_data, columns=['Location',
        'Autism','TotalLocation','Autism_Count'])
print("\nDescription of Partitions:")
print(df['Autism'].describe())
print("\nValue Counts in Each Partition:")
print(df['Autism'].value_counts())
plt.figure(figsize=(8, 6))
ax = df['Autism'].value_counts().plot(kind='bar', color='skyblue')
plt.title('Distribution of Data')
plt.xlabel('Partition')
plt.ylabel('Count')
plt.xticks(rotation=0)
for i, v in enumerate(df['Autism'].value_counts()):
    ax.text(i, v + 0.1, str(v), ha='center', va='bottom')
plt.show()
```

```
Description of Partitions:
count             2926
unique               2
top       non_autistic
freq              1463
Name: Autism, dtype: object

Value Counts in Each Partition:
Autism
non_autistic    1463
autistic        1463
Name: count, dtype: int64
```



We have been able to verify in the previous deductions that there are an equal number of cases for autism and non-autism. It was also determined that there are only these 2 data points for autism and non-autism.

Modeling and autism detection

Image Preproccesing

We divide the values generated previously in valid test and train randomly, for this we must select a sample proportional to the amount of data we have.

```
y=[]
x=[]
np.random.shuffle(train_data)
for i in range(len(train_data)):
    x.append(train_data[i][0])
    y.append(train_data[i][3])
```

```
y=np.array(y)
x=np.array(x)
x_train_full,x_test, y_train_full, y_test=
        train_test_split(x,y,test_size=0.2)
print(f'Y has {y_train_full.shape[0]} values and X has
        {x_train_full.shape[0]} values')
X_valid, X_train = x_train_full[:250] / 255., x_train_full[250:] /
        255.
y_valid, y_train = y_train_full[:250], y_train_full[250:]
X_test = x_test / 255.

Y has 2340 values and X has 2340 values

x_train_full.shape, y_test.shape

((2340, 200, 200, 3), (586,))
```

Training facial recognition models

Now we are going to make the declaration of the model that we are going to use. It will consist of:

Definition of input and output dimensions:

- `input_dimension`: Defines the input dimensions of the model, excluding the channel dimension.
- `dimension_output`: Defines the output dimension of the model, in this case, 1 for binary classification.

Sequential Model Definition:

- Sequential Keras model, a linear stack of layers.

Adding Layers to the Model:

- Convolutional and MaxPooling layers for feature extraction from images.
- Densely connected layers for final classification.
- Output layer with sigmoid activation for binary classification probability.

Model Summary:

- Prints the architecture of the model and the number of trainable parameters.

Model Compilation:

- Uses the Adam optimizer with a specified learning rate.

- Uses binary crossentropy loss function for binary classification problems.
- Defines metrics such as accuracy, precision, and recall to evaluate model performance during training.

```
dimension_entrada = x_train_full.shape[1:4]
dimension_salida = 1
model = models.Sequential()
model.add(layers.Conv2D(32, (3,3), activation="relu",
        input_shape=dimension_entrada))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3), activation="relu"))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(128, (3,3), activation="relu"))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(256, (3,3), activation="relu"))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(512, (3,3), activation="relu"))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Flatten())
# model.add(layers.Dense(512, activation="relu"))
# model.add(layers.Dropout(0.5))
model.add(layers.Dense(256, activation="relu"))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(128, activation="relu"))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(64, activation="relu"))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation="sigmoid"))
model.summary()
opt = optimizers.Adam(learning_rate=0.00001)
model.compile(loss="binary_crossentropy", optimizer=opt, metrics=
        ["accuracy"])
```

```
/opt/conda/lib/python3.10/site-
packages/keras/src/layers/convolutional/base_conv.py:99: 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__(
```

Model: "sequential"

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 198, 198, 32)
batch_normalization (BatchNormalization)	(None, 198, 198, 32)
max_pooling2d (MaxPooling2D)	(None, 99, 99, 32)
conv2d_1 (Conv2D)	(None, 97, 97, 64)
batch_normalization_1 (BatchNormalization)	(None, 97, 97, 64)
max_pooling2d_1 (MaxPooling2D)	(None, 48, 48, 64)
conv2d_2 (Conv2D)	(None, 46, 46, 128)
batch_normalization_2 (BatchNormalization)	(None, 46, 46, 128)
max_pooling2d_2 (MaxPooling2D)	(None, 23, 23, 128)
conv2d_3 (Conv2D)	(None, 21, 21, 256)
batch_normalization_3 (BatchNormalization)	(None, 21, 21, 256)
max_pooling2d_3 (MaxPooling2D)	(None, 10, 10, 256)
conv2d_4 (Conv2D)	(None, 8, 8, 512)
batch_normalization_4 (BatchNormalization)	(None, 8, 8, 512)
max_pooling2d_4 (MaxPooling2D)	(None, 4, 4, 512)
flatten (Flatten)	(None, 8192)
dense (Dense)	(None, 256)
dropout (Dropout)	(None, 256)
dense_1 (Dense)	(None, 128)
dropout_1 (Dropout)	(None, 128)
dense_2 (Dense)	(None, 64)
dropout_2 (Dropout)	(None, 64)

```
| dense_3 (Dense)                  | (None, 1)                 |
```

Total params: 3,711,169 (14.16 MB)

Trainable params: 3,709,185 (14.15 MB)

Non-trainable params: 1,984 (7.75 KB)

We will now start with the previously launched model:

```
early_stopping_cb =
        keras.callbacks.EarlyStopping(patience=10,restore_best_weights=True)
history =
        model.fit(X_train,y_train,epochs=sys.maxsize,validation_data=
        (X_valid,y_valid),callbacks=[early_stopping_cb])
```

```
Epoch 1/9223372036854775807

2024-05-12 20:58:28.711232: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 65: 5.41783, expected 4.72242
2024-05-12 20:58:28.711286: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 66: 4.33458, expected 3.63917
2024-05-12 20:58:28.711296: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 141: 5.05684, expected 4.36143
2024-05-12 20:58:28.711304: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 142: 5.83136, expected 5.13595
2024-05-12 20:58:28.711312: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 143: 5.29561, expected 4.6002
2024-05-12 20:58:28.711320: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 216: 5.34702, expected 4.65161
2024-05-12 20:58:28.711328: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 264: 5.29538, expected 4.59997
2024-05-12 20:58:28.711336: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 339: 5.43918, expected 4.74376
2024-05-12 20:58:28.711343: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 340: 4.69161, expected 3.9962
2024-05-12 20:58:28.711351: E
```

```
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 341: 4.86162, expected 4.1662
2024-05-12 20:58:28.729686: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:705]
Results mismatch between different convolution algorithms. This is
likely a bug/unexpected loss of precision in cudnn.
(f32[32,32,198,198]{3,2,1,0}, u8[0]{0}) custom-call(f32[32,3,200,200]
{3,2,1,0}, f32[32,3,3,3]{3,2,1,0}, f32[32]{0}), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
backend_config=
{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,'
for eng20{k2=2,k4=1,k5=1,k6=0,k7=0} vs eng15{k5=1,k6=0,k7=1,k10=1}
2024-05-12 20:58:28.729741: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:270]
Device: Tesla P100-PCIE-16GB
2024-05-12 20:58:28.729750: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:271]
Platform: Compute Capability 6.0
2024-05-12 20:58:28.729757: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:272]
Driver: 12020 (535.129.3)
2024-05-12 20:58:28.729764: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:273]
Runtime: <undefined>
2024-05-12 20:58:28.729780: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:280] cudnn
version: 8.9.0
2024-05-12 20:58:29.341777: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 65: 5.41783, expected 4.72242
2024-05-12 20:58:29.341836: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 66: 4.33458, expected 3.63917
2024-05-12 20:58:29.341846: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 141: 5.05684, expected 4.36143
2024-05-12 20:58:29.341854: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 142: 5.83136, expected 5.13595
2024-05-12 20:58:29.341861: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 143: 5.29561, expected 4.6002
```

```
2024-05-12 20:58:29.341869: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 216: 5.34702, expected 4.65161
2024-05-12 20:58:29.341877: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 264: 5.29538, expected 4.59997
2024-05-12 20:58:29.341885: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 339: 5.43918, expected 4.74376
2024-05-12 20:58:29.341893: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 340: 4.69161, expected 3.9962
2024-05-12 20:58:29.341900: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 341: 4.86162, expected 4.1662
2024-05-12 20:58:29.359351: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:705]
Results mismatch between different convolution algorithms. This is
likely a bug/unexpected loss of precision in cudnn.
(f32[32,32,198,198]{3,2,1,0}, u8[0]{0}) custom-call(f32[32,3,200,200]
{3,2,1,0}, f32[32,3,3,3]{3,2,1,0}, f32[32]{0}), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
backend_config=
{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,'
for eng20{k2=2,k4=1,k5=1,k6=0,k7=0} vs eng15{k5=1,k6=0,k7=1,k10=1}
2024-05-12 20:58:29.359394: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:270]
Device: Tesla P100-PCIE-16GB
2024-05-12 20:58:29.359403: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:271]
Platform: Compute Capability 6.0
2024-05-12 20:58:29.359410: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:272]
Driver: 12020 (535.129.3)
2024-05-12 20:58:29.359416: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:273]
Runtime: <undefined>
2024-05-12 20:58:29.359438: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:280] cudnn
version: 8.9.0
```

```
 5/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.5469 - loss:
1.5669

WARNING: All log messages before absl::InitializeLog() is called are
written to STDERR
I0000 00:00:1715547517.678552      71 device_compiler.h:186] Compiled
cluster using XLA!  This line is logged at most once for the lifetime
of the process.

65/66 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step - accuracy: 0.5111 - loss:
1.5027

2024-05-12 20:58:42.275288: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39270: 4.54897, expected 3.8847
2024-05-12 20:58:42.275350: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39271: 5.56504, expected 4.90076
2024-05-12 20:58:42.275360: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39346: 5.13362, expected 4.46934
2024-05-12 20:58:42.275368: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39347: 5.39993, expected 4.73565
2024-05-12 20:58:42.275376: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39468: 4.65743, expected 3.99315
2024-05-12 20:58:42.275384: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39544: 4.67595, expected 4.01167
2024-05-12 20:58:42.275392: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39545: 4.23047, expected 3.56619
2024-05-12 20:58:42.275399: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39618: 5.25602, expected 4.59174
2024-05-12 20:58:42.275408: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39742: 5.30235, expected 4.63807
2024-05-12 20:58:42.275415: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39743: 4.48614, expected 3.82186
2024-05-12 20:58:42.281194: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:705]
```

```
Results mismatch between different convolution algorithms. This is
likely a bug/unexpected loss of precision in cudnn.
(f32[10,32,198,198]{3,2,1,0}, u8[0]{0}) custom-call(f32[10,3,200,200]
{3,2,1,0}, f32[32,3,3,3]{3,2,1,0}, f32[32]{0}), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
backend_config=
{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,'
for eng20{k2=2,k4=1,k5=1,k6=0,k7=0} vs eng15{k5=1,k6=0,k7=1,k10=1}
2024-05-12 20:58:42.281225: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:270]
Device: Tesla P100-PCIE-16GB
2024-05-12 20:58:42.281234: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:271]
Platform: Compute Capability 6.0
2024-05-12 20:58:42.281242: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:272]
Driver: 12020 (535.129.3)
2024-05-12 20:58:42.281250: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:273]
Runtime: <undefined>
2024-05-12 20:58:42.281266: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:280] cudnn
version: 8.9.0
2024-05-12 20:58:42.435847: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39270: 4.54897, expected 3.8847
2024-05-12 20:58:42.435897: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39271: 5.56504, expected 4.90076
2024-05-12 20:58:42.435907: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39346: 5.13362, expected 4.46934
2024-05-12 20:58:42.435915: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39347: 5.39993, expected 4.73565
2024-05-12 20:58:42.435923: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39468: 4.65743, expected 3.99315
2024-05-12 20:58:42.435931: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39544: 4.67595, expected 4.01167
2024-05-12 20:58:42.435939: E
```

```
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39545: 4.23047, expected 3.56619
2024-05-12 20:58:42.435947: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39618: 5.2562, expected 4.59174
2024-05-12 20:58:42.435954: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39742: 5.30235, expected 4.63807
2024-05-12 20:58:42.435962: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 39743: 4.48614, expected 3.82186
2024-05-12 20:58:42.441508: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:705]
Results mismatch between different convolution algorithms. This is
likely a bug/unexpected loss of precision in cudnn.
(f32[10,32,198,198]{3,2,1,0}, u8[0]{0}) custom-call(f32[10,3,200,200]
{3,2,1,0}, f32[32,3,3,3]{3,2,1,0}, f32[32]{0}), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
backend_config=
{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,'
for eng20{k2=2,k4=1,k5=1,k6=0,k7=0} vs eng15{k5=1,k6=0,k7=1,k10=1}
2024-05-12 20:58:42.441541: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:270]
Device: Tesla P100-PCIE-16GB
2024-05-12 20:58:42.441551: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:271]
Platform: Compute Capability 6.0
2024-05-12 20:58:42.441558: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:272]
Driver: 12020 (535.129.3)
2024-05-12 20:58:42.441566: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:273]
Runtime: <undefined>
2024-05-12 20:58:42.441582: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:280] cudnn
version: 8.9.0


66/66 ━━━━━━━━━━━━━━━━━━━━ 0s 169ms/step - accuracy: 0.5111 - loss:
1.5006

2024-05-12 20:58:50.279918: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
```

```
Difference at 117678: 3.90812, expected 3.37558
2024-05-12 20:58:50.279974: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 117715: 4.24471, expected 3.71217
2024-05-12 20:58:50.279984: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 117754: 4.16554, expected 3.633
2024-05-12 20:58:50.279993: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 117827: 4.20254, expected 3.67
2024-05-12 20:58:50.280001: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 117952: 3.1382, expected 2.60566
2024-05-12 20:58:50.280010: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 118150: 3.74092, expected 3.20838
2024-05-12 20:58:50.280019: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 118418: 4.2852, expected 3.75266
2024-05-12 20:58:50.280028: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 118791: 4.29326, expected 3.76072
2024-05-12 20:58:50.280036: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 118871: 4.26869, expected 3.73616
2024-05-12 20:58:50.280044: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 118941: 3.25332, expected 2.72078
2024-05-12 20:58:50.294290: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:705]
Results mismatch between different convolution algorithms. This is
likely a bug/unexpected loss of precision in cudnn.
(f32[26,32,198,198]{3,2,1,0}, u8[0]{0}) custom-call(f32[26,3,200,200]
{3,2,1,0}, f32[32,3,3,3]{3,2,1,0}, f32[32]{0}), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
backend_config=
{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,'
for eng20{k2=2,k4=1,k5=1,k6=0,k7=0} vs eng15{k5=1,k6=0,k7=1,k10=1}
2024-05-12 20:58:50.294331: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:270]
Device: Tesla P100-PCIE-16GB
2024-05-12 20:58:50.294339: E
```

```
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:271]
Platform: Compute Capability 6.0
2024-05-12 20:58:50.294346: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:272]
Driver: 12020 (535.129.3)
2024-05-12 20:58:50.294353: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:273]
Runtime: <undefined>
2024-05-12 20:58:50.294368: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:280] cudnn
version: 8.9.0
2024-05-12 20:58:50.666434: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 117678: 3.90812, expected 3.37558
2024-05-12 20:58:50.666490: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 117715: 4.24471, expected 3.71217
2024-05-12 20:58:50.666500: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 117754: 4.16554, expected 3.633
2024-05-12 20:58:50.666508: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 117827: 4.20254, expected 3.67
2024-05-12 20:58:50.666517: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 117952: 3.1382, expected 2.60566
2024-05-12 20:58:50.666525: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 118150: 3.74092, expected 3.20838
2024-05-12 20:58:50.666534: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 118418: 4.2852, expected 3.75266
2024-05-12 20:58:50.666543: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 118791: 4.29326, expected 3.76072
2024-05-12 20:58:50.666551: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 118871: 4.26869, expected 3.73616
2024-05-12 20:58:50.666559: E
external/local_xla/xla/service/gpu/buffer_comparator.cc:1137]
Difference at 118941: 3.25332, expected 2.72078
2024-05-12 20:58:50.680777: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:705]
```

```
Results mismatch between different convolution algorithms. This is
likely a bug/unexpected loss of precision in cudnn.
(f32[26,32,198,198]{3,2,1,0}, u8[0]{0}) custom-call(f32[26,3,200,200]
{3,2,1,0}, f32[32,3,3,3]{3,2,1,0}, f32[32]{0}), window={size=3x3},
dim_labels=bf01_oi01->bf01,
custom_call_target="__cudnn$convBiasActivationForward",
backend_config=
{"conv_result_scale":1,"activation_mode":"kRelu","side_input_scale":0,'
for eng20{k2=2,k4=1,k5=1,k6=0,k7=0} vs eng15{k5=1,k6=0,k7=1,k10=1}
2024-05-12 20:58:50.680809: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:270]
Device: Tesla P100-PCIE-16GB
2024-05-12 20:58:50.680817: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:271]
Platform: Compute Capability 6.0
2024-05-12 20:58:50.680825: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:272]
Driver: 12020 (535.129.3)
2024-05-12 20:58:50.680831: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:273]
Runtime: <undefined>
2024-05-12 20:58:50.680846: E
external/local_xla/xla/service/gpu/conv_algorithm_picker.cc:280] cudnn
version: 8.9.0


66/66 ━━━━━━━━━━━━━━━━━━━━ 31s 222ms/step - accuracy: 0.5110 - loss:
1.4984 - val_accuracy: 0.4920 - val_loss: 0.7064
Epoch 2/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.5459 - loss:
1.0437 - val_accuracy: 0.4920 - val_loss: 0.7725
Epoch 3/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.5715 - loss:
0.9336 - val_accuracy: 0.4920 - val_loss: 0.8777
Epoch 4/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.5984 - loss:
0.8723 - val_accuracy: 0.4920 - val_loss: 0.9035
Epoch 5/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.6106 - loss:
0.7461 - val_accuracy: 0.5000 - val_loss: 0.8355
Epoch 6/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.6096 - loss:
0.7323 - val_accuracy: 0.5360 - val_loss: 0.7367
Epoch 7/9223372036854775807
```

```
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.6408 - loss:
0.7024 - val_accuracy: 0.5760 - val_loss: 0.6610
Epoch 8/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.6431 - loss:
0.6627 - val_accuracy: 0.6280 - val_loss: 0.6053
Epoch 9/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.6443 - loss:
0.6596 - val_accuracy: 0.7000 - val_loss: 0.5697
Epoch 10/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.6546 - loss:
0.6460 - val_accuracy: 0.7120 - val_loss: 0.5612
Epoch 11/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - accuracy: 0.7032 - loss:
0.5981 - val_accuracy: 0.7200 - val_loss: 0.5477
Epoch 12/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.6753 - loss:
0.6362 - val_accuracy: 0.7400 - val_loss: 0.5387
Epoch 13/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.6733 - loss:
0.6105 - val_accuracy: 0.7400 - val_loss: 0.5303
Epoch 14/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.6995 - loss:
0.5829 - val_accuracy: 0.7520 - val_loss: 0.5305
Epoch 15/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.6892 - loss:
0.5629 - val_accuracy: 0.7200 - val_loss: 0.5283
Epoch 16/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7037 - loss:
0.5474 - val_accuracy: 0.7240 - val_loss: 0.5295
Epoch 17/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7152 - loss:
0.5373 - val_accuracy: 0.7280 - val_loss: 0.5313
Epoch 18/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7350 - loss:
0.5437 - val_accuracy: 0.7400 - val_loss: 0.5227
Epoch 19/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7063 - loss:
0.5615 - val_accuracy: 0.7480 - val_loss: 0.5172
Epoch 20/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7239 - loss:
0.5382 - val_accuracy: 0.7440 - val_loss: 0.5178
Epoch 21/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7399 - loss:
```

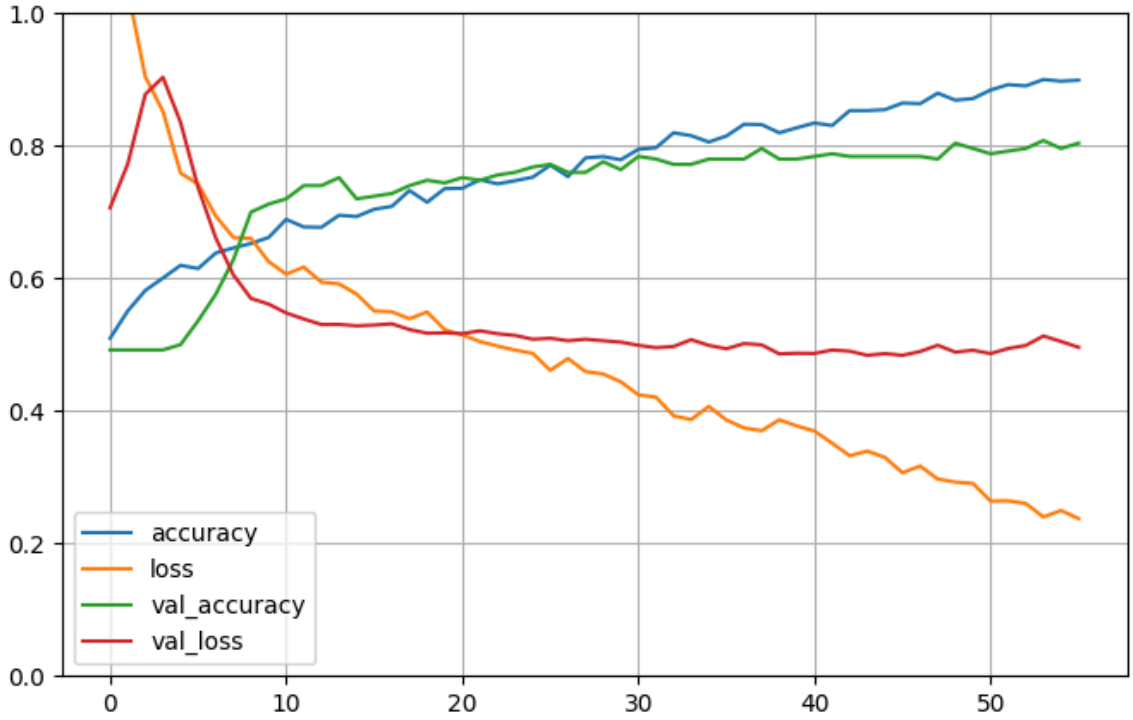
```
0.5182 - val_accuracy: 0.7520 - val_loss: 0.5166
Epoch 22/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7428 - loss:
0.5135 - val_accuracy: 0.7480 - val_loss: 0.5208
Epoch 23/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7393 - loss:
0.5050 - val_accuracy: 0.7560 - val_loss: 0.5165
Epoch 24/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7601 - loss:
0.4790 - val_accuracy: 0.7600 - val_loss: 0.5138
Epoch 25/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7597 - loss:
0.4750 - val_accuracy: 0.7680 - val_loss: 0.5081
Epoch 26/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7668 - loss:
0.4683 - val_accuracy: 0.7720 - val_loss: 0.5096
Epoch 27/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7582 - loss:
0.4726 - val_accuracy: 0.7600 - val_loss: 0.5059
Epoch 28/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7967 - loss:
0.4454 - val_accuracy: 0.7600 - val_loss: 0.5081
Epoch 29/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7784 - loss:
0.4504 - val_accuracy: 0.7760 - val_loss: 0.5059
Epoch 30/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7679 - loss:
0.4612 - val_accuracy: 0.7640 - val_loss: 0.5037
Epoch 31/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7888 - loss:
0.4295 - val_accuracy: 0.7840 - val_loss: 0.4990
Epoch 32/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.7766 - loss:
0.4351 - val_accuracy: 0.7800 - val_loss: 0.4953
Epoch 33/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8112 - loss:
0.3970 - val_accuracy: 0.7720 - val_loss: 0.4972
Epoch 34/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8220 - loss:
0.3747 - val_accuracy: 0.7720 - val_loss: 0.5074
Epoch 35/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8010 - loss:
0.4087 - val_accuracy: 0.7800 - val_loss: 0.4989
```

```
Epoch 36/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8222 - loss:
0.3780 - val_accuracy: 0.7800 - val_loss: 0.4937
Epoch 37/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8269 - loss:
0.3727 - val_accuracy: 0.7800 - val_loss: 0.5016
Epoch 38/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8280 - loss:
0.3725 - val_accuracy: 0.7960 - val_loss: 0.4997
Epoch 39/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - accuracy: 0.8185 - loss:
0.3875 - val_accuracy: 0.7800 - val_loss: 0.4860
Epoch 40/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8262 - loss:
0.3788 - val_accuracy: 0.7800 - val_loss: 0.4870
Epoch 41/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8284 - loss:
0.3830 - val_accuracy: 0.7840 - val_loss: 0.4867
Epoch 42/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8260 - loss:
0.3548 - val_accuracy: 0.7880 - val_loss: 0.4918
Epoch 43/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8539 - loss:
0.3373 - val_accuracy: 0.7840 - val_loss: 0.4901
Epoch 44/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8400 - loss:
0.3493 - val_accuracy: 0.7840 - val_loss: 0.4838
Epoch 45/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8434 - loss:
0.3301 - val_accuracy: 0.7840 - val_loss: 0.4864
Epoch 46/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8570 - loss:
0.3148 - val_accuracy: 0.7840 - val_loss: 0.4838
Epoch 47/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8633 - loss:
0.3270 - val_accuracy: 0.7840 - val_loss: 0.4897
Epoch 48/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8699 - loss:
0.3136 - val_accuracy: 0.7800 - val_loss: 0.4991
Epoch 49/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8646 - loss:
0.2934 - val_accuracy: 0.8040 - val_loss: 0.4886
Epoch 50/9223372036854775807
```

```
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8750 - loss:
0.2942 - val_accuracy: 0.7960 - val_loss: 0.4916
Epoch 51/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8929 - loss:
0.2549 - val_accuracy: 0.7880 - val_loss: 0.4865
Epoch 52/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 34ms/step - accuracy: 0.8973 - loss:
0.2602 - val_accuracy: 0.7920 - val_loss: 0.4941
Epoch 53/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8865 - loss:
0.2626 - val_accuracy: 0.7960 - val_loss: 0.4987
Epoch 54/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.9031 - loss:
0.2282 - val_accuracy: 0.8080 - val_loss: 0.5130
Epoch 55/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.8945 - loss:
0.2578 - val_accuracy: 0.7960 - val_loss: 0.5047
Epoch 56/9223372036854775807
66/66 ━━━━━━━━━━━━━━━━━━━━ 2s 33ms/step - accuracy: 0.9101 - loss:
0.2264 - val_accuracy: 0.8040 - val_loss: 0.4959
```

We can observe the different values that take the orders assigned by us in the following graph, we can observe that the val loss and val take similar trends as well as accuracy and val accuracy.

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



Evaluating autism detection capability

Finally, we will evaluate the model once the training is finished.

```
evaluation=model.evaluate(X_test, y_test)
accuracy = evaluation[1]
val_loss = evaluation[0]
# val_accuracy = evaluation[3]
print("Accuracy:", accuracy)
print("Validation Loss:", val_loss)
# print("Validation Accuracy:", val_accuracy)
```

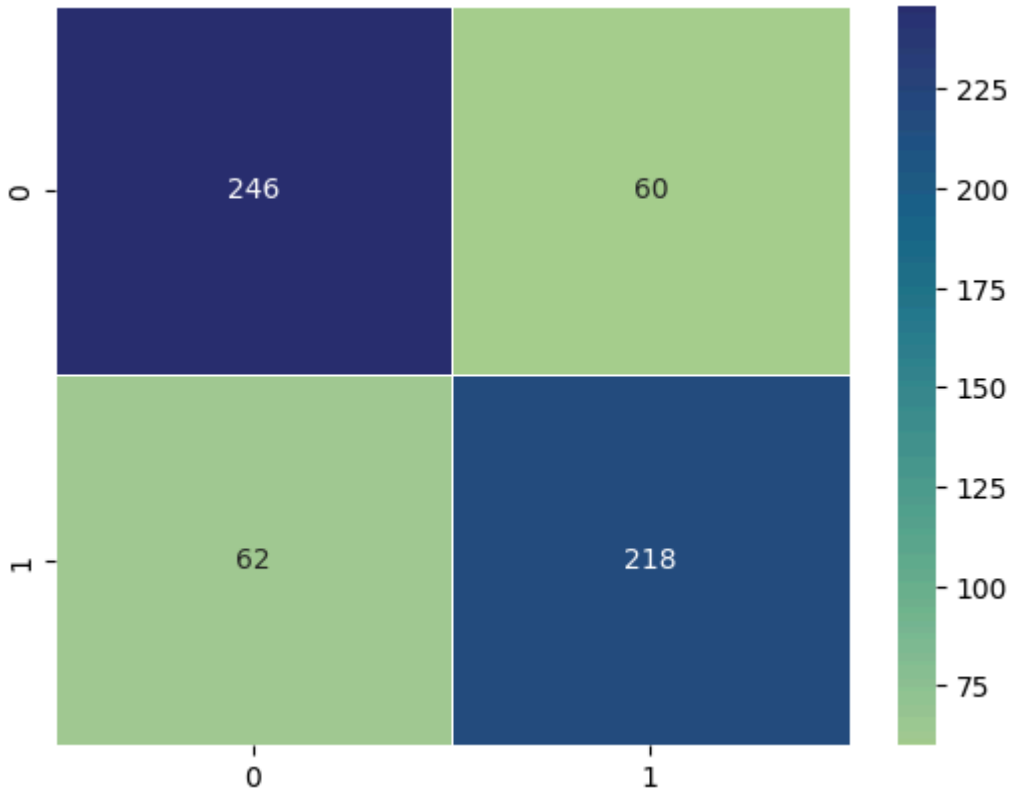
```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.7989 - loss:
0.4438
Accuracy: 0.7918089032173157
Validation Loss: 0.4540463089942932
```

Here we can see in a confusion matrix the data that have been correct in the diagonal and in the inverse diagonal the data that have been wrongly assigned by the model.

```
y_test_pred = model.predict(X_test)
y_test_pred_labels = y_test_pred > 0.5
y_test_true_labels = y_test
cm = confusion_matrix(y_test_true_labels, y_test_pred_labels)
```

```
sns.heatmap(cm, annot=True, fmt='d',cmap="crest",linewidths=.5)
plt.show()
```

```
19/19 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step
```



The `classification_report` function generates a detailed report that includes various metrics to evaluate the performance of a classification model. It compares the true labels (`y_test_true_labels`) with the predicted labels (`y_test_pred_labels`) and computes metrics such as precision, recall, F1-score, and support for each class. These metrics provide insights into how well the model is performing for each class in the classification problem.

```
report = classification_report(y_test_true_labels, y_test_pred_labels,
        digits=4)
print(report)
```

```
              precision    recall  f1-score   support

           0     0.7987    0.8039    0.8013       306
           1     0.7842    0.7786    0.7814       280

    accuracy                         0.7918       586
   macro avg     0.7914    0.7912    0.7913       586
weighted avg     0.7918    0.7918    0.7918       586
```

```
import matplotlib.pyplot as plt

def plot_image_with_labels(image, true_label, pred_label, ax):
    ax.imshow(image, cmap='gray')
    ax.axis('off')
    ax.set_title(f'Real: {true_label}\nPredicted: {pred_label}',
        fontsize=10)

fig, axs = plt.subplots(nrows=20, ncols=5, figsize=(12, 30))

for i in range(100):
    image = x_test[i]
    true_label = 'autist' if y_test_true_labels[i] == 0 else
        'non_autist'
    pred_label = 'autist' if y_test_pred_labels[i] == 0 else
        'non_autist'
    plot_image_with_labels(image, true_label, pred_label, axs[i // 5,
        i % 5])

for row in axs:
    for ax in row:
        ax.axis('off')

plt.tight_layout()
plt.show()
```


Real: non_autist Predicted: non_autist
Real: non_autist Predicted: autist
Real: non_autist Predicted: autist
Real: autist Predicted: autist
Real: autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: non_autist Predicted: autist
Real: autist Predicted: non_autist
Real: autist Predicted: autist
Real: autist Predicted: non_autist
Real: autist Predicted: autist
Real: autist Predicted: autist
Real: autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: non_autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: non_autist Predicted: autist
Real: autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: non_autist Predicted: non_autist
Real: non_autist Predicted: non_autist
Real: autist Predicted: autist
Real: autist Predicted: non_autist
Real: non_autist Predicted: non_autist
Real: non_autist Predicted: non_autist
Real: non_autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: autist Predicted: autist
Real: autist Predicted: autist
Real: autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: autist Predicted: non_autist
Real: autist Predicted: autist
Real: autist Predicted: non_autist
Real: non_autist Predicted: non_autist
Real: autist Predicted: autist
Real: autist Predicted: autist
Real: autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: non_autist Predicted: non_autist
Real: non_autist Predicted: non_autist
Real: autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: autist Predicted: autist
Real: autist Predicted: non_autist
Real: autist Predicted: autist
Real: autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: autist Predicted: autist
Real: autist Predicted: autist
Real: non_autist Predicted: non_autist
Real: non_autist Predicted: non_autist
Real: autist Predicted: autist
Real: autist Predicted: autist
Real: autist Predicted: non_autist
Real: autist Predicted: autist
Real: autist Predicted: non_autist
Real: non_autist Predicted: non_autist
Real: autist Predicted: autist

Real: autist
Predicted: non_autist

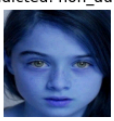

Real: autist
Predicted: non_autist


Real: autist
Predicted: autist


Real: non_autist
Predicted: non_autist

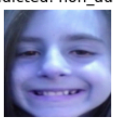

Real: non_autist
Predicted: non_autist


Real: autist
Predicted: autist


Real: non_autist
Predicted: non_autist


Real: non_autist
Predicted: non_autist


Real: autist
Predicted: autist


Real: non_autist
Predicted: non_autist


Real: autist
Predicted: autist


Real: non_autist
Predicted: non_autist


Real: non_autist
Predicted: non_autist


Real: autist
Predicted: autist


Real: non_autist
Predicted: autist


Real: non_autist
Predicted: autist


Real: non_autist
Predicted: non_autist


Real: autist
Predicted: autist


Real: autist
Predicted: autist


Real: non_autist
Predicted: non_autist


Real: non_autist
Predicted: autist


Real: autist
Predicted: autist


Real: autist
Predicted: autist

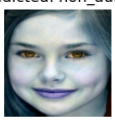

Real: non_autist
Predicted: non_autist


Real: autist
Predicted: autist


Real: autist
Predicted: autist


Real: non_autist
Predicted: non_autist


Real: autist
Predicted: non_autist


Real: non_autist
Predicted: non_autist


Real: non_autist
Predicted: non_autist


Real: autist
Predicted: autist


Real: autist
Predicted: autist


Real: autist
Predicted: autist


Real: non_autist
Predicted: non_autist


Real: non_autist
Predicted: non_autist


Real: autist
Predicted: autist

Real: autist
Predicted: autist

Real: autist
Predicted: autist

Real: autist
Predicted: autist

Real: non_autist
Predicted: non_autist