Deep Learning Project

February 27, 2024

Deep Learning Project - Image classification for Emotion Detection

Fahad Ahmad

Libraries needed for the Project

[2]:
```
!pip install numpy
!pip install opencv-python
!pip install keras
!pip3 install --upgrade tensorflow
!pip install pillow
!pip install tensorflow
!pip install tensorflow==2.8.0
```

```
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-
info due to invalid metadata entry 'name'
Requirement already satisfied: numpy in /opt/conda/lib/python3.11/site-
packages (1.26.2)
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-
info due to invalid metadata entry 'name'
WARNING: Skipping /opt/conda/lib/python3.11/site-
packages/nlopt-2.7.1.dist-info due to invalid metadata entry 'name'
Collecting opencv-python
  Using cached opencv_python-4.9.0.80-cp37-abi3-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (20 kB)
Requirement already satisfied: numpy>=1.21.2 in /opt/conda/lib/python3.11/site-
packages (from opencv-python) (1.26.2)
Using cached
opencv_python-4.9.0.80-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(62.2 MB)
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-
info due to invalid metadata entry 'name'
Installing collected packages: opencv-python
Successfully installed opencv-python-4.9.0.80
```

```
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-
info due to invalid metadata entry 'name'
Collecting keras
  Using cached keras-3.0.5-py3-none-any.whl.metadata (4.8 kB)
Collecting absl-py (from keras)
  Using cached absl_py-2.1.0-py3-none-any.whl.metadata (2.3 kB)
Requirement already satisfied: numpy in /opt/conda/lib/python3.11/site-packages
(from keras) (1.26.2)
Requirement already satisfied: rich in /opt/conda/lib/python3.11/site-packages
(from keras) (13.7.0)
Collecting namex (from keras)
  Downloading namex-0.0.7-py3-none-any.whl.metadata (246 bytes)
Requirement already satisfied: h5py in /opt/conda/lib/python3.11/site-packages
(from keras) (3.10.0)
Collecting dm-tree (from keras)
  Downloading dm_tree-0.1.8-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (1.9 kB)
Collecting ml-dtypes (from keras)
  Using cached ml_dtypes-0.3.2-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (20 kB)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/opt/conda/lib/python3.11/site-packages (from rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/opt/conda/lib/python3.11/site-packages (from rich->keras) (2.17.2)
Requirement already satisfied: mdurl~=0.1 in /opt/conda/lib/python3.11/site-
packages (from markdown-it-py>=2.2.0->rich->keras) (0.1.0)
Using cached keras-3.0.5-py3-none-any.whl (1.0 MB)
Using cached absl_py-2.1.0-py3-none-any.whl (133 kB)
Using cached
dm_tree-0.1.8-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (152
kB)
Using cached
ml_dtypes-0.3.2-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.2
MB)
Using cached namex-0.0.7-py3-none-any.whl (5.8 kB)
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-
info due to invalid metadata entry 'name'
Installing collected packages: namex, dm-tree, ml-dtypes, absl-py, keras
Successfully installed absl-py-2.1.0 dm-tree-0.1.8 keras-3.0.5 ml-dtypes-0.3.2
namex-0.0.7
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-
info due to invalid metadata entry 'name'
Collecting tensorflow
  Using cached tensorflow-2.15.0.post1-cp311-cp311-
```

```
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.2 kB)
Requirement already satisfied: absl-py>=1.0.0 in /opt/conda/lib/python3.11/site-
packages (from tensorflow) (2.1.0)
Collecting astunparse>=1.6.0 (from tensorflow)
  Using cached astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting flatbuffers>=23.5.26 (from tensorflow)
  Using cached flatbuffers-23.5.26-py2.py3-none-any.whl.metadata (850 bytes)
Collecting gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 (from tensorflow)
  Using cached gast-0.5.4-py3-none-any.whl.metadata (1.3 kB)
Collecting google-pasta>=0.1.1 (from tensorflow)
  Using cached google_pasta-0.2.0-py3-none-any.whl (57 kB)
Requirement already satisfied: h5py>=2.9.0 in /opt/conda/lib/python3.11/site-
packages (from tensorflow) (3.10.0)
Collecting libclang>=13.0.0 (from tensorflow)
  Using cached libclang-16.0.6-py2.py3-none-manylinux2010_x86_64.whl.metadata
(5.2 kB)
Collecting ml-dtypes~=0.2.0 (from tensorflow)
  Using cached ml_dtypes-0.2.0-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (20 kB)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (1.26.2)
Collecting opt-einsum>=2.3.2 (from tensorflow)
  Using cached opt_einsum-3.3.0-py3-none-any.whl (65 kB)
Requirement already satisfied: packaging in /opt/conda/lib/python3.11/site-
packages (from tensorflow) (23.2)
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 /opt/conda/lib/python3.11/site-packages (from tensorflow) (4.24.4)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.11/site-
packages (from tensorflow) (68.2.2)
Requirement already satisfied: six>=1.12.0 in /opt/conda/lib/python3.11/site-
packages (from tensorflow) (1.16.0)
Collecting termcolor>=1.1.0 (from tensorflow)
  Using cached termcolor-2.4.0-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: typing-extensions>=3.6.6 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (4.8.0)
Collecting wrapt<1.15,>=1.11.0 (from tensorflow)
  Using cached wrapt-1.14.1-cp311-cp311-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl.metadata (6.7 kB)
Collecting tensorflow-io-gcs-filesystem>=0.23.1 (from tensorflow)
  Using cached tensorflow_io_gcs_filesystem-0.36.0-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (14 kB)
Collecting grpcio<2.0,>=1.24.3 (from tensorflow)
  Downloading grpcio-1.62.0-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (4.0 kB)
Collecting tensorboard<2.16,>=2.15 (from tensorflow)
  Using cached tensorboard-2.15.2-py3-none-any.whl.metadata (1.7 kB)
```

```
Collecting tensorflow-estimator<2.16,>=2.15.0 (from tensorflow)
  Using cached tensorflow_estimator-2.15.0-py2.py3-none-any.whl.metadata (1.3
kB)
Collecting keras<2.16,>=2.15.0 (from tensorflow)
  Using cached keras-2.15.0-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/opt/conda/lib/python3.11/site-packages (from astunparse>=1.6.0->tensorflow)
(0.41.2)
Collecting google-auth<3,>=1.6.3 (from tensorboard<2.16,>=2.15->tensorflow)
  Downloading google_auth-2.28.1-py2.py3-none-any.whl.metadata (4.7 kB)
Collecting google-auth-oauthlib<2,>=0.5 (from
tensorboard<2.16,>=2.15->tensorflow)
  Using cached google_auth_oauthlib-1.2.0-py2.py3-none-any.whl.metadata (2.7 kB)
Requirement already satisfied: markdown>=2.6.8 in
/opt/conda/lib/python3.11/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (3.5.1)
Requirement already satisfied: requests<3,>=2.21.0 in
/opt/conda/lib/python3.11/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (2.28.2)
Collecting tensorboard-data-server<0.8.0,>=0.7.0 (from
tensorboard<2.16,>=2.15->tensorflow)
  Using cached tensorboard_data_server-0.7.2-py3-none-
manylinux_2_31_x86_64.whl.metadata (1.1 kB)
Collecting werkzeug>=1.0.1 (from tensorboard<2.16,>=2.15->tensorflow)
  Using cached werkzeug-3.0.1-py3-none-any.whl.metadata (4.1 kB)
Collecting cachetools<6.0,>=2.0.0 (from google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow)
  Downloading cachetools-5.3.3-py3-none-any.whl.metadata (5.3 kB)
Collecting pyasn1-modules>=0.2.1 (from google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow)
  Using cached pyasn1_modules-0.3.0-py2.py3-none-any.whl.metadata (3.6 kB)
Collecting rsa<5,>=3.1.4 (from google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow)
  Downloading rsa-4.9-py3-none-any.whl.metadata (4.2 kB)
Collecting requests-oauthlib>=0.7.0 (from google-auth-
oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow)
  Downloading requests_oauthlib-1.3.1-py2.py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.11/site-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.11/site-
packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (3.6)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/opt/conda/lib/python3.11/site-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (1.26.18)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.11/site-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (2023.11.17)
```

```
Requirement already satisfied: MarkupSafe>=2.1.1 in
/opt/conda/lib/python3.11/site-packages (from
werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow) (2.1.3)
Collecting pyasn1<0.6.0,>=0.4.6 (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow)
  Using cached pyasn1-0.5.1-py2.py3-none-any.whl.metadata (8.6 kB)
Requirement already satisfied: oauthlib>=3.0.0 in
/opt/conda/lib/python3.11/site-packages (from requests-oauthlib>=0.7.0->google-
auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (3.2.2)
Using cached tensorflow-2.15.0.post1-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (475.3 MB)
Using cached flatbuffers-23.5.26-py2.py3-none-any.whl (26 kB)
Using cached gast-0.5.4-py3-none-any.whl (19 kB)
Downloading
grpcio-1.62.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (5.5
MB)
                           5.5/5.5 MB
4.1 MB/s eta 0:00:00:00:0100:01
Using cached keras-2.15.0-py3-none-any.whl (1.7 MB)
Using cached libclang-16.0.6-py2.py3-none-manylinux2010_x86_64.whl (22.9 MB)
Using cached
ml_dtypes-0.2.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.0
MB)
Using cached tensorboard-2.15.2-py3-none-any.whl (5.5 MB)
Using cached tensorflow_estimator-2.15.0-py2.py3-none-any.whl (441 kB)
Using cached tensorflow_io_gcs_filesystem-0.36.0-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (5.1 MB)
Using cached termcolor-2.4.0-py3-none-any.whl (7.7 kB)
Using cached wrapt-1.14.1-cp311-cp311-
manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (78 kB)
Downloading google_auth-2.28.1-py2.py3-none-any.whl (186 kB)
                           186.9/186.9 kB
531.6 kB/s eta 0:00:0000:01
Using cached google_auth_oauthlib-1.2.0-py2.py3-none-any.whl (24 kB)
Using cached tensorboard_data_server-0.7.2-py3-none-manylinux_2_31_x86_64.whl
(6.6 MB)
Using cached werkzeug-3.0.1-py3-none-any.whl (226 kB)
Downloading cachetools-5.3.3-py3-none-any.whl (9.3 kB)
Using cached pyasn1_modules-0.3.0-py2.py3-none-any.whl (181 kB)
Using cached requests_oauthlib-1.3.1-py2.py3-none-any.whl (23 kB)
Using cached rsa-4.9-py3-none-any.whl (34 kB)
Using cached pyasn1-0.5.1-py2.py3-none-any.whl (84 kB)
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-
info due to invalid metadata entry 'name'
Installing collected packages: libclang, flatbuffers, wrapt, werkzeug,
```

```
termcolor, tensorflow-io-gcs-filesystem, tensorflow-estimator, tensorboard-data-
server, pyasn1, opt-einsum, ml-dtypes, keras, grpcio, google-pasta, gast,
cachetools, astunparse, rsa, requests-oauthlib, pyasn1-modules, google-auth,
google-auth-oauthlib, tensorboard, tensorflow
  Attempting uninstall: ml-dtypes
    Found existing installation: ml-dtypes 0.3.2
    Uninstalling ml-dtypes-0.3.2:
      Successfully uninstalled ml-dtypes-0.3.2
  Attempting uninstall: keras
    Found existing installation: keras 3.0.5
    Uninstalling keras-3.0.5:
      Successfully uninstalled keras-3.0.5
Successfully installed astunparse-1.6.3 cachetools-5.3.3 flatbuffers-23.5.26
gast-0.5.4 google-auth-2.28.1 google-auth-oauthlib-1.2.0 google-pasta-0.2.0
grpcio-1.62.0 keras-2.15.0 libclang-16.0.6 ml-dtypes-0.2.0 opt-einsum-3.3.0
pyasn1-0.5.1 pyasn1-modules-0.3.0 requests-oauthlib-1.3.1 rsa-4.9
tensorboard-2.15.2 tensorboard-data-server-0.7.2 tensorflow-2.15.0.post1
tensorflow-estimator-2.15.0 tensorflow-io-gcs-filesystem-0.36.0 termcolor-2.4.0
werkzeug-3.0.1 wrapt-1.14.1
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-

info due to invalid metadata entry 'name'

Requirement already satisfied: pillow in /opt/conda/lib/python3.11/site-
packages (10.0.1)
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-

info due to invalid metadata entry 'name'

WARNING: Skipping /opt/conda/lib/python3.11/site-

packages/nlopt-2.7.1.dist-info due to invalid metadata entry 'name'

Requirement already satisfied: tensorflow in /opt/conda/lib/python3.11/site-
packages (2.15.0.post1)
Requirement already satisfied: absl-py>=1.0.0 in /opt/conda/lib/python3.11/site-
packages (from tensorflow) (2.1.0)
Requirement already satisfied: astunparse>=1.6.0 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (23.5.26)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /opt/conda/lib/python3.11/site-
packages (from tensorflow) (3.10.0)
Requirement already satisfied: libclang>=13.0.0 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (16.0.6)
Requirement already satisfied: ml-dtypes~=0.2.0 in
```

```
/opt/conda/lib/python3.11/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (1.26.2)
Requirement already satisfied: opt-einsum>=2.3.2 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /opt/conda/lib/python3.11/site-
packages (from tensorflow) (23.2)
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 /opt/conda/lib/python3.11/site-packages (from tensorflow) (4.24.4)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.11/site-
packages (from tensorflow) (68.2.2)
Requirement already satisfied: six>=1.12.0 in /opt/conda/lib/python3.11/site-
packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (4.8.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (0.36.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (1.62.0)
Requirement already satisfied: tensorboard<2.16,>=2.15 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in
/opt/conda/lib/python3.11/site-packages (from tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/opt/conda/lib/python3.11/site-packages (from astunparse>=1.6.0->tensorflow)
(0.41.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/opt/conda/lib/python3.11/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (2.28.1)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in
/opt/conda/lib/python3.11/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in
/opt/conda/lib/python3.11/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (3.5.1)
Requirement already satisfied: requests<3,>=2.21.0 in
/opt/conda/lib/python3.11/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (2.28.2)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/opt/conda/lib/python3.11/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (0.7.2)
```

```
Requirement already satisfied: werkzeug>=1.0.1 in
/opt/conda/lib/python3.11/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (3.0.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/opt/conda/lib/python3.11/site-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
/opt/conda/lib/python3.11/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.3.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.11/site-
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
/opt/conda/lib/python3.11/site-packages (from google-auth-
oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/opt/conda/lib/python3.11/site-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.11/site-
packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (3.6)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/opt/conda/lib/python3.11/site-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (1.26.18)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.11/site-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (2023.11.17)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/opt/conda/lib/python3.11/site-packages (from
werkzeug>=1.0.1->tensorboard<2.16,>=2.15->tensorflow) (2.1.3)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in
/opt/conda/lib/python3.11/site-packages (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.5.1)
Requirement already satisfied: oauthlib>=3.0.0 in
/opt/conda/lib/python3.11/site-packages (from requests-oauthlib>=0.7.0->google-
auth-oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (3.2.2)
```

```
WARNING: Skipping /opt/conda/lib/python3.11/site-packages/nlopt-2.7.1.dist-
info due to invalid metadata entry 'name'
WARNING: Skipping /opt/conda/lib/python3.11/site-
packages/nlopt-2.7.1.dist-info due to invalid metadata entry 'name'
ERROR: Could not find a version that satisfies the requirement
tensorflow==2.8.0 (from versions: 2.12.0rc0, 2.12.0rc1, 2.12.0, 2.12.1,
2.13.0rc0, 2.13.0rc1, 2.13.0rc2, 2.13.0, 2.13.1, 2.14.0rc0, 2.14.0rc1, 2.14.0,
2.14.1, 2.15.0rc0, 2.15.0rc1, 2.15.0, 2.15.0.post1, 2.16.0rc0)
ERROR: No matching distribution found for tensorflow==2.8.0
```

Simple Neural Network

[3]:
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
2024-02-27 03:04:46.745628: 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`.
2024-02-27 03:04:46.799554: I external/local_tsl/tsl/cuda/cudart_stub.cc:31]
Could not find cuda drivers on your machine, GPU will not be used.
2024-02-27 03:04:46.922601: 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
2024-02-27 03:04:46.922648: 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
2024-02-27 03:04:46.932655: 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
2024-02-27 03:04:46.978595: I external/local_tsl/tsl/cuda/cudart_stub.cc:31]
Could not find cuda drivers on your machine, GPU will not be used.
2024-02-27 03:04:46.980134: 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.
```

```
2024-02-27 03:04:48.197234: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not
find TensorRT
```

[4]:
```
# Set the path to your dataset
train_dir = 'images/train'
validation_dir = 'images/test'

# Initialize the data generators
train_datagen = ImageDataGenerator(rescale=1./255)
validation_datagen = ImageDataGenerator(rescale=1./255)

# Load images from directories
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(48, 48),  # Assuming all images are resized to 48x48 pixels
    batch_size=32,
    color_mode='grayscale',  # Assuming images are grayscale
    class_mode='categorical')

validation_generator = validation_datagen.flow_from_directory(
    validation_dir,
    target_size=(48, 48),
    batch_size=32,
    color_mode='grayscale',
    class_mode='categorical')
```

```
Found 28822 images belonging to 7 classes.
Found 7066 images belonging to 7 classes.
```

[5]:
```
model = Sequential([
    Flatten(input_shape=(48, 48, 1)),  # Adjust based on your input image size
    Dense(128, activation='relu'),  # Hidden layer with 128 neurons
    Dense(7, activation='softmax')  # Output layer with 7 neurons (for 7␣
 ↪emotions)
])
```

[6]:
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

[7]:
```
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=validation_generator)
```

```
Epoch 1/10
901/901 [==============================] - 216s 239ms/step - loss: 1.7779 -
```

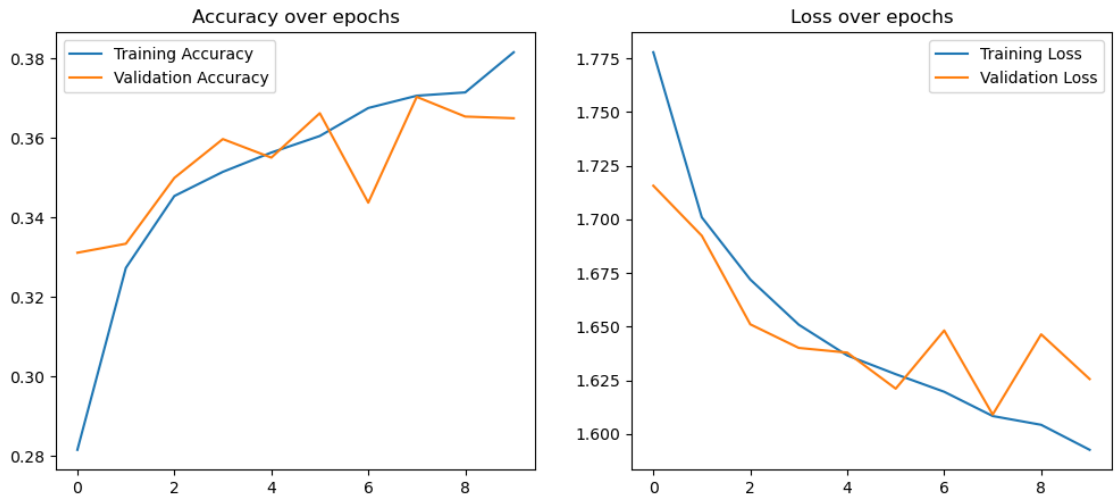
```
accuracy: 0.2816 - val_loss: 1.7157 - val_accuracy: 0.3312
Epoch 2/10
901/901 [==============================] - 186s 207ms/step - loss: 1.7010 -
accuracy: 0.3274 - val_loss: 1.6924 - val_accuracy: 0.3334
Epoch 3/10
901/901 [==============================] - 181s 200ms/step - loss: 1.6720 -
accuracy: 0.3454 - val_loss: 1.6511 - val_accuracy: 0.3500
Epoch 4/10
901/901 [==============================] - 174s 193ms/step - loss: 1.6509 -
accuracy: 0.3515 - val_loss: 1.6401 - val_accuracy: 0.3598
Epoch 5/10
901/901 [==============================] - 196s 218ms/step - loss: 1.6366 -
accuracy: 0.3564 - val_loss: 1.6379 - val_accuracy: 0.3551
Epoch 6/10
901/901 [==============================] - 221s 246ms/step - loss: 1.6278 -
accuracy: 0.3605 - val_loss: 1.6211 - val_accuracy: 0.3663
Epoch 7/10
901/901 [==============================] - 191s 212ms/step - loss: 1.6196 -
accuracy: 0.3676 - val_loss: 1.6482 - val_accuracy: 0.3438
Epoch 8/10
901/901 [==============================] - 226s 252ms/step - loss: 1.6083 -
accuracy: 0.3707 - val_loss: 1.6090 - val_accuracy: 0.3704
Epoch 9/10
901/901 [==============================] - 202s 223ms/step - loss: 1.6042 -
accuracy: 0.3715 - val_loss: 1.6464 - val_accuracy: 0.3654
Epoch 10/10
901/901 [==============================] - 208s 231ms/step - loss: 1.5926 -
accuracy: 0.3816 - val_loss: 1.6256 - val_accuracy: 0.3650
```

[8]:
```python
val_loss, val_accuracy = model.evaluate(validation_generator)
print(f"Validation loss: {val_loss}")
print(f"Validation accuracy: {val_accuracy}")
```

```
221/221 [==============================] - 41s 184ms/step - loss: 1.6256 -
accuracy: 0.3650
Validation loss: 1.6255671977996826
Validation accuracy: 0.36498725414276123
```

[9]:
```python
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy over epochs')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Training Loss')
```

```
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Loss over epochs')
plt.legend()

plt.show()
```



In our deep learning project focused on emotion detection, for a simple model we employed a straightforward neural network architecture to classify images into one of seven emotions. The data preprocessing was executed using ImageDataGenerators for both training and validation datasets, which rescaled the images to a consistent size of 48x48 pixels and converted them to grayscale to reduce complexity while preserving the essential features for emotion recognition. Our model's architecture began with a Flatten layer to transform the 2D image arrays into a 1D vector. This was followed by a dense layer containing 128 neurons with ReLU activation to enable the learning of complex patterns through non-linear transformations. The output layer consisted of 7 neurons, each representing a different emotion, and used softmax activation to generate a probability distribution across these classes.

The model was compiled with the Adam optimizer and the categorical crossentropy loss function, which are commonly used in multi-class classification tasks. After training over 10 epochs, the model reached a validation accuracy of approximately 36.5% and a validation loss of 1.626. These outcomes suggest that while the model can identify emotions to a certain degree, its performance is modest, highlighting the model's limitations in effectively discerning emotions from images.

[ ]:
```
######################
```

Moving to complex models

Convolutional Neural Network

[12]:
```
# import required packages
import cv2
from keras.models import Sequential
```

```
from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
```

Preprocessing of Data for CNN and CNN Model

[13]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten,␣
 ↪Dense
from tensorflow.keras.optimizers import Adam
import cv2

# Initialize image data generator with rescaling
train_data_gen = ImageDataGenerator(rescale=1./255)
validation_data_gen = ImageDataGenerator(rescale=1./255)

# Preprocess all training images
train_generator = train_data_gen.flow_from_directory(
        'images/train',
        target_size=(48, 48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical')

# Preprocess all validation images
validation_generator = validation_data_gen.flow_from_directory(
        'images/test',
        target_size=(48, 48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical')

# Create model structure
emotion_model = Sequential()

emotion_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',␣
 ↪input_shape=(48, 48, 1)))
emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
```

```
emotion_model.add(Flatten())
emotion_model.add(Dense(1024, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation='softmax'))

cv2.ocl.setUseOpenCL(False)

# Compile the model with updated parameters
emotion_model.compile(loss='categorical_crossentropy',␣
 ↪optimizer=Adam(learning_rate=0.0001), metrics=['accuracy'])
```

```
Found 28822 images belonging to 7 classes.
Found 7066 images belonging to 7 classes.
```

This code outlines the creation of a Convolutional Neural Network (CNN) for facial emotion recognition, leveraging a dataset of 48x48 pixel grayscale images. It employs a sequential model architecture with layers designed for feature extraction—specifically, convolutional layers with ReLU activation followed by max pooling to reduce dimensionality, and dropout layers to prevent overfitting, with dropout rates of 0.25 and 0.5 at different stages of the model.

The network finalizes with a dense layer of 1024 neurons, leading to a 7-neuron output layer with softmax activation for classifying images into seven emotion categories. This setup uses the Adam optimizer with a learning rate of 0.0001 and aims to minimize the categorical crossentropy loss, focusing on achieving high accuracy in emotion detection from facial images.

[14]:
```
# Train the neural network/model using `fit` instead of `fit_generator`
emotion_model_info = emotion_model.fit(
        train_generator,
        steps_per_epoch=28709 // 64,
        epochs=50,
        validation_data=validation_generator,
        validation_steps=7178 // 64)
```

```
Epoch 1/50
448/448 [==============================] - ETA: 0s - loss: 1.8011 - accuracy:
0.2570WARNING:tensorflow:Your input ran out of data; interrupting training. Make
sure that your dataset or generator can generate at least `steps_per_epoch *
epochs` batches (in this case, 112 batches). You may need to use the repeat()
function when building your dataset.
448/448 [==============================] - 242s 537ms/step - loss: 1.8011 -
accuracy: 0.2570 - val_loss: 1.6986 - val_accuracy: 0.3558
Epoch 2/50
448/448 [==============================] - 239s 534ms/step - loss: 1.6167 -
accuracy: 0.3730
Epoch 3/50
448/448 [==============================] - 217s 485ms/step - loss: 1.5106 -
```

```
accuracy: 0.4186
Epoch 4/50
448/448 [==============================] - 208s 464ms/step - loss: 1.4349 -
accuracy: 0.4498
Epoch 5/50
448/448 [==============================] - 218s 486ms/step - loss: 1.3743 -
accuracy: 0.4771
Epoch 6/50
448/448 [==============================] - 214s 478ms/step - loss: 1.3250 -
accuracy: 0.4954
Epoch 7/50
448/448 [==============================] - 222s 496ms/step - loss: 1.2850 -
accuracy: 0.5117
Epoch 8/50
448/448 [==============================] - 218s 488ms/step - loss: 1.2447 -
accuracy: 0.5319
Epoch 9/50
448/448 [==============================] - 228s 508ms/step - loss: 1.2172 -
accuracy: 0.5428
Epoch 10/50
448/448 [==============================] - 214s 477ms/step - loss: 1.1848 -
accuracy: 0.5539
Epoch 11/50
448/448 [==============================] - 224s 500ms/step - loss: 1.1587 -
accuracy: 0.5625
Epoch 12/50
448/448 [==============================] - 217s 484ms/step - loss: 1.1360 -
accuracy: 0.5755
Epoch 13/50
448/448 [==============================] - 217s 484ms/step - loss: 1.1114 -
accuracy: 0.5791
Epoch 14/50
448/448 [==============================] - 216s 482ms/step - loss: 1.0802 -
accuracy: 0.5953
Epoch 15/50
448/448 [==============================] - 235s 525ms/step - loss: 1.0578 -
accuracy: 0.6045
Epoch 16/50
448/448 [==============================] - 225s 503ms/step - loss: 1.0292 -
accuracy: 0.6149
Epoch 17/50
448/448 [==============================] - 220s 490ms/step - loss: 1.0112 -
accuracy: 0.6223
Epoch 18/50
448/448 [==============================] - 224s 500ms/step - loss: 0.9868 -
accuracy: 0.6331
Epoch 19/50
448/448 [==============================] - 228s 508ms/step - loss: 0.9678 -
```

```
accuracy: 0.6426
Epoch 20/50
448/448 [==============================] - 226s 504ms/step - loss: 0.9448 -
accuracy: 0.6487
Epoch 21/50
448/448 [==============================] - 224s 501ms/step - loss: 0.9237 -
accuracy: 0.6607
Epoch 22/50
448/448 [==============================] - 213s 475ms/step - loss: 0.9037 -
accuracy: 0.6664
Epoch 23/50
448/448 [==============================] - 217s 485ms/step - loss: 0.8769 -
accuracy: 0.6740
Epoch 24/50
448/448 [==============================] - 219s 488ms/step - loss: 0.8544 -
accuracy: 0.6867
Epoch 25/50
448/448 [==============================] - 219s 488ms/step - loss: 0.8301 -
accuracy: 0.6953
Epoch 26/50
448/448 [==============================] - 222s 495ms/step - loss: 0.8066 -
accuracy: 0.7015
Epoch 27/50
448/448 [==============================] - 218s 487ms/step - loss: 0.7835 -
accuracy: 0.7122
Epoch 28/50
448/448 [==============================] - 225s 502ms/step - loss: 0.7702 -
accuracy: 0.7171
Epoch 29/50
448/448 [==============================] - 221s 494ms/step - loss: 0.7372 -
accuracy: 0.7301
Epoch 30/50
448/448 [==============================] - 224s 501ms/step - loss: 0.7185 -
accuracy: 0.7389
Epoch 31/50
448/448 [==============================] - 230s 513ms/step - loss: 0.6995 -
accuracy: 0.7457
Epoch 32/50
448/448 [==============================] - 228s 508ms/step - loss: 0.6828 -
accuracy: 0.7498
Epoch 33/50
448/448 [==============================] - 227s 507ms/step - loss: 0.6653 -
accuracy: 0.7567
Epoch 34/50
448/448 [==============================] - 234s 522ms/step - loss: 0.6393 -
accuracy: 0.7669
Epoch 35/50
448/448 [==============================] - 226s 504ms/step - loss: 0.6164 -
```

```
accuracy: 0.7752
Epoch 36/50
448/448 [==============================] - 228s 508ms/step - loss: 0.5967 -
accuracy: 0.7806
Epoch 37/50
448/448 [==============================] - 229s 511ms/step - loss: 0.5738 -
accuracy: 0.7931
Epoch 38/50
448/448 [==============================] - 227s 507ms/step - loss: 0.5586 -
accuracy: 0.7991
Epoch 39/50
448/448 [==============================] - 222s 495ms/step - loss: 0.5409 -
accuracy: 0.8037
Epoch 40/50
448/448 [==============================] - 223s 498ms/step - loss: 0.5182 -
accuracy: 0.8120
Epoch 41/50
448/448 [==============================] - 226s 505ms/step - loss: 0.5064 -
accuracy: 0.8202
Epoch 42/50
448/448 [==============================] - 226s 504ms/step - loss: 0.4817 -
accuracy: 0.8268
Epoch 43/50
448/448 [==============================] - 226s 504ms/step - loss: 0.4759 -
accuracy: 0.8271
Epoch 44/50
448/448 [==============================] - 230s 513ms/step - loss: 0.4540 -
accuracy: 0.8375
Epoch 45/50
448/448 [==============================] - 238s 532ms/step - loss: 0.4410 -
accuracy: 0.8409
Epoch 46/50
448/448 [==============================] - 222s 496ms/step - loss: 0.4299 -
accuracy: 0.8429
Epoch 47/50
448/448 [==============================] - 223s 498ms/step - loss: 0.4200 -
accuracy: 0.8494
Epoch 48/50
448/448 [==============================] - 223s 497ms/step - loss: 0.4004 -
accuracy: 0.8574
Epoch 49/50
448/448 [==============================] - 224s 500ms/step - loss: 0.3908 -
accuracy: 0.8623
Epoch 50/50
448/448 [==============================] - 228s 508ms/step - loss: 0.3755 -
accuracy: 0.8662
```

[16]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Initialize the test data generator
test_data_gen = ImageDataGenerator(rescale=1./255)

# Load test images
test_generator = test_data_gen.flow_from_directory(
        'images/test',  # Path to the test data
        target_size=(48, 48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical',
        shuffle=False)  # Important for later evaluation steps
```

```
Found 7066 images belonging to 7 classes.
```

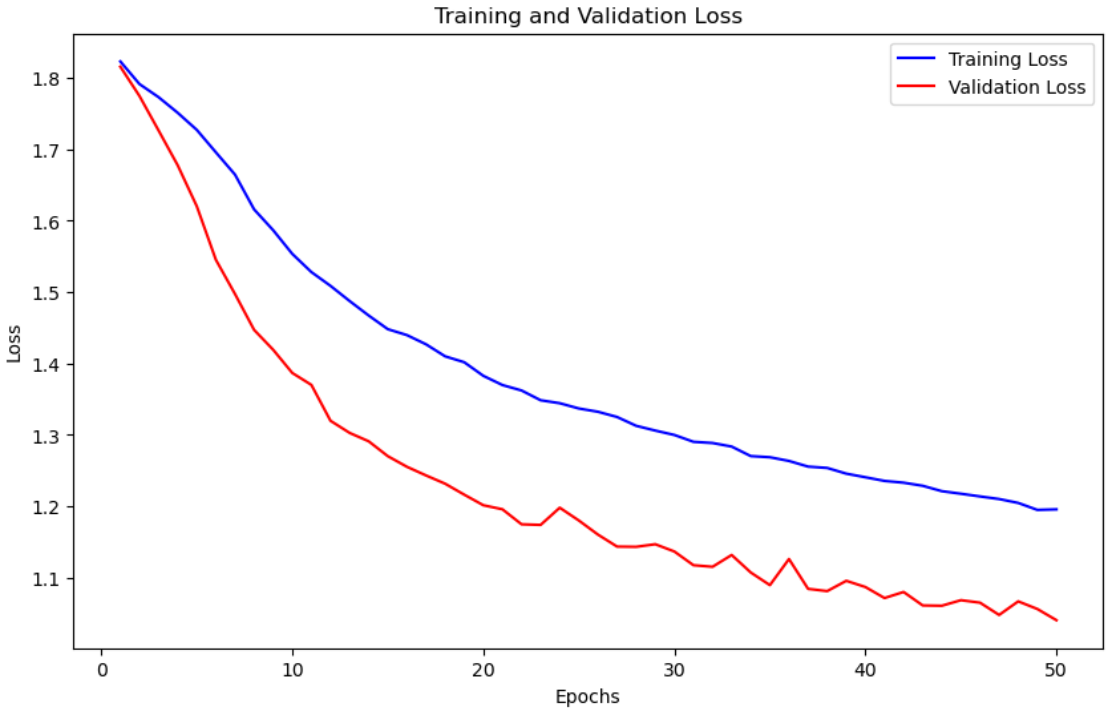
[17]:
```
# Evaluate the model on the test data
test_loss, test_accuracy = emotion_model.evaluate(test_generator,␣
 ↪steps=test_generator.samples // test_generator.batch_size)

print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
110/110 [==============================] - 15s 137ms/step - loss: 1.1978 -
accuracy: 0.6308
Test Loss: 1.1978124380111694
Test Accuracy: 0.6308238506317139
```

The evaluation results show that the model achieves a loss of 1.1978 and an accuracy of 63.08% on the test data. The "loss" measures how well the model's predictions match the actual labels, with lower values indicating better performance. The "accuracy" is the proportion of correctly predicted instances among the total number of predictions, with higher percentages indicating better performance. In this case, the model correctly predicts the facial expression category for approximately 63.08% of the test images, which can be considered as its effectiveness in recognizing facial expressions based on the learned patterns during training.

[39]:
```
import matplotlib.pyplot as plt

# Extracting loss values from the history object
training_loss = emotion_model_info.history['loss']
validation_loss = emotion_model_info.history['val_loss']

# Extracting the number of epochs
epochs = range(1, len(training_loss) + 1)

# Plotting training and validation loss
plt.figure(figsize=(10, 6))
plt.plot(epochs, training_loss, 'b-', label='Training Loss')
```

```
plt.plot(epochs, validation_loss, 'r-', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



[24]:
```
import numpy as np

# Make predictions
predictions = emotion_model.predict(test_generator, steps=test_generator.
 ↪samples // test_generator.batch_size)
predicted_classes = np.argmax(predictions, axis=1)

# Get true labels
true_classes = test_generator.classes
class_labels = list(test_generator.class_indices.keys())  # Getting the mapping␣
 ↪of class indices to labels

# Compare predictions with true labels for analysis
from sklearn.metrics import classification_report
print(classification_report(true_classes[:len(predicted_classes)],␣
 ↪predicted_classes, target_names=class_labels))
```

```
110/110 [==============================] - 16s 145ms/step
              precision    recall  f1-score   support

       angry       0.54      0.55      0.54       960
     disgust       0.71      0.56      0.63       111
        fear       0.59      0.38      0.46      1018
       happy       0.81      0.83      0.82      1825
     neutral       0.54      0.60      0.57      1216
         sad       0.47      0.56      0.51      1139
    surprise       0.75      0.76      0.76       771

    accuracy                           0.63      7040
   macro avg       0.63      0.61      0.61      7040
weighted avg       0.63      0.63      0.63      7040
```

[ ]:
```
##########################
```

Model training with learning rate scheduler to improve accuracy.

[26]:
```
from tensorflow.keras.callbacks import LearningRateScheduler
```

[27]:
```
from tensorflow.keras.callbacks import LearningRateScheduler
import numpy as np

# Define a scheduler function
def scheduler(epoch, lr):
    if epoch < 10:
        return lr
    else:
        return lr * np.exp(-0.1)

# Create a callback for the learning rate scheduler
lr_scheduler = LearningRateScheduler(scheduler)
```

[28]:
```
# Train the neural network/model using `fit` and include the learning rate␣
 ↪scheduler callback
emotion_model_info = emotion_model.fit(
    train_generator,
    steps_per_epoch=28709 // 64,
    epochs=50,
    validation_data=validation_generator,
    validation_steps=7178 // 64,
    callbacks=[lr_scheduler]  # Add the learning rate scheduler callback here
)
```

```
Epoch 1/50
448/448 [==============================] - ETA: 0s - loss: 0.3659 - accuracy:
```

```
0.8696WARNING:tensorflow:Your input ran out of data; interrupting training. Make
sure that your dataset or generator can generate at least `steps_per_epoch *
epochs` batches (in this case, 112 batches). You may need to use the repeat()
function when building your dataset.
448/448 [==============================] - 244s 545ms/step - loss: 0.3659 -
accuracy: 0.8696 - val_loss: 1.2197 - val_accuracy: 0.6316 - lr: 1.0000e-04
Epoch 2/50
448/448 [==============================] - 228s 508ms/step - loss: 0.3568 -
accuracy: 0.8733 - lr: 1.0000e-04
Epoch 3/50
448/448 [==============================] - 231s 515ms/step - loss: 0.3433 -
accuracy: 0.8788 - lr: 1.0000e-04
Epoch 4/50
448/448 [==============================] - 228s 508ms/step - loss: 0.3318 -
accuracy: 0.8812 - lr: 1.0000e-04
Epoch 5/50
448/448 [==============================] - 226s 504ms/step - loss: 0.3265 -
accuracy: 0.8861 - lr: 1.0000e-04
Epoch 6/50
448/448 [==============================] - 223s 497ms/step - loss: 0.3197 -
accuracy: 0.8850 - lr: 1.0000e-04
Epoch 7/50
448/448 [==============================] - 222s 495ms/step - loss: 0.3084 -
accuracy: 0.8899 - lr: 1.0000e-04
Epoch 8/50
448/448 [==============================] - 226s 505ms/step - loss: 0.2964 -
accuracy: 0.8952 - lr: 1.0000e-04
Epoch 9/50
448/448 [==============================] - 224s 502ms/step - loss: 0.2873 -
accuracy: 0.8968 - lr: 1.0000e-04
Epoch 10/50
448/448 [==============================] - 229s 510ms/step - loss: 0.2829 -
accuracy: 0.8992 - lr: 1.0000e-04
Epoch 11/50
448/448 [==============================] - 228s 508ms/step - loss: 0.2705 -
accuracy: 0.9051 - lr: 9.0484e-05
Epoch 12/50
448/448 [==============================] - 224s 499ms/step - loss: 0.2608 -
accuracy: 0.9071 - lr: 8.1873e-05
Epoch 13/50
448/448 [==============================] - 225s 502ms/step - loss: 0.2516 -
accuracy: 0.9112 - lr: 7.4082e-05
Epoch 14/50
448/448 [==============================] - 212s 472ms/step - loss: 0.2427 -
accuracy: 0.9147 - lr: 6.7032e-05
Epoch 15/50
448/448 [==============================] - 217s 485ms/step - loss: 0.2301 -
accuracy: 0.9200 - lr: 6.0653e-05
```

```
Epoch 16/50
448/448 [==============================] - 222s 494ms/step - loss: 0.2230 -
accuracy: 0.9210 - lr: 5.4881e-05
Epoch 17/50
448/448 [==============================] - 227s 507ms/step - loss: 0.2178 -
accuracy: 0.9229 - lr: 4.9659e-05
Epoch 18/50
448/448 [==============================] - 213s 475ms/step - loss: 0.2109 -
accuracy: 0.9270 - lr: 4.4933e-05
Epoch 19/50
448/448 [==============================] - 229s 511ms/step - loss: 0.2067 -
accuracy: 0.9270 - lr: 4.0657e-05
Epoch 20/50
448/448 [==============================] - 231s 516ms/step - loss: 0.1985 -
accuracy: 0.9310 - lr: 3.6788e-05
Epoch 21/50
448/448 [==============================] - 222s 496ms/step - loss: 0.1933 -
accuracy: 0.9330 - lr: 3.3287e-05
Epoch 22/50
448/448 [==============================] - 224s 501ms/step - loss: 0.1907 -
accuracy: 0.9342 - lr: 3.0119e-05
Epoch 23/50
448/448 [==============================] - 220s 491ms/step - loss: 0.1891 -
accuracy: 0.9339 - lr: 2.7253e-05
Epoch 24/50
448/448 [==============================] - 223s 498ms/step - loss: 0.1830 -
accuracy: 0.9375 - lr: 2.4660e-05
Epoch 25/50
448/448 [==============================] - 228s 508ms/step - loss: 0.1800 -
accuracy: 0.9367 - lr: 2.2313e-05
Epoch 26/50
448/448 [==============================] - 220s 490ms/step - loss: 0.1781 -
accuracy: 0.9385 - lr: 2.0190e-05
Epoch 27/50
448/448 [==============================] - 221s 494ms/step - loss: 0.1740 -
accuracy: 0.9394 - lr: 1.8268e-05
Epoch 28/50
448/448 [==============================] - 225s 502ms/step - loss: 0.1741 -
accuracy: 0.9393 - lr: 1.6530e-05
Epoch 29/50
448/448 [==============================] - 226s 505ms/step - loss: 0.1737 -
accuracy: 0.9399 - lr: 1.4957e-05
Epoch 30/50
448/448 [==============================] - 222s 496ms/step - loss: 0.1678 -
accuracy: 0.9411 - lr: 1.3534e-05
Epoch 31/50
448/448 [==============================] - 226s 504ms/step - loss: 0.1730 -
accuracy: 0.9399 - lr: 1.2246e-05
```

```
Epoch 32/50
448/448 [==============================] - 220s 491ms/step - loss: 0.1700 -
accuracy: 0.9416 - lr: 1.1080e-05
Epoch 33/50
448/448 [==============================] - 224s 501ms/step - loss: 0.1685 -
accuracy: 0.9416 - lr: 1.0026e-05
Epoch 34/50
448/448 [==============================] - 222s 495ms/step - loss: 0.1624 -
accuracy: 0.9440 - lr: 9.0718e-06
Epoch 35/50
448/448 [==============================] - 220s 491ms/step - loss: 0.1669 -
accuracy: 0.9423 - lr: 8.2085e-06
Epoch 36/50
448/448 [==============================] - 222s 496ms/step - loss: 0.1654 -
accuracy: 0.9423 - lr: 7.4274e-06
Epoch 37/50
448/448 [==============================] - 223s 498ms/step - loss: 0.1599 -
accuracy: 0.9451 - lr: 6.7206e-06
Epoch 38/50
448/448 [==============================] - 221s 494ms/step - loss: 0.1628 -
accuracy: 0.9428 - lr: 6.0810e-06
Epoch 39/50
448/448 [==============================] - 227s 507ms/step - loss: 0.1640 -
accuracy: 0.9443 - lr: 5.5023e-06
Epoch 40/50
448/448 [==============================] - 226s 505ms/step - loss: 0.1653 -
accuracy: 0.9431 - lr: 4.9787e-06
Epoch 41/50
448/448 [==============================] - 227s 506ms/step - loss: 0.1636 -
accuracy: 0.9426 - lr: 4.5049e-06
Epoch 42/50
448/448 [==============================] - 221s 493ms/step - loss: 0.1613 -
accuracy: 0.9436 - lr: 4.0762e-06
Epoch 43/50
448/448 [==============================] - 219s 490ms/step - loss: 0.1643 -
accuracy: 0.9436 - lr: 3.6883e-06
Epoch 44/50
448/448 [==============================] - 220s 491ms/step - loss: 0.1580 -
accuracy: 0.9454 - lr: 3.3373e-06
Epoch 45/50
448/448 [==============================] - 220s 491ms/step - loss: 0.1637 -
accuracy: 0.9430 - lr: 3.0197e-06
Epoch 46/50
448/448 [==============================] - 217s 484ms/step - loss: 0.1617 -
accuracy: 0.9453 - lr: 2.7324e-06
Epoch 47/50
448/448 [==============================] - 217s 485ms/step - loss: 0.1592 -
accuracy: 0.9442 - lr: 2.4724e-06
```

```
Epoch 48/50
448/448 [==============================] - 225s 503ms/step - loss: 0.1618 -
accuracy: 0.9451 - lr: 2.2371e-06
Epoch 49/50
448/448 [==============================] - 219s 488ms/step - loss: 0.1597 -
accuracy: 0.9448 - lr: 2.0242e-06
Epoch 50/50
448/448 [==============================] - 221s 492ms/step - loss: 0.1589 -
accuracy: 0.9449 - lr: 1.8316e-06
```

[29]:
```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Initialize the test data generator
test_data_gen = ImageDataGenerator(rescale=1./255)

# Load test images
test_generator = test_data_gen.flow_from_directory(
        'images/test',  # Path to the test data
        target_size=(48, 48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical',
        shuffle=False)  # Important for later evaluation steps
```

```
Found 7066 images belonging to 7 classes.
```

[30]:
```python
# Evaluate the model on the test data
test_loss, test_accuracy = emotion_model.evaluate(test_generator,␣
 ↪steps=test_generator.samples // test_generator.batch_size)

print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
110/110 [==============================] - 15s 139ms/step - loss: 1.4224 -
accuracy: 0.6320
Test Loss: 1.42244291305542
Test Accuracy: 0.6319602131843567
```

In our project, following the adoption of a Convolutional Neural Network (CNN) to enhance emotion detection accuracy, we further experimented with a Learning Rate Scheduler to optimize model performance. The Learning Rate Scheduler is advantageous as it dynamically adjusts the learning rate during training, promoting faster convergence and preventing the model from getting stuck in local minima. However, despite these efforts, we did not observe a significant increase in accuracy, maintaining a level similar to the initial CNN results at 63%, indicating the complexity of achieving substantial gains through this method alone.

[40]:
```python
import matplotlib.pyplot as plt
```

```
# Extracting loss values from the history object
training_loss = emotion_model_info.history['loss']
validation_loss = emotion_model_info.history['val_loss']

# Extracting the number of epochs
epochs = range(1, len(training_loss) + 1)

# Plotting training and validation loss
plt.figure(figsize=(10, 6))
plt.plot(epochs, training_loss, 'b-', label='Training Loss')
plt.plot(epochs, validation_loss, 'r-', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

[ ]:
```
##########################
```

Model Training with data augmentation to improve accuracy.

[33]:
```
# Import necessary libraries
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten,␣
↪Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import LearningRateScheduler
import numpy as np

# Initialize image data generators
# For training data with data augmentation
train_data_gen = ImageDataGenerator(
    rescale=1./255,
    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'
)

# For validation data without data augmentation
validation_data_gen = ImageDataGenerator(rescale=1./255)

# Preprocess all training images
train_generator = train_data_gen.flow_from_directory(
    'images/train',
    target_size=(48, 48),
    batch_size=64,
    color_mode="grayscale",
    class_mode='categorical')

# Preprocess all validation images
validation_generator = validation_data_gen.flow_from_directory(
    'images/test',
    target_size=(48, 48),
    batch_size=64,
    color_mode="grayscale",
    class_mode='categorical')

# Create model structure (same as before)
emotion_model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48, 48, 1)),
    Conv2D(64, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),

    Conv2D(128, kernel_size=(3, 3), activation='relu'),
```

```
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(128, kernel_size=(3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Dropout(0.25),

    Flatten(),
    Dense(1024, activation='relu'),
    Dropout(0.5),
    Dense(7, activation='softmax')
])

# Compile the model
emotion_model.compile(loss='categorical_crossentropy',␣
 ↪optimizer=Adam(learning_rate=0.0001), metrics=['accuracy'])

# Train the model with the augmented data
emotion_model_info = emotion_model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    epochs=50,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // validation_generator.
 ↪batch_size
)

# Evaluate the model on the validation set
val_loss, val_accuracy = emotion_model.evaluate(validation_generator,␣
 ↪steps=validation_generator.samples // validation_generator.batch_size)
print(f"Validation Loss: {val_loss}")
print(f"Validation Accuracy: {val_accuracy}")
```

```
Found 28822 images belonging to 7 classes.
Found 7066 images belonging to 7 classes.
Epoch 1/50
450/450 [==============================] - 257s 567ms/step - loss: 1.8229 -
accuracy: 0.2476 - val_loss: 1.8156 - val_accuracy: 0.2661
Epoch 2/50
450/450 [==============================] - 236s 523ms/step - loss: 1.7913 -
accuracy: 0.2507 - val_loss: 1.7746 - val_accuracy: 0.2666
Epoch 3/50
450/450 [==============================] - 241s 536ms/step - loss: 1.7729 -
accuracy: 0.2634 - val_loss: 1.7263 - val_accuracy: 0.3104
Epoch 4/50
450/450 [==============================] - 240s 533ms/step - loss: 1.7511 -
accuracy: 0.2792 - val_loss: 1.6776 - val_accuracy: 0.3330
Epoch 5/50
450/450 [==============================] - 246s 546ms/step - loss: 1.7272 -
```

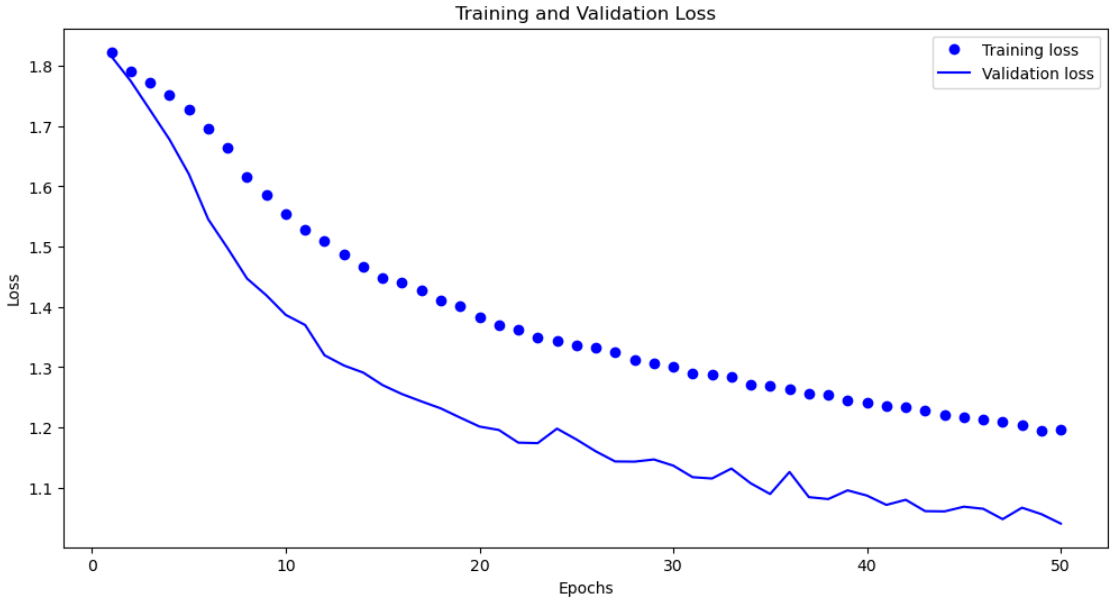
```
accuracy: 0.2987 - val_loss: 1.6202 - val_accuracy: 0.3679
Epoch 6/50
450/450 [==============================] - 241s 537ms/step - loss: 1.6957 -
accuracy: 0.3191 - val_loss: 1.5452 - val_accuracy: 0.4108
Epoch 7/50
450/450 [==============================] - 237s 527ms/step - loss: 1.6645 -
accuracy: 0.3350 - val_loss: 1.4973 - val_accuracy: 0.4278
Epoch 8/50
450/450 [==============================] - 251s 558ms/step - loss: 1.6158 -
accuracy: 0.3674 - val_loss: 1.4471 - val_accuracy: 0.4359
Epoch 9/50
450/450 [==============================] - 247s 549ms/step - loss: 1.5866 -
accuracy: 0.3816 - val_loss: 1.4192 - val_accuracy: 0.4570
Epoch 10/50
450/450 [==============================] - 251s 557ms/step - loss: 1.5535 -
accuracy: 0.3939 - val_loss: 1.3868 - val_accuracy: 0.4776
Epoch 11/50
450/450 [==============================] - 244s 541ms/step - loss: 1.5280 -
accuracy: 0.4079 - val_loss: 1.3700 - val_accuracy: 0.4858
Epoch 12/50
450/450 [==============================] - 245s 544ms/step - loss: 1.5087 -
accuracy: 0.4168 - val_loss: 1.3198 - val_accuracy: 0.4979
Epoch 13/50
450/450 [==============================] - 243s 540ms/step - loss: 1.4875 -
accuracy: 0.4277 - val_loss: 1.3028 - val_accuracy: 0.5060
Epoch 14/50
450/450 [==============================] - 248s 551ms/step - loss: 1.4672 -
accuracy: 0.4368 - val_loss: 1.2910 - val_accuracy: 0.5091
Epoch 15/50
450/450 [==============================] - 245s 544ms/step - loss: 1.4481 -
accuracy: 0.4455 - val_loss: 1.2703 - val_accuracy: 0.5172
Epoch 16/50
450/450 [==============================] - 236s 525ms/step - loss: 1.4398 -
accuracy: 0.4505 - val_loss: 1.2554 - val_accuracy: 0.5280
Epoch 17/50
450/450 [==============================] - 239s 531ms/step - loss: 1.4269 -
accuracy: 0.4562 - val_loss: 1.2433 - val_accuracy: 0.5328
Epoch 18/50
450/450 [==============================] - 247s 548ms/step - loss: 1.4101 -
accuracy: 0.4582 - val_loss: 1.2317 - val_accuracy: 0.5366
Epoch 19/50
450/450 [==============================] - 240s 534ms/step - loss: 1.4018 -
accuracy: 0.4640 - val_loss: 1.2164 - val_accuracy: 0.5436
Epoch 20/50
450/450 [==============================] - 240s 532ms/step - loss: 1.3827 -
accuracy: 0.4742 - val_loss: 1.2015 - val_accuracy: 0.5459
Epoch 21/50
450/450 [==============================] - 245s 544ms/step - loss: 1.3699 -
```

```
accuracy: 0.4773 - val_loss: 1.1959 - val_accuracy: 0.5479
Epoch 22/50
450/450 [==============================] - 242s 539ms/step - loss: 1.3622 -
accuracy: 0.4801 - val_loss: 1.1749 - val_accuracy: 0.5558
Epoch 23/50
450/450 [==============================] - 244s 543ms/step - loss: 1.3485 -
accuracy: 0.4848 - val_loss: 1.1740 - val_accuracy: 0.5562
Epoch 24/50
450/450 [==============================] - 249s 553ms/step - loss: 1.3444 -
accuracy: 0.4866 - val_loss: 1.1981 - val_accuracy: 0.5491
Epoch 25/50
450/450 [==============================] - 248s 552ms/step - loss: 1.3370 -
accuracy: 0.4898 - val_loss: 1.1802 - val_accuracy: 0.5513
Epoch 26/50
450/450 [==============================] - 247s 549ms/step - loss: 1.3324 -
accuracy: 0.4919 - val_loss: 1.1604 - val_accuracy: 0.5628
Epoch 27/50
450/450 [==============================] - 249s 554ms/step - loss: 1.3251 -
accuracy: 0.4953 - val_loss: 1.1436 - val_accuracy: 0.5678
Epoch 28/50
450/450 [==============================] - 246s 548ms/step - loss: 1.3128 -
accuracy: 0.4995 - val_loss: 1.1433 - val_accuracy: 0.5712
Epoch 29/50
450/450 [==============================] - 244s 542ms/step - loss: 1.3061 -
accuracy: 0.5030 - val_loss: 1.1470 - val_accuracy: 0.5733
Epoch 30/50
450/450 [==============================] - 250s 555ms/step - loss: 1.2999 -
accuracy: 0.5066 - val_loss: 1.1368 - val_accuracy: 0.5706
Epoch 31/50
450/450 [==============================] - 250s 556ms/step - loss: 1.2904 -
accuracy: 0.5113 - val_loss: 1.1176 - val_accuracy: 0.5800
Epoch 32/50
450/450 [==============================] - 252s 559ms/step - loss: 1.2887 -
accuracy: 0.5116 - val_loss: 1.1154 - val_accuracy: 0.5768
Epoch 33/50
450/450 [==============================] - 249s 553ms/step - loss: 1.2837 -
accuracy: 0.5140 - val_loss: 1.1319 - val_accuracy: 0.5726
Epoch 34/50
450/450 [==============================] - 240s 533ms/step - loss: 1.2703 -
accuracy: 0.5181 - val_loss: 1.1074 - val_accuracy: 0.5818
Epoch 35/50
450/450 [==============================] - 254s 563ms/step - loss: 1.2688 -
accuracy: 0.5157 - val_loss: 1.0896 - val_accuracy: 0.5876
Epoch 36/50
450/450 [==============================] - 250s 557ms/step - loss: 1.2635 -
accuracy: 0.5204 - val_loss: 1.1262 - val_accuracy: 0.5800
Epoch 37/50
450/450 [==============================] - 248s 552ms/step - loss: 1.2556 -
```

```
accuracy: 0.5247 - val_loss: 1.0845 - val_accuracy: 0.5913
Epoch 38/50
450/450 [==============================] - 248s 551ms/step - loss: 1.2538 -
accuracy: 0.5241 - val_loss: 1.0813 - val_accuracy: 0.5977
Epoch 39/50
450/450 [==============================] - 242s 539ms/step - loss: 1.2458 -
accuracy: 0.5286 - val_loss: 1.0957 - val_accuracy: 0.5832
Epoch 40/50
450/450 [==============================] - 250s 556ms/step - loss: 1.2407 -
accuracy: 0.5307 - val_loss: 1.0870 - val_accuracy: 0.5908
Epoch 41/50
450/450 [==============================] - 244s 543ms/step - loss: 1.2356 -
accuracy: 0.5311 - val_loss: 1.0716 - val_accuracy: 0.5974
Epoch 42/50
450/450 [==============================] - 251s 557ms/step - loss: 1.2331 -
accuracy: 0.5363 - val_loss: 1.0801 - val_accuracy: 0.5936
Epoch 43/50
450/450 [==============================] - 243s 541ms/step - loss: 1.2288 -
accuracy: 0.5335 - val_loss: 1.0612 - val_accuracy: 0.6017
Epoch 44/50
450/450 [==============================] - 246s 547ms/step - loss: 1.2212 -
accuracy: 0.5355 - val_loss: 1.0608 - val_accuracy: 0.6051
Epoch 45/50
450/450 [==============================] - 255s 567ms/step - loss: 1.2177 -
accuracy: 0.5383 - val_loss: 1.0686 - val_accuracy: 0.5969
Epoch 46/50
450/450 [==============================] - 252s 561ms/step - loss: 1.2139 -
accuracy: 0.5402 - val_loss: 1.0651 - val_accuracy: 0.5976
Epoch 47/50
450/450 [==============================] - 255s 567ms/step - loss: 1.2102 -
accuracy: 0.5378 - val_loss: 1.0479 - val_accuracy: 0.6107
Epoch 48/50
450/450 [==============================] - 249s 554ms/step - loss: 1.2048 -
accuracy: 0.5397 - val_loss: 1.0669 - val_accuracy: 0.5967
Epoch 49/50
450/450 [==============================] - 260s 577ms/step - loss: 1.1950 -
accuracy: 0.5480 - val_loss: 1.0563 - val_accuracy: 0.5974
Epoch 50/50
450/450 [==============================] - 249s 553ms/step - loss: 1.1958 -
accuracy: 0.5464 - val_loss: 1.0406 - val_accuracy: 0.6114
110/110 [==============================] - 16s 141ms/step - loss: 1.0405 -
accuracy: 0.6111
Validation Loss: 1.0405356884002686
Validation Accuracy: 0.6110795736312866
```

[34]:
```
import matplotlib.pyplot as plt
```

```
# Extract the history from the training process
history = emotion_model_info.history

# Training and validation loss
loss = history['loss']
val_loss = history['val_loss']

epochs = range(1, len(loss) + 1)

# Plot training and validation loss
plt.figure(figsize=(12, 6))
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```



Following the utilization of a Learning Rate Scheduler, which achieved a 63% accuracy, we proceeded to implement Data Augmentation in our emotion detection project in an effort to further enhance model performance. Despite our anticipation for improved robustness and recognition capabilities, the model's accuracy slightly decreased to 61% after applying Data Augmentation.

[ ]:
```
##########################
```

Model Training with Pre-trained model VGG16 to improve accuracy.

[36]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import VGG16

# Initialize image data generator with rescaling
train_data_gen = ImageDataGenerator(
    rescale=1./255,
    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'
)

validation_data_gen = ImageDataGenerator(rescale=1./255)

# Preprocess all training images
train_generator = train_data_gen.flow_from_directory(
        'images/train',
        target_size=(48, 48),
        batch_size=64,
        color_mode="rgb",
        class_mode='categorical')

# Preprocess all validation images
validation_generator = validation_data_gen.flow_from_directory(
        'images/test',
        target_size=(48, 48),
        batch_size=64,
        color_mode="rgb",
        class_mode='categorical')

# Load VGG16 model pre-trained on ImageNet, without the top layer, adapted for␣
 ↪grayscale input by duplicating channels
base_model = VGG16(weights='imagenet', include_top=False, input_shape=(48, 48,␣
 ↪3))

# Freeze the layers of the base model
for layer in base_model.layers:
    layer.trainable = False

# Add custom layers on top of VGG16
x = base_model.output
```

```
x = Flatten()(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(7, activation='softmax')(x)

# This is the model to train
model = Model(inputs=base_model.input, outputs=predictions)

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001),␣
 ↪loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    epochs=50,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // validation_generator.
 ↪batch_size
)

# Evaluate the model on the validation data
val_loss, val_accuracy = model.evaluate(validation_generator,␣
 ↪steps=validation_generator.samples // validation_generator.batch_size)

# Print out the results
print(f'Validation Loss: {val_loss}')
print(f'Validation Accuracy: {val_accuracy}')
```

```
Found 28822 images belonging to 7 classes.
Found 7066 images belonging to 7 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 4s 0us/step
Epoch 1/50
450/450 [==============================] - 570s 1s/step - loss: 1.7993 -
accuracy: 0.2605 - val_loss: 1.6528 - val_accuracy: 0.3425
Epoch 2/50
450/450 [==============================] - 564s 1s/step - loss: 1.7133 -
accuracy: 0.3082 - val_loss: 1.6257 - val_accuracy: 0.3665
Epoch 3/50
450/450 [==============================] - 559s 1s/step - loss: 1.6879 -
accuracy: 0.3228 - val_loss: 1.6181 - val_accuracy: 0.3692
Epoch 4/50
450/450 [==============================] - 557s 1s/step - loss: 1.6773 -
accuracy: 0.3277 - val_loss: 1.5949 - val_accuracy: 0.3839
```

```
Epoch 5/50
450/450 [==============================] - 555s 1s/step - loss: 1.6689 -
accuracy: 0.3343 - val_loss: 1.5876 - val_accuracy: 0.3855
Epoch 6/50
450/450 [==============================] - 567s 1s/step - loss: 1.6566 -
accuracy: 0.3400 - val_loss: 1.5887 - val_accuracy: 0.3788
Epoch 7/50
450/450 [==============================] - 562s 1s/step - loss: 1.6553 -
accuracy: 0.3422 - val_loss: 1.5822 - val_accuracy: 0.3871
Epoch 8/50
450/450 [==============================] - 563s 1s/step - loss: 1.6487 -
accuracy: 0.3466 - val_loss: 1.5794 - val_accuracy: 0.3852
Epoch 9/50
450/450 [==============================] - 562s 1s/step - loss: 1.6417 -
accuracy: 0.3481 - val_loss: 1.5738 - val_accuracy: 0.3893
Epoch 10/50
450/450 [==============================] - 552s 1s/step - loss: 1.6388 -
accuracy: 0.3526 - val_loss: 1.5651 - val_accuracy: 0.3950
Epoch 11/50
450/450 [==============================] - 557s 1s/step - loss: 1.6343 -
accuracy: 0.3533 - val_loss: 1.5640 - val_accuracy: 0.3972
Epoch 12/50
450/450 [==============================] - 555s 1s/step - loss: 1.6322 -
accuracy: 0.3524 - val_loss: 1.5613 - val_accuracy: 0.3949
Epoch 13/50
450/450 [==============================] - 555s 1s/step - loss: 1.6271 -
accuracy: 0.3557 - val_loss: 1.5560 - val_accuracy: 0.3982
Epoch 14/50
450/450 [==============================] - 563s 1s/step - loss: 1.6260 -
accuracy: 0.3552 - val_loss: 1.5547 - val_accuracy: 0.3997
Epoch 15/50
450/450 [==============================] - 554s 1s/step - loss: 1.6231 -
accuracy: 0.3612 - val_loss: 1.5538 - val_accuracy: 0.4020
Epoch 16/50
450/450 [==============================] - 563s 1s/step - loss: 1.6191 -
accuracy: 0.3593 - val_loss: 1.5572 - val_accuracy: 0.3946
Epoch 17/50
450/450 [==============================] - 563s 1s/step - loss: 1.6160 -
accuracy: 0.3648 - val_loss: 1.5522 - val_accuracy: 0.4003
Epoch 18/50
450/450 [==============================] - 578s 1s/step - loss: 1.6196 -
accuracy: 0.3576 - val_loss: 1.5473 - val_accuracy: 0.4034
Epoch 19/50
450/450 [==============================] - 561s 1s/step - loss: 1.6141 -
accuracy: 0.3669 - val_loss: 1.5531 - val_accuracy: 0.3987
Epoch 20/50
450/450 [==============================] - 556s 1s/step - loss: 1.6116 -
accuracy: 0.3646 - val_loss: 1.5480 - val_accuracy: 0.4001
```

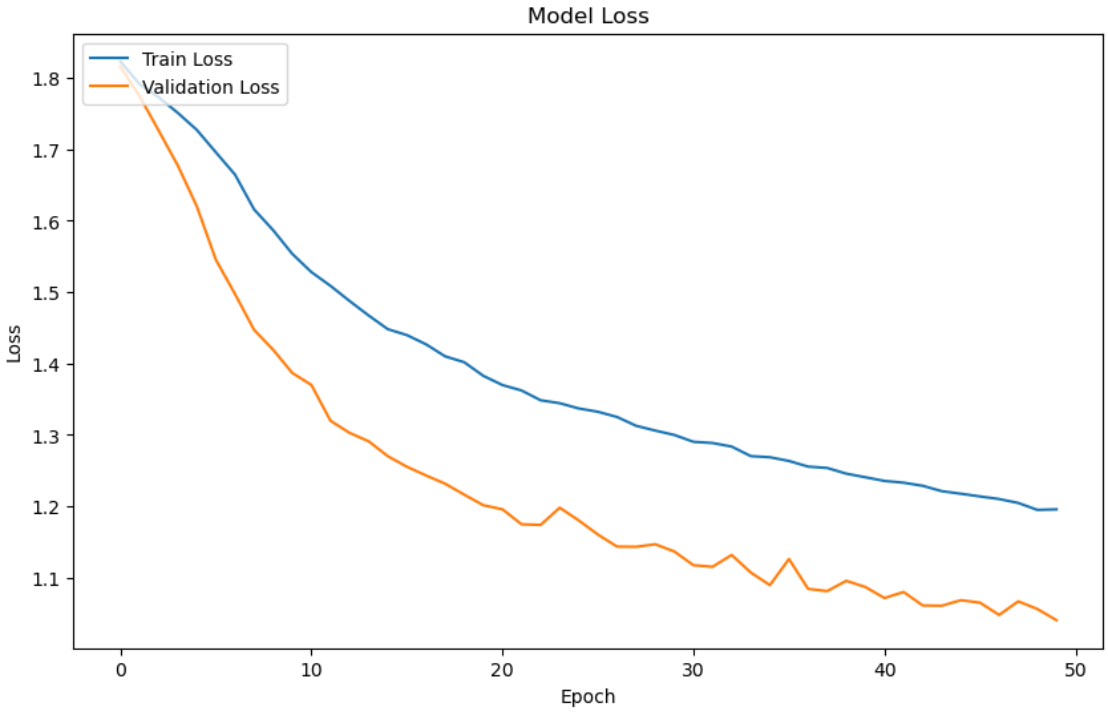
```
Epoch 21/50
450/450 [==============================] - 558s 1s/step - loss: 1.6124 -
accuracy: 0.3644 - val_loss: 1.5405 - val_accuracy: 0.4047
Epoch 22/50
450/450 [==============================] - 565s 1s/step - loss: 1.6061 -
accuracy: 0.3665 - val_loss: 1.5409 - val_accuracy: 0.4085
Epoch 23/50
450/450 [==============================] - 558s 1s/step - loss: 1.6072 -
accuracy: 0.3677 - val_loss: 1.5401 - val_accuracy: 0.4071
Epoch 24/50
450/450 [==============================] - 558s 1s/step - loss: 1.6044 -
accuracy: 0.3666 - val_loss: 1.5374 - val_accuracy: 0.4048
Epoch 25/50
450/450 [==============================] - 567s 1s/step - loss: 1.5962 -
accuracy: 0.3734 - val_loss: 1.5353 - val_accuracy: 0.4099
Epoch 26/50
450/450 [==============================] - 551s 1s/step - loss: 1.5969 -
accuracy: 0.3750 - val_loss: 1.5359 - val_accuracy: 0.4094
Epoch 27/50
450/450 [==============================] - 548s 1s/step - loss: 1.6013 -
accuracy: 0.3676 - val_loss: 1.5339 - val_accuracy: 0.4104
Epoch 28/50
450/450 [==============================] - 553s 1s/step - loss: 1.5963 -
accuracy: 0.3723 - val_loss: 1.5343 - val_accuracy: 0.4091
Epoch 29/50
450/450 [==============================] - 571s 1s/step - loss: 1.5943 -
accuracy: 0.3711 - val_loss: 1.5284 - val_accuracy: 0.4105
Epoch 30/50
450/450 [==============================] - 571s 1s/step - loss: 1.5945 -
accuracy: 0.3733 - val_loss: 1.5288 - val_accuracy: 0.4124
Epoch 31/50
450/450 [==============================] - 570s 1s/step - loss: 1.5914 -
accuracy: 0.3768 - val_loss: 1.5288 - val_accuracy: 0.4118
Epoch 32/50
450/450 [==============================] - 573s 1s/step - loss: 1.5917 -
accuracy: 0.3756 - val_loss: 1.5258 - val_accuracy: 0.4099
Epoch 33/50
450/450 [==============================] - 573s 1s/step - loss: 1.5899 -
accuracy: 0.3761 - val_loss: 1.5226 - val_accuracy: 0.4165
Epoch 34/50
450/450 [==============================] - 572s 1s/step - loss: 1.5874 -
accuracy: 0.3730 - val_loss: 1.5204 - val_accuracy: 0.4102
Epoch 35/50
450/450 [==============================] - 576s 1s/step - loss: 1.5840 -
accuracy: 0.3749 - val_loss: 1.5298 - val_accuracy: 0.4082
Epoch 36/50
450/450 [==============================] - 574s 1s/step - loss: 1.5828 -
accuracy: 0.3785 - val_loss: 1.5231 - val_accuracy: 0.4172
```

```
Epoch 37/50
450/450 [==============================] - 570s 1s/step - loss: 1.5834 -
accuracy: 0.3784 - val_loss: 1.5231 - val_accuracy: 0.4163
Epoch 38/50
450/450 [==============================] - 575s 1s/step - loss: 1.5827 -
accuracy: 0.3803 - val_loss: 1.5227 - val_accuracy: 0.4161
Epoch 39/50
450/450 [==============================] - 573s 1s/step - loss: 1.5810 -
accuracy: 0.3809 - val_loss: 1.5209 - val_accuracy: 0.4152
Epoch 40/50
450/450 [==============================] - 571s 1s/step - loss: 1.5841 -
accuracy: 0.3745 - val_loss: 1.5163 - val_accuracy: 0.4162
Epoch 41/50
450/450 [==============================] - 577s 1s/step - loss: 1.5825 -
accuracy: 0.3760 - val_loss: 1.5132 - val_accuracy: 0.4230
Epoch 42/50
450/450 [==============================] - 569s 1s/step - loss: 1.5780 -
accuracy: 0.3843 - val_loss: 1.5136 - val_accuracy: 0.4185
Epoch 43/50
450/450 [==============================] - 570s 1s/step - loss: 1.5801 -
accuracy: 0.3782 - val_loss: 1.5164 - val_accuracy: 0.4156
Epoch 44/50
450/450 [==============================] - 571s 1s/step - loss: 1.5754 -
accuracy: 0.3816 - val_loss: 1.5077 - val_accuracy: 0.4229
Epoch 45/50
450/450 [==============================] - 570s 1s/step - loss: 1.5706 -
accuracy: 0.3843 - val_loss: 1.5117 - val_accuracy: 0.4236
Epoch 46/50
450/450 [==============================] - 568s 1s/step - loss: 1.5706 -
accuracy: 0.3831 - val_loss: 1.5066 - val_accuracy: 0.4229
Epoch 47/50
450/450 [==============================] - 571s 1s/step - loss: 1.5722 -
accuracy: 0.3879 - val_loss: 1.5121 - val_accuracy: 0.4202
Epoch 48/50
450/450 [==============================] - 570s 1s/step - loss: 1.5680 -
accuracy: 0.3870 - val_loss: 1.5124 - val_accuracy: 0.4179
Epoch 49/50
450/450 [==============================] - 571s 1s/step - loss: 1.5699 -
accuracy: 0.3838 - val_loss: 1.5045 - val_accuracy: 0.4244
Epoch 50/50
450/450 [==============================] - 570s 1s/step - loss: 1.5645 -
accuracy: 0.3873 - val_loss: 1.5083 - val_accuracy: 0.4222
110/110 [==============================] - 107s 968ms/step - loss: 1.5083 -
accuracy: 0.4222
Validation Loss: 1.508292555809021
Validation Accuracy: 0.4221591055393219
```

In our final attempt to enhance the accuracy of our emotion detection model, we explored the

utilization of a pre-trained model, specifically VGG16, known for its effectiveness in image recognition tasks. VGG16, a convolutional neural network model pre-trained on the ImageNet dataset, is widely acclaimed for its high accuracy in detecting and classifying a myriad of image types. The model achieved a validation accuracy of only 42%, the lowest among all our experiments.

[38]:
```
import matplotlib.pyplot as plt

# Plot training & validation loss values
plt.figure(figsize=(10, 6))
plt.plot(emotion_model_info.history['loss'], label='Train Loss')
plt.plot(emotion_model_info.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper left')
plt.show()
```



END OF CODING PART.