In [1]:
```
import tensorflow as tf
import tensorflow.keras as keras
from tensorflow.keras import models, layers
```

In [2]:
```
from tensorflow.keras.datasets import mnist
import numpy
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical
```

In [3]:
```
(train_images, train_labels), (test_images, test_labels) = mnist.load_data(
```

In [4]:
```
train_images = train_images.astype('float32')/255
test_images = test_images.astype('float32')/255
```

In [5]:
```
train_images = train_images.reshape(train_images.shape[0],28,28,1)
```

In [6]:
```
test_images = test_images.reshape(test_images.shape[0],28,28,1)
```

In [7]:
```
train_images[0].shape
```

Out[7]:
```
(28, 28, 1)
```

In [8]:
```
data_augmentation = tf.keras.Sequential([
    layers.RandomRotation(0.1),
    layers.RandomFlip(),
    layers.RandomZoom(0.1),
    layers.RandomTranslation(0.1, 0.1)
])
```

In [9]:
```
def create_model():
    inputs = keras.Input(shape=(28,28,1))
    x = data_augmentation(inputs)

    x = keras.layers.Conv2D(32, 3, activation="relu", padding="same")(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Conv2D(32, 3, activation="relu", padding="same")(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.MaxPool2D(2)(x)
    x = keras.layers.SpatialDropout2D(0.2)(x)


    x = keras.layers.Conv2D(64, 3, activation="relu", padding="same")(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Conv2D(64, 3, activation="relu", padding="same")(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.MaxPool2D(2)(x)
    x = keras.layers.SpatialDropout2D(0.2)(x)

    residual = x
    x = keras.layers.Conv2D(64, 3, activation="relu", padding="same")(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.Conv2D(64, 3, activation="relu", padding="same")(x)
    x = keras.layers.BatchNormalization()(x)
    x = keras.layers.add([x,residual])


    x = keras.layers.GlobalAveragePooling2D()(x)
    x = keras.layers.Dense(256, activation="relu")(x)
    outputs = keras.layers.Dense(10, activation="softmax")(x)
```

```
    model =keras.Model(inputs, outputs)
    return model
```

In [10]:
```
from tensorflow.keras.callbacks import EarlyStopping
```

In [11]:
```
callbacks = [EarlyStopping(patience=10, restore_best_weights=True)]
```

In [12]:
```
model = create_model()
```

In [13]:
```
from tensorflow.keras.optimizers import Adam
```

In [14]:
```
model.compile(optimizer=Adam(learning_rate=0.001), loss = "sparse_categorica
```

In [15]:
```
history = model.fit(train_images, train_labels, callbacks=callbacks, epochs=
```

```
Epoch 1/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 128s 17ms/step - accuracy: 0.6502 - loss: 0.
9945 - val_accuracy: 0.9339 - val_loss: 0.1922
Epoch 2/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 135s 18ms/step - accuracy: 0.8729 - loss: 0.
3833 - val_accuracy: 0.9429 - val_loss: 0.1740
Epoch 3/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 141s 19ms/step - accuracy: 0.8922 - loss: 0.
3250 - val_accuracy: 0.9495 - val_loss: 0.1475
Epoch 4/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 134s 18ms/step - accuracy: 0.9057 - loss: 0.
2872 - val_accuracy: 0.9341 - val_loss: 0.1870
Epoch 5/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 132s 18ms/step - accuracy: 0.9103 - loss: 0.
2681 - val_accuracy: 0.9473 - val_loss: 0.1589
Epoch 6/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 131s 17ms/step - accuracy: 0.9147 - loss: 0.
2535 - val_accuracy: 0.9529 - val_loss: 0.1483
Epoch 7/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 144s 19ms/step - accuracy: 0.9190 - loss: 0.
2459 - val_accuracy: 0.9584 - val_loss: 0.1307
Epoch 8/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 161s 22ms/step - accuracy: 0.9248 - loss: 0.
2265 - val_accuracy: 0.9542 - val_loss: 0.1417
Epoch 9/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 169s 22ms/step - accuracy: 0.9261 - loss: 0.
2214 - val_accuracy: 0.9586 - val_loss: 0.1237
Epoch 10/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 176s 23ms/step - accuracy: 0.9272 - loss: 0.
2182 - val_accuracy: 0.9556 - val_loss: 0.1356
Epoch 11/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 180s 24ms/step - accuracy: 0.9305 - loss: 0.
2098 - val_accuracy: 0.9567 - val_loss: 0.1292
Epoch 12/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 179s 24ms/step - accuracy: 0.9324 - loss: 0.
2009 - val_accuracy: 0.9639 - val_loss: 0.1128
Epoch 13/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 181s 24ms/step - accuracy: 0.9313 - loss: 0.
2060 - val_accuracy: 0.9503 - val_loss: 0.1435
Epoch 14/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 183s 24ms/step - accuracy: 0.9340 - loss: 0.
2003 - val_accuracy: 0.9502 - val_loss: 0.1538
Epoch 15/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 165s 22ms/step - accuracy: 0.9361 - loss: 0.
1949 - val_accuracy: 0.9571 - val_loss: 0.1319
Epoch 16/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 183s 24ms/step - accuracy: 0.9366 - loss: 0.
1911 - val_accuracy: 0.9573 - val_loss: 0.1286
Epoch 17/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 183s 24ms/step - accuracy: 0.9355 - loss: 0.
1907 - val_accuracy: 0.9536 - val_loss: 0.1351
Epoch 18/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 200s 27ms/step - accuracy: 0.9391 - loss: 0.
1856 - val_accuracy: 0.9585 - val_loss: 0.1241
Epoch 19/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 202s 27ms/step - accuracy: 0.9379 - loss: 0.
1898 - val_accuracy: 0.9604 - val_loss: 0.1218
Epoch 20/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 212s 28ms/step - accuracy: 0.9399 - loss: 0.
1820 - val_accuracy: 0.9534 - val_loss: 0.1380
Epoch 21/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 208s 28ms/step - accuracy: 0.9417 - loss: 0.
1786 - val_accuracy: 0.9541 - val_loss: 0.1342
Epoch 22/100
```

```
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 202s 27ms/step - accuracy: 0.9393 - loss: 0.
1831 - val_accuracy: 0.9625 - val_loss: 0.1102
Epoch 23/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 200s 27ms/step - accuracy: 0.9402 - loss: 0.
1787 - val_accuracy: 0.9587 - val_loss: 0.1201
Epoch 24/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 211s 28ms/step - accuracy: 0.9427 - loss: 0.
1726 - val_accuracy: 0.9603 - val_loss: 0.1212
Epoch 25/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 189s 25ms/step - accuracy: 0.9413 - loss: 0.
1756 - val_accuracy: 0.9689 - val_loss: 0.0972
Epoch 26/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 187s 25ms/step - accuracy: 0.9418 - loss: 0.
1792 - val_accuracy: 0.9588 - val_loss: 0.1273
Epoch 27/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 189s 25ms/step - accuracy: 0.9424 - loss: 0.
1721 - val_accuracy: 0.9598 - val_loss: 0.1226
Epoch 28/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 188s 25ms/step - accuracy: 0.9418 - loss: 0.
1741 - val_accuracy: 0.9574 - val_loss: 0.1332
Epoch 29/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 189s 25ms/step - accuracy: 0.9413 - loss: 0.
1766 - val_accuracy: 0.9601 - val_loss: 0.1258
Epoch 30/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 188s 25ms/step - accuracy: 0.9457 - loss: 0.
1632 - val_accuracy: 0.9648 - val_loss: 0.1075
Epoch 31/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 187s 25ms/step - accuracy: 0.9447 - loss: 0.
1698 - val_accuracy: 0.9638 - val_loss: 0.1095
Epoch 32/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 177s 24ms/step - accuracy: 0.9458 - loss: 0.
1650 - val_accuracy: 0.9637 - val_loss: 0.1097
Epoch 33/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 190s 25ms/step - accuracy: 0.9431 - loss: 0.
1689 - val_accuracy: 0.9681 - val_loss: 0.1019
Epoch 34/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 187s 25ms/step - accuracy: 0.9440 - loss: 0.
1682 - val_accuracy: 0.9668 - val_loss: 0.1081
Epoch 35/100
7500/7500 ━━━━━━━━━━━━━━━━━━━━ 192s 26ms/step - accuracy: 0.9453 - loss: 0.
1653 - val_accuracy: 0.9640 - val_loss: 0.1097
```

In [1]:

```python
import joblib
import dill

# Assuming 'model' is your trained model object
joblib.dump(model, 'model.pkl', compress=3, protocol=dill.HIGHEST_PROTOCOL)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[1], line 5
      2 import dill
      4 # Assuming 'model' is your trained model object
----> 5 joblib.dump(model, 'model.pkl', compress=3, protocol=dill.HIGHEST_P
ROTOCOL)

NameError: name 'model' is not defined
```

In [17]:

```python
import joblib
joblib.dump(model, 'model.pkl')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[17], line 2
      1 import joblib
----> 2 joblib.dump(model, 'model.pkl')

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:553, i
n dump(value, filename, compress, protocol, cache_size)
    551 elif is_filename:
    552     with open(filename, 'wb') as f:
--> 553         NumpyPickler(f, protocol=protocol).dump(value)
    554 else:
    555     NumpyPickler(filename, protocol=protocol).dump(value)

File ~/anaconda3/lib/python3.11/pickle.py:487, in _Pickler.dump(self, obj)
    485 if self.proto >= 4:
    486     self.framer.start_framing()
--> 487 self.save(obj)
    488 self.write(STOP)
    489 self.framer.end_framing()

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:603, in _Pickler.save(self, obj,
save_persistent_id)
    599     raise PicklingError("Tuple returned by %s must have "
    600                         "two to six elements" % reduce)
    602 # Save the reduce() output and finally memoize the object
--> 603 self.save_reduce(obj=obj, *rv)

File ~/anaconda3/lib/python3.11/pickle.py:717, in _Pickler.save_reduce(sel
f, func, args, state, listitems, dictitems, state_setter, obj)
    715 if state is not None:
    716     if state_setter is None:
--> 717         save(state)
    718         write(BUILD)
    719     else:
    720         # If a state_setter is specified, call it instead of load_b
uild
    721         # to update obj's with its previous state.
    722         # First, push state_setter and its tuple of expected argume
nts
    723         # (obj, state) onto the stack.

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:560, in _Pickler.save(self, obj,
save_persistent_id)
    558 f = self.dispatch.get(t)
    559 if f is not None:
--> 560     f(self, obj)  # Call unbound method with explicit self
    561     return
    563 # Check private dispatch table if any, or else
    564 # copyreg.dispatch_table

File ~/anaconda3/lib/python3.11/site-packages/dill/_dill.py:1186, in save_m
```

```
odule_dict(pickler, obj)
   1183     if is_dill(pickler, child=False) and pickler._session:
   1184         # we only care about session the first pass thru
   1185         pickler._first_pass = False
-> 1186     StockPickler.save_dict(pickler, obj)
   1187     logger.trace(pickler, "# D2")
   1188 return

File ~/anaconda3/lib/python3.11/pickle.py:972, in _Pickler.save_dict(self,
obj)
    969     self.write(MARK + DICT)
    971 self.memoize(obj)
--> 972 self._batch_setitems(obj.items())

File ~/anaconda3/lib/python3.11/pickle.py:998, in _Pickler._batch_setitems
(self, items)
    996     for k, v in tmp:
    997         save(k)
--> 998         save(v)
    999     write(SETITEMS)
   1000 elif n:

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:603, in _Pickler.save(self, obj,
save_persistent_id)
    599     raise PicklingError("Tuple returned by %s must have "
    600                         "two to six elements" % reduce)
    602 # Save the reduce() output and finally memoize the object
--> 603 self.save_reduce(obj=obj, *rv)

File ~/anaconda3/lib/python3.11/pickle.py:692, in _Pickler.save_reduce(sel
f, func, args, state, listitems, dictitems, state_setter, obj)
    690 else:
    691     save(func)
--> 692     save(args)
    693     write(REDUCE)
    695 if obj is not None:
    696     # If the object is already in the memo, this means it is
    697     # recursive. In this case, throw away everything we put on the
    698     # stack, and fetch the object back from the memo.

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:560, in _Pickler.save(self, obj,
save_persistent_id)
    558 f = self.dispatch.get(t)
    559 if f is not None:
--> 560     f(self, obj)  # Call unbound method with explicit self
    561     return
    563 # Check private dispatch table if any, or else
    564 # copyreg.dispatch_table

File ~/anaconda3/lib/python3.11/pickle.py:887, in _Pickler.save_tuple(self,
obj)
    885 if n <= 3 and self.proto >= 2:
```

```
    886     for element in obj:
--> 887         save(element)
    888     # Subtle.  Same as in the big comment below.
    889     if id(obj) in memo:

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:560, in _Pickler.save(self, obj,
save_persistent_id)
    558 f = self.dispatch.get(t)
    559 if f is not None:
--> 560     f(self, obj)  # Call unbound method with explicit self
    561     return
    563 # Check private dispatch table if any, or else
    564 # copyreg.dispatch_table

File ~/anaconda3/lib/python3.11/pickle.py:932, in _Pickler.save_list(self,
obj)
    929     self.write(MARK + LIST)
    931 self.memoize(obj)
--> 932 self._batch_appends(obj)

File ~/anaconda3/lib/python3.11/pickle.py:956, in _Pickler._batch_appends(s
elf, items)
    954     write(MARK)
    955     for x in tmp:
--> 956         save(x)
    957     write(APPENDS)
    958 elif n:

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:603, in _Pickler.save(self, obj,
save_persistent_id)
    599     raise PicklingError("Tuple returned by %s must have "
    600                         "two to six elements" % reduce)
    602 # Save the reduce() output and finally memoize the object
--> 603 self.save_reduce(obj=obj, *rv)

File ~/anaconda3/lib/python3.11/pickle.py:717, in _Pickler.save_reduce(sel
f, func, args, state, listitems, dictitems, state_setter, obj)
    715 if state is not None:
    716     if state_setter is None:
--> 717         save(state)
    718         write(BUILD)
    719     else:
    720         # If a state_setter is specified, call it instead of load_b
uild
    721         # to update obj's with its previous state.
    722         # First, push state_setter and its tuple of expected argume
nts
    723         # (obj, state) onto the stack.

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
```

```
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:560, in _Pickler.save(self, obj, save_persistent_id)
    558 f = self.dispatch.get(t)
    559 if f is not None:
--> 560     f(self, obj)  # Call unbound method with explicit self
    561     return
    563 # Check private dispatch table if any, or else
    564 # copyreg.dispatch_table

File ~/anaconda3/lib/python3.11/site-packages/dill/_dill.py:1186, in save_module_dict(pickler, obj)
   1183     if is_dill(pickler, child=False) and pickler._session:
   1184         # we only care about session the first pass thru
   1185         pickler._first_pass = False
-> 1186     StockPickler.save_dict(pickler, obj)
   1187     logger.trace(pickler, "# D2")
   1188 return

File ~/anaconda3/lib/python3.11/pickle.py:972, in _Pickler.save_dict(self, obj)
    969     self.write(MARK + DICT)
    971 self.memoize(obj)
--> 972 self._batch_setitems(obj.items())

File ~/anaconda3/lib/python3.11/pickle.py:998, in _Pickler._batch_setitems(self, items)
    996     for k, v in tmp:
    997         save(k)
--> 998         save(v)
    999     write(SETITEMS)
   1000 elif n:

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, in NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:603, in _Pickler.save(self, obj, save_persistent_id)
    599     raise PicklingError("Tuple returned by %s must have "
    600                         "two to six elements" % reduce)
    602 # Save the reduce() output and finally memoize the object
--> 603 self.save_reduce(obj=obj, *rv)

File ~/anaconda3/lib/python3.11/pickle.py:710, in _Pickler.save_reduce(self, func, args, state, listitems, dictitems, state_setter, obj)
    704 # More new special cases (that work with older protocols as
    705 # well): when __reduce__ returns a tuple with 4 or 5 items,
    706 # the 4th and 5th item should be iterators that provide list
    707 # items and dict items (as (key, value) tuples), or None.
    709 if listitems is not None:
--> 710     self._batch_appends(listitems)
    712 if dictitems is not None:
    713     self._batch_setitems(dictitems)

File ~/anaconda3/lib/python3.11/pickle.py:959, in _Pickler._batch_appends(self, items)
    957     write(APPENDS)
    958 elif n:
--> 959     save(tmp[0])
```

```
    960     write(APPEND)
    961 # else tmp is empty, and we're done

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:603, in _Pickler.save(self, obj,
save_persistent_id)
    599     raise PicklingError("Tuple returned by %s must have "
    600                         "two to six elements" % reduce)
    602 # Save the reduce() output and finally memoize the object
--> 603 self.save_reduce(obj=obj, *rv)

File ~/anaconda3/lib/python3.11/pickle.py:717, in _Pickler.save_reduce(sel
f, func, args, state, listitems, dictitems, state_setter, obj)
    715 if state is not None:
    716     if state_setter is None:
--> 717         save(state)
    718         write(BUILD)
    719     else:
    720         # If a state_setter is specified, call it instead of load_b
uild
    721         # to update obj's with its previous state.
    722         # First, push state_setter and its tuple of expected argume
nts
    723         # (obj, state) onto the stack.

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:560, in _Pickler.save(self, obj,
save_persistent_id)
    558 f = self.dispatch.get(t)
    559 if f is not None:
--> 560     f(self, obj)  # Call unbound method with explicit self
    561     return
    563 # Check private dispatch table if any, or else
    564 # copyreg.dispatch_table

File ~/anaconda3/lib/python3.11/site-packages/dill/_dill.py:1186, in save_m
odule_dict(pickler, obj)
   1183     if is_dill(pickler, child=False) and pickler._session:
   1184         # we only care about session the first pass thru
   1185         pickler._first_pass = False
-> 1186     StockPickler.save_dict(pickler, obj)
   1187     logger.trace(pickler, "# D2")
   1188 return

File ~/anaconda3/lib/python3.11/pickle.py:972, in _Pickler.save_dict(self,
obj)
    969     self.write(MARK + DICT)
    971 self.memoize(obj)
--> 972 self._batch_setitems(obj.items())

File ~/anaconda3/lib/python3.11/pickle.py:998, in _Pickler._batch_setitems
(self, items)
    996     for k, v in tmp:
    997         save(k)
```

```
--> 998         save(v)
    999     write(SETITEMS)
   1000 elif n:

    [... skipping similar frames: NumpyPickler.save at line 355 (1 times)]

File ~/anaconda3/lib/python3.11/pickle.py:603, in _Pickler.save(self, obj,
save_persistent_id)
    599     raise PicklingError("Tuple returned by %s must have "
    600                         "two to six elements" % reduce)
    602 # Save the reduce() output and finally memoize the object
--> 603 self.save_reduce(obj=obj, *rv)

File ~/anaconda3/lib/python3.11/pickle.py:717, in _Pickler.save_reduce(sel
f, func, args, state, listitems, dictitems, state_setter, obj)
    715 if state is not None:
    716     if state_setter is None:
--> 717         save(state)
    718         write(BUILD)
    719     else:
    720         # If a state_setter is specified, call it instead of load_b
uild
    721         # to update obj's with its previous state.
    722         # First, push state_setter and its tuple of expected argume
nts
    723         # (obj, state) onto the stack.

    [... skipping similar frames: NumpyPickler.save at line 355 (1 times)]

File ~/anaconda3/lib/python3.11/pickle.py:560, in _Pickler.save(self, obj,
save_persistent_id)
    558 f = self.dispatch.get(t)
    559 if f is not None:
--> 560     f(self, obj)  # Call unbound method with explicit self
    561     return
    563 # Check private dispatch table if any, or else
    564 # copyreg.dispatch_table

File ~/anaconda3/lib/python3.11/site-packages/dill/_dill.py:1186, in save_m
odule_dict(pickler, obj)
   1183     if is_dill(pickler, child=False) and pickler._session:
   1184         # we only care about session the first pass thru
   1185         pickler._first_pass = False
-> 1186     StockPickler.save_dict(pickler, obj)
   1187     logger.trace(pickler, "# D2")
   1188 return

File ~/anaconda3/lib/python3.11/pickle.py:972, in _Pickler.save_dict(self,
obj)
    969     self.write(MARK + DICT)
    971 self.memoize(obj)
--> 972 self._batch_setitems(obj.items())

File ~/anaconda3/lib/python3.11/pickle.py:998, in _Pickler._batch_setitems
(self, items)
    996     for k, v in tmp:
    997         save(k)
--> 998         save(v)
    999     write(SETITEMS)
   1000 elif n:

    [... skipping similar frames: NumpyPickler.save at line 355 (1 times)]

File ~/anaconda3/lib/python3.11/pickle.py:603, in _Pickler.save(self, obj,
```

```
save_persistent_id)
    599     raise PicklingError("Tuple returned by %s must have "
    600                         "two to six elements" % reduce)
    602 # Save the reduce() output and finally memoize the object
--> 603 self.save_reduce(obj=obj, *rv)

File ~/anaconda3/lib/python3.11/pickle.py:717, in _Pickler.save_reduce(sel
f, func, args, state, listitems, dictitems, state_setter, obj)
    715 if state is not None:
    716     if state_setter is None:
--> 717         save(state)
    718         write(BUILD)
    719     else:
    720         # If a state_setter is specified, call it instead of load_b
uild
    721         # to update obj's with its previous state.
    722         # First, push state_setter and its tuple of expected argume
nts
    723         # (obj, state) onto the stack.

    [... skipping similar frames: NumpyPickler.save at line 355 (2 times),
_Pickler.save at line 560 (2 times), _Pickler._batch_setitems at line 998
(1 times), _Pickler.save_dict at line 972 (1 times), save_module_dict at li
ne 1186 (1 times)]

File ~/anaconda3/lib/python3.11/site-packages/dill/_dill.py:1186, in save_m
odule_dict(pickler, obj)
   1183     if is_dill(pickler, child=False) and pickler._session:
   1184         # we only care about session the first pass thru
   1185         pickler._first_pass = False
-> 1186     StockPickler.save_dict(pickler, obj)
   1187     logger.trace(pickler, "# D2")
   1188 return

File ~/anaconda3/lib/python3.11/pickle.py:972, in _Pickler.save_dict(self,
obj)
    969     self.write(MARK + DICT)
    971 self.memoize(obj)
--> 972 self._batch_setitems(obj.items())

File ~/anaconda3/lib/python3.11/pickle.py:998, in _Pickler._batch_setitems
(self, items)
    996     for k, v in tmp:
    997         save(k)
--> 998         save(v)
    999     write(SETITEMS)
   1000 elif n:

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:560, in _Pickler.save(self, obj,
save_persistent_id)
    558 f = self.dispatch.get(t)
    559 if f is not None:
--> 560     f(self, obj)  # Call unbound method with explicit self
    561     return
    563 # Check private dispatch table if any, or else
    564 # copyreg.dispatch_table

File ~/anaconda3/lib/python3.11/pickle.py:887, in _Pickler.save_tuple(self,
```

```
obj)
    885 if n <= 3 and self.proto >= 2:
    886     for element in obj:
--> 887         save(element)
    888     # Subtle.  Same as in the big comment below.
    889     if id(obj) in memo:

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:560, in _Pickler.save(self, obj,
save_persistent_id)
    558 f = self.dispatch.get(t)
    559 if f is not None:
--> 560     f(self, obj)  # Call unbound method with explicit self
    561     return
    563 # Check private dispatch table if any, or else
    564 # copyreg.dispatch_table

File ~/anaconda3/lib/python3.11/pickle.py:932, in _Pickler.save_list(self,
obj)
    929     self.write(MARK + LIST)
    931 self.memoize(obj)
--> 932 self._batch_appends(obj)

File ~/anaconda3/lib/python3.11/pickle.py:956, in _Pickler._batch_appends(s
elf, items)
    954     write(MARK)
    955     for x in tmp:
--> 956         save(x)
    957     write(APPENDS)
    958 elif n:

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:603, in _Pickler.save(self, obj,
save_persistent_id)
    599     raise PicklingError("Tuple returned by %s must have "
    600                         "two to six elements" % reduce)
    602 # Save the reduce() output and finally memoize the object
--> 603 self.save_reduce(obj=obj, *rv)

File ~/anaconda3/lib/python3.11/pickle.py:717, in _Pickler.save_reduce(sel
f, func, args, state, listitems, dictitems, state_setter, obj)
    715 if state is not None:
    716     if state_setter is None:
--> 717         save(state)
    718         write(BUILD)
    719     else:
    720         # If a state_setter is specified, call it instead of load_b
uild
    721         # to update obj's with its previous state.
    722         # First, push state_setter and its tuple of expected argume
nts
    723         # (obj, state) onto the stack.

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
```

```
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:560, in _Pickler.save(self, obj,
save_persistent_id)
    558 f = self.dispatch.get(t)
    559 if f is not None:
--> 560     f(self, obj)  # Call unbound method with explicit self
    561     return
    563 # Check private dispatch table if any, or else
    564 # copyreg.dispatch_table

File ~/anaconda3/lib/python3.11/site-packages/dill/_dill.py:1186, in save_m
odule_dict(pickler, obj)
   1183     if is_dill(pickler, child=False) and pickler._session:
   1184         # we only care about session the first pass thru
   1185         pickler._first_pass = False
-> 1186     StockPickler.save_dict(pickler, obj)
   1187     logger.trace(pickler, "# D2")
   1188 return

File ~/anaconda3/lib/python3.11/pickle.py:972, in _Pickler.save_dict(self,
obj)
    969     self.write(MARK + DICT)
    971 self.memoize(obj)
--> 972 self._batch_setitems(obj.items())

File ~/anaconda3/lib/python3.11/pickle.py:998, in _Pickler._batch_setitems
(self, items)
    996     for k, v in tmp:
    997         save(k)
--> 998         save(v)
    999     write(SETITEMS)
   1000 elif n:

File ~/anaconda3/lib/python3.11/site-packages/joblib/numpy_pickle.py:355, i
n NumpyPickler.save(self, obj)
    352     wrapper.write_array(obj, self)
    353     return
--> 355 return Pickler.save(self, obj)

File ~/anaconda3/lib/python3.11/pickle.py:560, in _Pickler.save(self, obj,
save_persistent_id)
    558 f = self.dispatch.get(t)
    559 if f is not None:
--> 560     f(self, obj)  # Call unbound method with explicit self
    561     return
    563 # Check private dispatch table if any, or else
    564 # copyreg.dispatch_table

File ~/anaconda3/lib/python3.11/site-packages/dill/_dill.py:1799, in save_f
unction(pickler, obj)
   1794 if globs_copy is not None and globs is not globs_copy:
   1795     # In the case that the globals are copied, we need to ensure th
at
   1796     # the globals dictionary is updated when all objects in the
   1797     # dictionary are already created.
   1798     glob_ids = {id(g) for g in globs_copy.values()}
-> 1799     for stack_element in _postproc:
   1800         if stack_element in glob_ids:
   1801             _postproc[stack_element].append((_setitems, (globs, glo
bs_copy)))
```

```
TypeError: 'NoneType' object is not iterable
```

In [ ]:
```
from sklearn.metrics import classification_report
```

In [ ]:
```
import pprint
```

In [ ]:
```
y_pred = model.predict(test_images).argmax(axis=1)
```

In [ ]:
```
pprint.pprint(classification_report(test_labels, y_pred))
```

In [ ]:
```
import seaborn as sns
```

In [ ]:
```
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.tight_layout()
plt.show()

# Function to plot confusion matrix
def plot_confusion_matrix(y_true, y_pred, classes):
    cm = tf.math.confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes,
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted Label')
    plt.ylabel('True Label')
    plt.show()

# Make predictions and plot confusion matrix
class_names = ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight
plot_confusion_matrix(test_labels, y_pred, class_names)
```

In [ ]:
```
import numpy as np
from PIL import Image
import tensorflow as tf

def preprocess_image(image_path):
    # Open the image
    img = Image.open(image_path)

    # Convert to grayscale if it's not already
    if img.mode != 'L':
        img = img.convert('L')

    # Resize to 28x28 pixels
    img = img.resize((28, 28))
```

```
    # Convert to numpy array and normalize
    img_array = np.array(img).astype('float32') / 255.0

    # Reshape to (1, 28, 28, 1) for model input
    img_array = img_array.reshape(1, 28, 28, 1)

    return img_array
```

In [ ]:
```
def predict_class(model, image_array):
    # Make prediction
    prediction = model.predict(image_array)
    print(prediction)
    # Get the class with highest probability
    class_index = np.argmax(prediction)

    print(class_index)

    # Define class names
    class_names = ['one', 'two', 'three', 'four', 'five',
                   'six', 'seven', 'eight', 'nine', 'ten']

    print(len(class_names))

    # Get the predicted class name
    predicted_class = class_names[class_index]

    # Get the confidence (probability)
    confidence = prediction[0][class_index]

    return predicted_class, confidence
```

In [ ]:
```
# Assume 'model' is your trained model
# Replace 'path_to_your_image.jpg' with the actual path to your image
image_path = 'seven.jpg'

# Preprocess the image
processed_image = preprocess_image(image_path)

# Predict the class
predicted_class, confidence = predict_class(model, processed_image)

print(f"Predicted class: {predicted_class}")
print(f"Confidence: {confidence:.2f}")
```

In [ ]:

In [ ]: