In [1]:
```
# data augmentation: mirror (use read_dataset2, dataset2)
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.python.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.python.keras.models import Model
from tensorflow.python.keras import layers, Sequential,losses, metrics

image_height = 48
image_width = 48
emotions_count = 8
emotion_labels = ['neutral', 'happiness', 'surprise', 'sadness',
                  'anger', 'disgust', 'fear', 'contempt']

samples = 67251 # 2~67252
training_samples = 28317*2  # 2~56635 (Training)
validation_samples = 3541*2 # 56636~63717 (PublicTest)
test_samples = 3535         # 63718~67252 (PrivateTest)

image_path = "./dataset2/images.npy"
emotion_multi_path = "./dataset2/emotions_multi.npy"
emotion_single_path = "./dataset2/emotions_single.npy"
```

In [2]:
```
images = np.load(image_path)
emotions_multi = np.load(emotion_multi_path)
emotions_single = np.load(emotion_single_path)

print(images.shape)
print(emotions_multi.shape)
print(emotions_single.shape)
```

```
(67251, 48, 48, 1)
(67251, 8)
(67251, 8)
```

In [3]:
```
tf.config.run_functions_eagerly(True)
def model_acc(y_true, y_pred):
    size = y_true.shape[0]
    acc = 0
    for i in range(size):
        true = y_true[i]
        pred = y_pred[i]
        index_max = tf.argmax(pred).numpy()
        if true[index_max].numpy()==tf.reduce_max(true).numpy():
            acc += 1
    return acc/size
```

In [4]:

```
#emotions = emotions_single
emotions = emotions_multi

images = tf.convert_to_tensor(images)
images = tf.image.grayscale_to_rgb(images)
emotions = tf.convert_to_tensor(emotions)
# images = tf.image.resize(images, [224,224])
images = layers.Rescaling(1./127.5, offset= -1)(images)

training_size = training_samples + validation_samples
test_size = test_samples

training_images = images[:training_size]
test_images = images[training_size:]
training_emotions = emotions[:training_size]
test_emotions = emotions[training_size:]

print("training_images shape:", training_images.shape)
print("training_emotions shape:", training_emotions.shape)
print("test_images shape:", test_images.shape)
print("test_emotions shape:", test_emotions.shape)
```

```
training_images shape: (63716, 48, 48, 3)
training_emotions shape: (63716, 8)
test_images shape: (3535, 48, 48, 3)
test_emotions shape: (3535, 8)
```

In [5]:

```
from tensorflow.python.keras.applications import vgg16, resnet_v2
from tensorflow.python.keras import optimizers
from tensorflow.python.keras.optimizer_v2 import adam
```

In [6]:

```
base_model = vgg16.VGG16(include_top=False,
                         weights="imagenet",
                         input_shape=(48,48,3))
```

```
base_model.trainable=True
model = Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(4096, activation='relu'),
    layers.Dense(4096, activation='relu'),
    layers.Dense(emotions_count, activation='softmax'),
])

model.compile(optimizer=adam.Adam(learning_rate=1e-4),
              loss=losses.CategoricalCrossentropy(),
              metrics = "accuracy")

model.fit(x=training_images,
          y=training_emotions,
          batch_size=32,
          epochs=40,
          validation_data=(test_images, test_emotions))
```

```
C:\Users\Darkl\anaconda3\lib\site-packages\tensorflow\python\data\ops\dataset_ops.py:3703: UserWarning: Even though the `tf.c
onfig.experimental_run_functions_eagerly` option is set, this option does not apply to tf.data functions. To force eager exec
ution of tf.data functions, please use `tf.data.experimental.enable.debug_mode()`.
  warnings.warn(
Epoch 1/40
1992/1992 [==============================] - 82s 40ms/step - loss: 1.1042 - accuracy: 0.7042 - val_loss: 0.9480 - val_accurac
y: 0.7649
Epoch 2/40
1992/1992 [==============================] - 79s 40ms/step - loss: 0.8938 - accuracy: 0.8057 - val_loss: 0.8894 - val_accurac
y: 0.7844
Epoch 3/40
1992/1992 [==============================] - 79s 40ms/step - loss: 0.8260 - accuracy: 0.8396 - val_loss: 0.8701 - val_accurac
y: 0.8136
Epoch 4/40
1992/1992 [==============================] - 79s 40ms/step - loss: 0.7801 - accuracy: 0.8645 - val_loss: 0.8490 - val_accurac
y: 0.8184
Epoch 5/40
1992/1992 [==============================] - 79s 40ms/step - loss: 0.7460 - accuracy: 0.8835 - val_loss: 0.8436 - val_accurac
y: 0.8195
Epoch 6/40
1992/1992 [==============================] - 81s 41ms/step - loss: 0.7180 - accuracy: 0.8973 - val_loss: 0.8357 - val_accurac
y: 0.8274
Epoch 7/40
1992/1992 [==============================] - 82s 41ms/step - loss: 0.6940 - accuracy: 0.9064 - val_loss: 0.8414 - val_accurac
y: 0.8283
```

```
Epoch 8/40
1992/1992 [==============================] - 85s 43ms/step - loss: 0.6713 - accuracy: 0.9147 - val_loss: 0.8567 - val_accurac
y: 0.8252
Epoch 9/40
1992/1992 [==============================] - 81s 41ms/step - loss: 0.6505 - accuracy: 0.9196 - val_loss: 0.8579 - val_accurac
y: 0.8368
Epoch 10/40
1992/1992 [==============================] - 85s 43ms/step - loss: 0.6319 - accuracy: 0.9250 - val_loss: 0.8790 - val_accurac
y: 0.8348
Epoch 11/40
1992/1992 [==============================] - 83s 42ms/step - loss: 0.6152 - accuracy: 0.9288 - val_loss: 0.9126 - val_accurac
y: 0.8277
Epoch 12/40
1992/1992 [==============================] - 83s 42ms/step - loss: 0.6029 - accuracy: 0.9315 - val_loss: 0.9485 - val_accurac
y: 0.8320
Epoch 13/40
1992/1992 [==============================] - 80s 40ms/step - loss: 0.5933 - accuracy: 0.9344 - val_loss: 0.9552 - val_accurac
y: 0.8303
Epoch 14/40
1992/1992 [==============================] - 80s 40ms/step - loss: 0.5869 - accuracy: 0.9360 - val_loss: 0.9536 - val_accurac
y: 0.8331
Epoch 15/40
1992/1992 [==============================] - 81s 41ms/step - loss: 0.5818 - accuracy: 0.9374 - val_loss: 0.9846 - val_accurac
y: 0.8342
Epoch 16/40
1992/1992 [==============================] - 81s 41ms/step - loss: 0.5774 - accuracy: 0.9391 - val_loss: 1.0133 - val_accurac
y: 0.8297
Epoch 17/40
1992/1992 [==============================] - 80s 40ms/step - loss: 0.5744 - accuracy: 0.9400 - val_loss: 0.9964 - val_accurac
y: 0.8339
Epoch 18/40
1992/1992 [==============================] - 90s 45ms/step - loss: 0.5719 - accuracy: 0.9407 - val_loss: 1.0330 - val_accurac
y: 0.8314
Epoch 19/40
1992/1992 [==============================] - 82s 41ms/step - loss: 0.5695 - accuracy: 0.9411 - val_loss: 1.0191 - val_accurac
y: 0.8286
Epoch 20/40
1992/1992 [==============================] - 91s 45ms/step - loss: 0.5687 - accuracy: 0.9419 - val_loss: 1.0283 - val_accurac
y: 0.8356
Epoch 21/40
1992/1992 [==============================] - 85s 43ms/step - loss: 0.5655 - accuracy: 0.9433 - val_loss: 1.0516 - val_accurac
y: 0.8317
Epoch 22/40
1992/1992 [==============================] - 81s 41ms/step - loss: 0.5649 - accuracy: 0.9424 - val_loss: 1.0485 - val_accurac
```

```
y: 0.8272
Epoch 23/40
1992/1992 [==============================] - 79s 40ms/step - loss: 0.5641 - accuracy: 0.9446 - val_loss: 1.0375 - val_accurac
y: 0.8280
Epoch 24/40
1992/1992 [==============================] - 96s 48ms/step - loss: 0.5618 - accuracy: 0.9456 - val_loss: 1.0475 - val_accurac
y: 0.8286
Epoch 25/40
1992/1992 [==============================] - 79s 40ms/step - loss: 0.5612 - accuracy: 0.9462 - val_loss: 1.0561 - val_accurac
y: 0.8325
Epoch 26/40
1992/1992 [==============================] - 78s 39ms/step - loss: 0.5597 - accuracy: 0.9461 - val_loss: 1.0657 - val_accurac
y: 0.8249
Epoch 27/40
1992/1992 [==============================] - 79s 40ms/step - loss: 0.5585 - accuracy: 0.9467 - val_loss: 1.1004 - val_accurac
y: 0.8274
Epoch 28/40
1992/1992 [==============================] - 78s 39ms/step - loss: 0.5586 - accuracy: 0.9469 - val_loss: 1.0597 - val_accurac
y: 0.8286
Epoch 29/40
1992/1992 [==============================] - 79s 40ms/step - loss: 0.5580 - accuracy: 0.9478 - val_loss: 1.0770 - val_accurac
y: 0.8286
Epoch 30/40
1992/1992 [==============================] - 80s 40ms/step - loss: 0.5566 - accuracy: 0.9481 - val_loss: 1.0651 - val_accurac
y: 0.8339
Epoch 31/40
1992/1992 [==============================] - 79s 40ms/step - loss: 0.5558 - accuracy: 0.9479 - val_loss: 1.0853 - val_accurac
y: 0.8257
Epoch 32/40
1992/1992 [==============================] - 80s 40ms/step - loss: 0.5557 - accuracy: 0.9485 - val_loss: 1.0661 - val_accurac
y: 0.8342
Epoch 33/40
1992/1992 [==============================] - 78s 39ms/step - loss: 0.5551 - accuracy: 0.9504 - val_loss: 1.0775 - val_accurac
y: 0.8300
Epoch 34/40
1992/1992 [==============================] - 79s 40ms/step - loss: 0.5541 - accuracy: 0.9505 - val_loss: 1.0867 - val_accurac
y: 0.8294
Epoch 35/40
1992/1992 [==============================] - 78s 39ms/step - loss: 0.5530 - accuracy: 0.9509 - val_loss: 1.0931 - val_accurac
y: 0.8331
Epoch 36/40
1992/1992 [==============================] - 80s 40ms/step - loss: 0.5529 - accuracy: 0.9513 - val_loss: 1.0784 - val_accurac
y: 0.8308
Epoch 37/40
```

```
1992/1992 [==============================] - 81s 41ms/step - loss: 0.5527 - accuracy: 0.9503 - val_loss: 1.0733 - val_accurac
y: 0.8272
Epoch 38/40
1992/1992 [==============================] - 81s 41ms/step - loss: 0.5522 - accuracy: 0.9504 - val_loss: 1.0785 - val_accurac
y: 0.8300
Epoch 39/40
1992/1992 [==============================] - 81s 41ms/step - loss: 0.5509 - accuracy: 0.9518 - val_loss: 1.0989 - val_accurac
y: 0.8337
Epoch 40/40
1992/1992 [==============================] - 80s 40ms/step - loss: 0.5515 - accuracy: 0.9525 - val_loss: 1.0855 - val_accurac
y: 0.8286
```

Out[6]:

```
<tensorflow.python.keras.callbacks.History at 0x27580a46d90>
```

In [7]:

```python
base_model = resnet_v2.ResNet50V2(include_top=False,
                                  weights="imagenet",
                                  input_shape=(48,48,3))
base_model.trainable=True
model = Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(2048, activation='relu'),
    layers.Dense(2048, activation='relu'),
    layers.Dense(emotions_count, activation='softmax'),
])

model.compile(optimizer=adam.Adam(learning_rate=1e-4),
              loss=losses.CategoricalCrossentropy(),
              metrics = [model_acc])

model.fit(x=training_images,
          y=training_emotions,
          batch_size=32,
          epochs=40,
          validation_data=(test_images, test_emotions))
```

```
Epoch 1/40
1919/1992 [===========================>..] - ETA: 10s - loss: 1.2105 - model_acc: 0.6581
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_2356/3271828104.py in <module>
     15               metrics = [model_acc])
     16 
---> 17 model.fit(x=training_images,
     18         y=training_emotions,
     19         batch_size=32,

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py in fit(self, x, y, batch_size, epochs, verbose, call
backs, validation_split, validation_data, shuffle, class_weight, sample_weight, initial_epoch, steps_per_epoch, validation_st
eps, validation_batch_size, validation_freq, max_queue_size, workers, use_multiprocessing)
   1181                 _r=1):
   1182               callbacks.on_train_batch_begin(step)
-> 1183               tmp_logs = self.train_function(iterator)
   1184               if data_handler.should_sync:
   1185                 context.async_wait()

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py in train_function(iterator)
    853       def train_function(iterator):
    854         """Runs a training execution with one step."""
--> 855         return step_function(self, iterator)
    856 
    857     else:

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py in step_function(model, iterator)
    843 
    844       data = next(iterator)
--> 845       outputs = model.distribute_strategy.run(run_step, args=(data,))
    846       outputs = reduce_per_replica(
    847           outputs, self.distribute_strategy, reduction='first')

~\anaconda3\lib\site-packages\tensorflow\python\distribute\distribute_lib.py in run(***failed resolving arguments***)
   1283       fn = autograph.tf_convert(
   1284           fn, autograph_ctx.control_status_ctx(), convert_by_default=False)
-> 1285       return self._extended.call_for_each_replica(fn, args=args, kwargs=kwargs)
   1286 
   1287   def reduce(self, reduce_op, value, axis):

~\anaconda3\lib\site-packages\tensorflow\python\distribute\distribute_lib.py in call_for_each_replica(self, fn, args, kwargs)
   2831       kwargs = {}
   2832     with self._container_strategy().scope():
-> 2833       return self._call_for_each_replica(fn, args, kwargs)
   2834 
   2835   def _call_for_each_replica(self, fn, args, kwargs):

~\anaconda3\lib\site-packages\tensorflow\python\distribute\distribute_lib.py in _call_for_each_replica(self, fn, args, kwarg
s)
   3606   def _call_for_each_replica(self, fn, args, kwargs):
   3607     with ReplicaContext(self._container_strategy(), replica_id_in_sync_group=0):
```

```
-> 3608     return fn(*args, **kwargs)
   3609 
   3610   def _reduce_to(self, reduce_op, value, destinations, options):

~\anaconda3\lib\site-packages\tensorflow\python\autograph\impl\api.py in wrapper(*args, **kwargs)
    595   def wrapper(*args, **kwargs):
    596     with ag_ctx.ControlStatusCtx(status=ag_ctx.Status.UNSPECIFIED):
--> 597       return func(*args, **kwargs)
    598 
    599   if inspect.isfunction(func) or inspect.ismethod(func):

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py in run_step(data)
    836 
    837       def run_step(data):
--> 838         outputs = model.train_step(data)
    839         # Ensure counter is updated only if `train_step` succeeds.
    840         with ops.control_dependencies(_minimum_control_deps(outputs)):

~\anaconda3\lib\site-packages\tensorflow\python\keras\engine\training.py in train_step(self, data)
    797           y, y_pred, sample_weight, regularization_losses=self.losses)
    798     # Run backwards pass.
--> 799     self.optimizer.minimize(loss, self.trainable_variables, tape=tape)
    800     self.compiled_metrics.update_state(y, y_pred, sample_weight)
    801     # Collect metrics to return

~\anaconda3\lib\site-packages\tensorflow\python\keras\optimizer_v2\optimizer_v2.py in minimize(self, loss, var_list, grad_los
s, name, tape)
    526 
    527     """
--> 528     grads_and_vars = self._compute_gradients(
    529         loss, var_list=var_list, grad_loss=grad_loss, tape=tape)
    530     return self.apply_gradients(grads_and_vars, name=name)

~\anaconda3\lib\site-packages\tensorflow\python\keras\optimizer_v2\optimizer_v2.py in _compute_gradients(self, loss, var_lis
t, grad_loss, tape)
    578     var_list = nest.flatten(var_list)
    579     with ops.name_scope_v2(self._name + "/gradients"):
--> 580       grads_and_vars = self._get_gradients(tape, loss, var_list, grad_loss)
    581 
    582     self._assert_valid_dtypes([

~\anaconda3\lib\site-packages\tensorflow\python\keras\optimizer_v2\optimizer_v2.py in _get_gradients(self, tape, loss, var_li
st, grad_loss)
    471   def _get_gradients(self, tape, loss, var_list, grad_loss=None):
```

```
    472     """Called in `minimize` to compute gradients from loss."""
--> 473     grads = tape.gradient(loss, var_list, grad_loss)
    474     return list(zip(grads, var_list))
    475 

~\anaconda3\lib\site-packages\tensorflow\python\eager\backprop.py in gradient(self, target, sources, output_gradients, unconn
ected_gradients)
   1072                               for x in nest.flatten(output_gradients)]
   1073 
-> 1074     flat_grad = imperative_grad.imperative_grad(
   1075         self._tape,
   1076         flat_targets,

~\anaconda3\lib\site-packages\tensorflow\python\eager\imperative_grad.py in imperative_grad(tape, target, sources, output_gra
dients, sources_raw, unconnected_gradients)
     69         "Unknown value for unconnected_gradients: %r" % unconnected_gradients)
     70 
---> 71   return pywrap_tfe.TFE_Py_TapeGradient(
     72       tape._tape,  # pylint: disable=protected-access
     73       target,

~\anaconda3\lib\site-packages\tensorflow\python\eager\backprop.py in _gradient_function(op_name, attr_tuple, num_inputs, inpu
ts, outputs, out_grads, skip_input_indices, forward_pass_name_scope)
    157       gradient_name_scope += forward_pass_name_scope + "/"
    158     with ops.name_scope(gradient_name_scope):
--> 159       return grad_fn(mock_op, *out_grads)
    160   else:
    161     return grad_fn(mock_op, *out_grads)

~\anaconda3\lib\site-packages\tensorflow\python\ops\nn_grad.py in _Conv2DGrad(op, grad)
    579   # in Eager mode.
    580   return [
--> 581       gen_nn_ops.conv2d_backprop_input(
    582           shape_0,
    583           op.inputs[1],

~\anaconda3\lib\site-packages\tensorflow\python\ops\gen_nn_ops.py in conv2d_backprop_input(input_sizes, filter, out_backprop,
strides, padding, use_cudnn_on_gpu, explicit_paddings, data_format, dilations, name)
   1237   if tld.is_eager:
   1238     try:
-> 1239       _result = pywrap_tfe.TFE_Py_FastPathExecute(
   1240         _ctx, "Conv2DBackpropInput", name, input_sizes, filter, out_backprop,
   1241         "strides", strides, "use_cudnn_on_gpu", use_cudnn_on_gpu, "padding",
KeyboardInterrupt:
```

```
In [ ]:
```