In [4]:

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf

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

samples = 35393 # 2~35394
training_samples = 28317  # 2~28318 (Training)
validation_samples = 3541 # 28319~31859 (PublicTest)
test_samples = 3535       # 31860~35394 (PrivateTest)
```

In [5]:

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

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)
```

```
(35393, 48, 48, 1)
(35393, 8)
(35393, 8)
```

In [6]:

```
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)
print("images shape:", images.shape)
print("emotions shape:", emotions.shape)
```

```
images shape: (35393, 48, 48, 3)
```

```
emotions shape: (35393, 8)
```

In [7]:

```
# images = tf.image.resize(images, [224,224])
# print("images shape:", images.shape)
```

In [8]:

```
from tensorflow.python.keras import layers
# choose one method:
images = layers.Rescaling(1./127.5, offset= -1)(images)
```

In [9]:

```
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: (31858, 48, 48, 3)
training_emotions shape: (31858, 8)
test_images shape: (3535, 48, 48, 3)
test_emotions shape: (3535, 8)
```

In [26]:

```
from tensorflow.python.keras.applications import vgg16, resnet
from tensorflow.python.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.python.keras.models import Model
from tensorflow.python.keras import layers, Sequential

base_model = resnet.ResNet101(include_top=False, weights="imagenet", input_shape=(48,48,3))
#base_model = resnet.ResNet50(include_top=False, weights="imagenet", input_shape=(48,48,3))
base_model.trainable=False
model = Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(4096, activation='relu'),
    layers.Dense(emotions_count, activation='softmax'),
])
#model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet101_weights_tf_dim_ordering_tf_ker
nels_notop.h5
171450368/171446536 [==============================] - 7s 0us/step
```

In [27]:

```
from tensorflow.python.keras import losses, metrics
from tensorflow.python.keras.optimizer_v2 import adam
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
model.compile(optimizer=adam.Adam(learning_rate=1e-4), loss=losses.CategoricalCrossentropy(), metrics = [model_acc])
```

In [28]:

```
model.fit(x=training_images,
          y=training_emotions,
          batch_size=32,
          epochs=25,
          validation_data=(test_images, test_emotions))
```

```
Epoch 1/25
996/996 [==============================] - 154s 155ms/step - loss: 1.5678 - accuracy: 0.4035 - val_loss: 1.5528 - val_accuracy: 0.
4170
Epoch 2/25
996/996 [==============================] - 156s 157ms/step - loss: 1.5095 - accuracy: 0.4337 - val_loss: 1.5530 - val_accuracy: 0.
4141
Epoch 3/25
996/996 [==============================] - 156s 157ms/step - loss: 1.4832 - accuracy: 0.4506 - val_loss: 1.5370 - val_accuracy: 0.
4076
Epoch 4/25
996/996 [==============================] - 156s 156ms/step - loss: 1.4619 - accuracy: 0.4600 - val_loss: 1.5290 - val_accuracy: 0.
4294
Epoch 5/25
996/996 [==============================] - 157s 157ms/step - loss: 1.4514 - accuracy: 0.4646 - val_loss: 1.5351 - val_accuracy: 0.
```

```
4354
Epoch 6/25
996/996 [==============================] - 157s 158ms/step - loss: 1.4360 - accuracy: 0.4721 - val_loss: 1.4655 - val_accuracy: 0.
4600
Epoch 7/25
996/996 [==============================] - 157s 158ms/step - loss: 1.4231 - accuracy: 0.4786 - val_loss: 1.4715 - val_accuracy: 0.
4651
Epoch 8/25
996/996 [==============================] - 158s 159ms/step - loss: 1.4139 - accuracy: 0.4801 - val_loss: 1.4616 - val_accuracy: 0.
4580
Epoch 9/25
996/996 [==============================] - 169s 170ms/step - loss: 1.4050 - accuracy: 0.4854 - val_loss: 1.4750 - val_accuracy: 0.
4577
Epoch 10/25
996/996 [==============================] - 155s 155ms/step - loss: 1.3964 - accuracy: 0.4851 - val_loss: 1.4767 - val_accuracy: 0.
4636
Epoch 11/25
996/996 [==============================] - 165s 166ms/step - loss: 1.3859 - accuracy: 0.4936 - val_loss: 1.4476 - val_accuracy: 0.
4673
Epoch 12/25
996/996 [==============================] - 167s 167ms/step - loss: 1.3792 - accuracy: 0.4962 - val_loss: 1.4802 - val_accuracy: 0.
4634
Epoch 13/25
996/996 [==============================] - 165s 166ms/step - loss: 1.3701 - accuracy: 0.4973 - val_loss: 1.4913 - val_accuracy: 0.
4286
Epoch 14/25
996/996 [==============================] - 206s 207ms/step - loss: 1.3641 - accuracy: 0.5028 - val_loss: 1.4472 - val_accuracy: 0.
4625
Epoch 15/25
996/996 [==============================] - 289s 291ms/step - loss: 1.3547 - accuracy: 0.5013 - val_loss: 1.4536 - val_accuracy: 0.
4501
Epoch 16/25
996/996 [==============================] - 244s 245ms/step - loss: 1.3433 - accuracy: 0.5087 - val_loss: 1.4361 - val_accuracy: 0.
4710
Epoch 17/25
996/996 [==============================] - 221s 222ms/step - loss: 1.3407 - accuracy: 0.5089 - val_loss: 1.4323 - val_accuracy: 0.
4750
Epoch 18/25
996/996 [==============================] - 180s 181ms/step - loss: 1.3333 - accuracy: 0.5131 - val_loss: 1.4286 - val_accuracy: 0.
4803
Epoch 19/25
996/996 [==============================] - 252s 253ms/step - loss: 1.3226 - accuracy: 0.5160 - val_loss: 1.4470 - val_accuracy: 0.
4659
Epoch 20/25
```

```
996/996 [==============================] - 333s 335ms/step - loss: 1.3159 - accuracy: 0.5199 - val_loss: 1.4296 - val_accuracy: 0.
4818
Epoch 21/25
996/996 [==============================] - 288s 289ms/step - loss: 1.3074 - accuracy: 0.5246 - val_loss: 1.4437 - val_accuracy: 0.
4741
Epoch 22/25
996/996 [==============================] - 263s 264ms/step - loss: 1.3022 - accuracy: 0.5222 - val_loss: 1.4450 - val_accuracy: 0.
4670
Epoch 23/25
996/996 [==============================] - 217s 217ms/step - loss: 1.2968 - accuracy: 0.5249 - val_loss: 1.4385 - val_accuracy: 0.
4738
Epoch 24/25
996/996 [==============================] - 159s 160ms/step - loss: 1.2853 - accuracy: 0.5325 - val_loss: 1.4277 - val_accuracy: 0.
4761
Epoch 25/25
996/996 [==============================] - 158s 159ms/step - loss: 1.2805 - accuracy: 0.5327 - val_loss: 1.4656 - val_accuracy: 0.
4583
```

Out[28]:

```
<tensorflow.python.keras.callbacks.History at 0x2c45217cc40>
```

In [ ]: