In [1]:

```
import csv, json, random
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
```

In [2]:

```
samples = 35393 # 2~35394

image_height = 48
image_width = 48
emotions_count = 8

images = []
emotions = []
emotion_labels = []
with open('./dataset.csv') as file:
    reader = csv.reader(file)
    for line in reader:
        image_pixels = line[-1].split()
        if len(image_pixels) == 1:
            emotion_labels = line[2:2+emotions_count]
            assert(emotion_labels == ['neutral', 'happiness', 'surprise', 'sadness', 'anger
            continue

        image = []
        for i in range(image_height):
            row = []
            for j in range(image_width):
                row.append(image_pixels[image_width*i+j])
            row = list(map(int, row))
            image.append(row)
        images.append(image)

        emotion = []
        for i in range(2, 2+emotions_count):
            emotion.append(line[i])
        emotion = list(map(float, emotion))
        emotions.append(emotion)

images = np.array(images).reshape(samples, image_height, image_width, 1)
emotions = np.array(emotions)

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


# # check the last 9 images
# plt.figure(figsize=(10, 10))
# for i in range(35384,35393):
#     ax = plt.subplot(3, 3, i-35383)
#     plt.imshow(images[i].astype("uint8"))
#     plt.title(emotion_labels[np.argmax(emotions[i])])
#     plt.axis("off")
```

```
images shape: (35393, 48, 48, 1)
emotions shape: (35393, 8)
emotion_labels: ['neutral', 'happiness', 'surprise', 'sadness', 'anger', 'di
sgust', 'fear', 'contempt']
```

In [3]:

```
for i in range(emotions.shape[0]):
    max_indices = []
    maximum = np.amax(emotions[i])
    for j in range(emotions.shape[1]):
        if emotions[i][j] == maximum:
            max_indices.append(j)
    no_of_maxs = len(max_indices)
    for j in range(emotions.shape[1]):
        if j in max_indices:
            emotions[i][j] = 1/no_of_maxs
        else:
            emotions[i][j] = 0
emotions
```

Out[3]:

```
array([[1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       [1., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 1., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [4]:

```
images = tf.convert_to_tensor(images)
emotions = tf.convert_to_tensor(emotions)
images = tf.image.grayscale_to_rgb(images)
print("images shape:", images.shape)
print("emotions shape:", emotions.shape)
```

```
images shape: (35393, 48, 48, 3)
emotions shape: (35393, 8)
```

In [5]:

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

In [6]:

```
from tensorflow.python.keras.applications.vgg16 import preprocess_input
from tensorflow.python.keras import layers

# choose one method:
images = layers.Rescaling(1./127.5, offset= -1)(images)
#images = preprocess_input(images)
```

In [7]:

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

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 [8]:

```
# # check the last 9 images
# plt.figure(figsize=(10, 10))
# for i in range(6):
#     ax = plt.subplot(3, 3, i+1)
#     plt.imshow(special_images[i].astype("uint8"))
#     plt.axis("off")
```

In [9]:

```
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 [27]:

```
from tensorflow.python.keras.applications.vgg16 import VGG16
from tensorflow.python.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.python.keras.models import Model
from tensorflow.python.keras import layers, Sequential

base_model = VGG16(include_top=False, weights="imagenet", input_shape=(48,48,3))
base_model.trainable=False
model = Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(1024, activation='relu'),
    layers.Dense(emotions_count, activation='softmax'),
])
#model.summary()
```

In [28]:

```
from tensorflow.python.keras import losses, metrics
from tensorflow.python.keras.optimizer_v2 import adam

#m = top_k_categorical_accuracy(y_true, y_pred, k=3)
#model.compile(optimizer='sgd', loss=losses.CategoricalCrossentropy(), metrics = [metrics.C
model.compile(optimizer=adam.Adam(learning_rate=1e-4), loss=losses.CategoricalCrossentropy(
```

In [29]:

```
model.fit(x=training_images,
        y=training_emotions,
        batch_size=32,
        epochs=10,
        validation_data=(test_images, test_emotions))
```

```
Epoch 1/10
996/996 [==============================] - 897s 900ms/step - loss: 1.4026 -
model_acc: 0.4917 - val_loss: 1.3558 - val_model_acc: 0.5257
Epoch 2/10
996/996 [==============================] - 942s 946ms/step - loss: 1.2900 -
model_acc: 0.5423 - val_loss: 1.3190 - val_model_acc: 0.5313
Epoch 3/10
996/996 [==============================] - 994s 998ms/step - loss: 1.2468 -
model_acc: 0.5578 - val_loss: 1.2915 - val_model_acc: 0.5431
Epoch 4/10
996/996 [==============================] - 867s 870ms/step - loss: 1.2165 -
model_acc: 0.5705 - val_loss: 1.2762 - val_model_acc: 0.5468
Epoch 5/10
996/996 [==============================] - 862s 865ms/step - loss: 1.1921 -
model_acc: 0.5784 - val_loss: 1.2604 - val_model_acc: 0.5505
Epoch 6/10
996/996 [==============================] - 864s 868ms/step - loss: 1.1687 -
model_acc: 0.5875 - val_loss: 1.2560 - val_model_acc: 0.5521
Epoch 7/10
996/996 [==============================] - 866s 869ms/step - loss: 1.1467 -
model_acc: 0.5963 - val_loss: 1.2357 - val_model_acc: 0.5614
Epoch 8/10
996/996 [==============================] - 864s 868ms/step - loss: 1.1265 -
model_acc: 0.6069 - val_loss: 1.2302 - val_model_acc: 0.5626
Epoch 9/10
996/996 [==============================] - 863s 867ms/step - loss: 1.1061 -
model_acc: 0.6144 - val_loss: 1.2215 - val_model_acc: 0.5692
Epoch 10/10
996/996 [==============================] - 863s 867ms/step - loss: 1.0873 -
model_acc: 0.6218 - val_loss: 1.2161 - val_model_acc: 0.5673
```

Out[29]:

```
<tensorflow.python.keras.callbacks.History at 0x267943d95e0>
```

In [30]:

```
base_model = VGG16(include_top=False, weights="imagenet", input_shape=(48,48,3))
base_model.trainable=False
model = Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(1024, activation='relu'),
    layers.Dense(emotions_count, activation='softmax'),
])
model.compile(optimizer=adam.Adam(learning_rate=4e-5), loss=losses.CategoricalCrossentropy(
model.fit(x=training_images,
        y=training_emotions,
        batch_size=32,
        epochs=20,
        validation_data=(test_images, test_emotions))
```

```
Epoch 1/20
996/996 [==============================] - 862s 865ms/step - loss: 1.4667 -
model_acc: 0.4660 - val_loss: 1.4012 - val_model_acc: 0.5121
Epoch 2/20
996/996 [==============================] - 865s 868ms/step - loss: 1.3431 -
model_acc: 0.5220 - val_loss: 1.3574 - val_model_acc: 0.5254
Epoch 3/20
996/996 [==============================] - 864s 868ms/step - loss: 1.3037 -
model_acc: 0.5378 - val_loss: 1.3322 - val_model_acc: 0.5330
Epoch 4/20
996/996 [==============================] - 866s 869ms/step - loss: 1.2772 -
model_acc: 0.5471 - val_loss: 1.3172 - val_model_acc: 0.5364
Epoch 5/20
996/996 [==============================] - 864s 868ms/step - loss: 1.2566 -
model_acc: 0.5550 - val_loss: 1.3024 - val_model_acc: 0.5468
Epoch 6/20
996/996 [==============================] - 862s 866ms/step - loss: 1.2410 -
model_acc: 0.5609 - val_loss: 1.2951 - val_model_acc: 0.5507
Epoch 7/20
996/996 [==============================] - 863s 867ms/step - loss: 1.2260 -
model_acc: 0.5681 - val_loss: 1.2817 - val_model_acc: 0.5510
Epoch 8/20
996/996 [==============================] - 862s 865ms/step - loss: 1.2136 -
model_acc: 0.5712 - val_loss: 1.2767 - val_model_acc: 0.5501
Epoch 9/20
996/996 [==============================] - 863s 867ms/step - loss: 1.2015 -
model_acc: 0.5749 - val_loss: 1.2689 - val_model_acc: 0.5521
Epoch 10/20
996/996 [==============================] - 861s 865ms/step - loss: 1.1912 -
model_acc: 0.5800 - val_loss: 1.2646 - val_model_acc: 0.5527
Epoch 11/20
996/996 [==============================] - 862s 865ms/step - loss: 1.1800 -
model_acc: 0.5836 - val_loss: 1.2606 - val_model_acc: 0.5583
Epoch 12/20
996/996 [==============================] - 862s 865ms/step - loss: 1.1704 -
model_acc: 0.5868 - val_loss: 1.2536 - val_model_acc: 0.5555
Epoch 13/20
996/996 [==============================] - 862s 866ms/step - loss: 1.1610 -
model_acc: 0.5931 - val_loss: 1.2536 - val_model_acc: 0.5558
Epoch 14/20
996/996 [==============================] - 863s 866ms/step - loss: 1.1522 -
model_acc: 0.5956 - val_loss: 1.2504 - val_model_acc: 0.5583
Epoch 15/20
996/996 [==============================] - 862s 865ms/step - loss: 1.1427 -
```

```
model_acc: 0.6000 - val_loss: 1.2445 - val_model_acc: 0.5611
Epoch 16/20
996/996 [==============================] - 862s 865ms/step - loss: 1.1342 -
model_acc: 0.6048 - val_loss: 1.2401 - val_model_acc: 0.5636
Epoch 17/20
996/996 [==============================] - 861s 865ms/step - loss: 1.1253 -
model_acc: 0.6087 - val_loss: 1.2403 - val_model_acc: 0.5651
Epoch 18/20
996/996 [==============================] - 862s 865ms/step - loss: 1.1177 -
model_acc: 0.6108 - val_loss: 1.2306 - val_model_acc: 0.5625
Epoch 19/20
996/996 [==============================] - 861s 865ms/step - loss: 1.1088 -
model_acc: 0.6170 - val_loss: 1.2278 - val_model_acc: 0.5670
Epoch 20/20
996/996 [==============================] - 862s 865ms/step - loss: 1.1008 -
model_acc: 0.6190 - val_loss: 1.2237 - val_model_acc: 0.5752
```

Out[30]:

```
<tensorflow.python.keras.callbacks.History at 0x267956428b0>
```

In [31]:

```
base_model = VGG16(include_top=False, weights="imagenet", input_shape=(48,48,3))
base_model.trainable=False
model = Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(1024, activation='relu'),
    layers.Dense(emotions_count, activation='softmax'),
])
model.compile(optimizer=adam.Adam(learning_rate=5e-4), loss=losses.CategoricalCrossentropy(
model.fit(x=training_images,
        y=training_emotions,
        batch_size=32,
        epochs=10,
        validation_data=(test_images, test_emotions))
```

```
Epoch 1/10
996/996 [==============================] - 862s 866ms/step - loss: 1.3431 -
model_acc: 0.5150 - val_loss: 1.2840 - val_model_acc: 0.5507
Epoch 2/10
996/996 [==============================] - 865s 868ms/step - loss: 1.2302 -
model_acc: 0.5621 - val_loss: 1.2525 - val_model_acc: 0.5558
Epoch 3/10
996/996 [==============================] - 866s 869ms/step - loss: 1.1661 -
model_acc: 0.5841 - val_loss: 1.2267 - val_model_acc: 0.5591
Epoch 4/10
996/996 [==============================] - 864s 868ms/step - loss: 1.1060 -
model_acc: 0.6092 - val_loss: 1.2135 - val_model_acc: 0.5670
Epoch 5/10
996/996 [==============================] - 863s 867ms/step - loss: 1.0485 -
model_acc: 0.6324 - val_loss: 1.1799 - val_model_acc: 0.5807
Epoch 6/10
996/996 [==============================] - 864s 867ms/step - loss: 0.9929 -
model_acc: 0.6579 - val_loss: 1.1798 - val_model_acc: 0.5867
Epoch 7/10
996/996 [==============================] - 863s 867ms/step - loss: 0.9351 -
model_acc: 0.6829 - val_loss: 1.1924 - val_model_acc: 0.5805
Epoch 8/10
996/996 [==============================] - 864s 868ms/step - loss: 0.8780 -
model_acc: 0.7021 - val_loss: 1.1847 - val_model_acc: 0.5912
Epoch 9/10
996/996 [==============================] - 865s 868ms/step - loss: 0.8239 -
model_acc: 0.7263 - val_loss: 1.1987 - val_model_acc: 0.5971
Epoch 10/10
996/996 [==============================] - 862s 866ms/step - loss: 0.7712 -
model_acc: 0.7470 - val_loss: 1.1855 - val_model_acc: 0.5949
```

Out[31]:

```
<tensorflow.python.keras.callbacks.History at 0x2679788a8b0>
```

In [32]:

```
base_model = VGG16(include_top=False, weights="imagenet", input_shape=(48,48,3))
base_model.trainable=False
model = Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(2048, activation='relu'),
    layers.Dense(emotions_count, activation='softmax'),
])
model.compile(optimizer=adam.Adam(learning_rate=1e-4), loss=losses.CategoricalCrossentropy(
model.fit(x=training_images,
        y=training_emotions,
        batch_size=32,
        epochs=15,
        validation_data=(test_images, test_emotions))
```

```
Epoch 1/15
996/996 [==============================] - 856s 860ms/step - loss: 1.3780
- model_acc: 0.5010 - val_loss: 1.3402 - val_model_acc: 0.5276
Epoch 2/15
996/996 [==============================] - 857s 860ms/step - loss: 1.2703
- model_acc: 0.5462 - val_loss: 1.3046 - val_model_acc: 0.5343
Epoch 3/15
996/996 [==============================] - 858s 861ms/step - loss: 1.2251
- model_acc: 0.5652 - val_loss: 1.2696 - val_model_acc: 0.5496
Epoch 4/15
996/996 [==============================] - 866s 870ms/step - loss: 1.1894
- model_acc: 0.5805 - val_loss: 1.2607 - val_model_acc: 0.5569
Epoch 5/15
996/996 [==============================] - 861s 865ms/step - loss: 1.1584
- model_acc: 0.5928 - val_loss: 1.2372 - val_model_acc: 0.5653
Epoch 6/15
996/996 [==============================] - 862s 865ms/step - loss: 1.1272
- model_acc: 0.6063 - val_loss: 1.2275 - val_model_acc: 0.5679
Epoch 7/15
996/996 [==============================] - 860s 863ms/step - loss: 1.1012
- model_acc: 0.6158 - val_loss: 1.2330 - val_model_acc: 0.5633
Epoch 8/15
996/996 [==============================] - 859s 863ms/step - loss: 1.0730
- model_acc: 0.6287 - val_loss: 1.2154 - val_model_acc: 0.5715
Epoch 9/15
996/996 [==============================] - 859s 863ms/step - loss: 1.0482
- model_acc: 0.6392 - val_loss: 1.1958 - val_model_acc: 0.5777
Epoch 10/15
996/996 [==============================] - 858s 862ms/step - loss: 1.0212
- model_acc: 0.6516 - val_loss: 1.1913 - val_model_acc: 0.5850
Epoch 11/15
996/996 [==============================] - 859s 863ms/step - loss: 0.9959
- model_acc: 0.6613 - val_loss: 1.1939 - val_model_acc: 0.5758
Epoch 12/15
996/996 [==============================] - 859s 863ms/step - loss: 0.9700
- model_acc: 0.6712 - val_loss: 1.1828 - val_model_acc: 0.5850
Epoch 13/15
996/996 [==============================] - 859s 863ms/step - loss: 0.9460
- model_acc: 0.6838 - val_loss: 1.1759 - val_model_acc: 0.5923
Epoch 14/15
996/996 [==============================] - 859s 862ms/step - loss: 0.9223
- model_acc: 0.6939 - val_loss: 1.1677 - val_model_acc: 0.5903
Epoch 15/15
```

```
996/996 [==============================] - 859s 863ms/step - loss: 0.8962
- model_acc: 0.7054 - val_loss: 1.1791 - val_model_acc: 0.5887
```

Out[32]:

```
<tensorflow.python.keras.callbacks.History at 0x2679c4f6460>
```

In [ ]:

In [ ]:

In [ ]: