영어 모음을 나타내는

# CNN model을 활용한 수화 이미지 분류하기

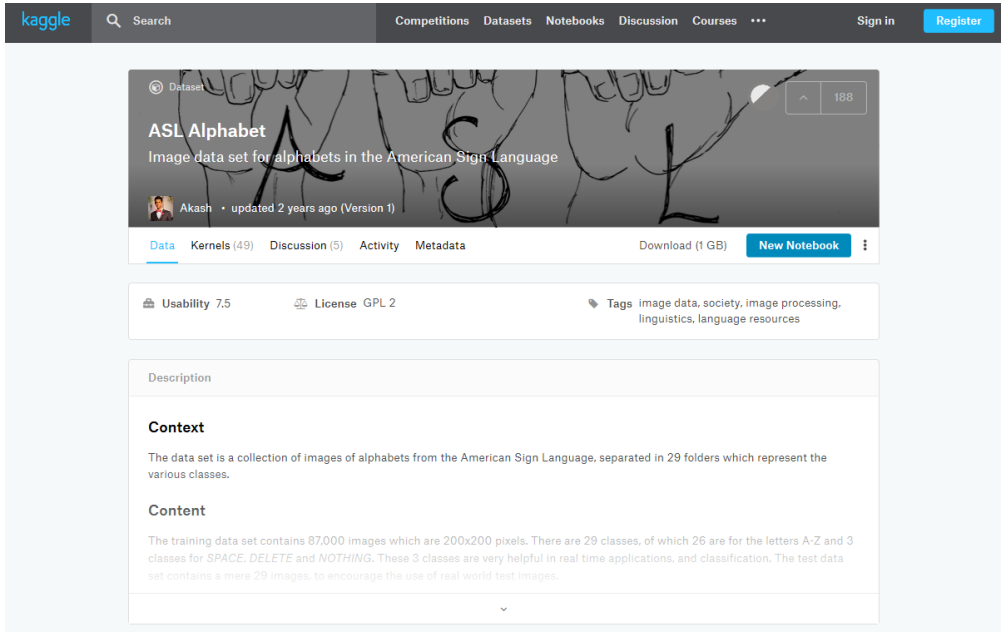
18학번 양주미

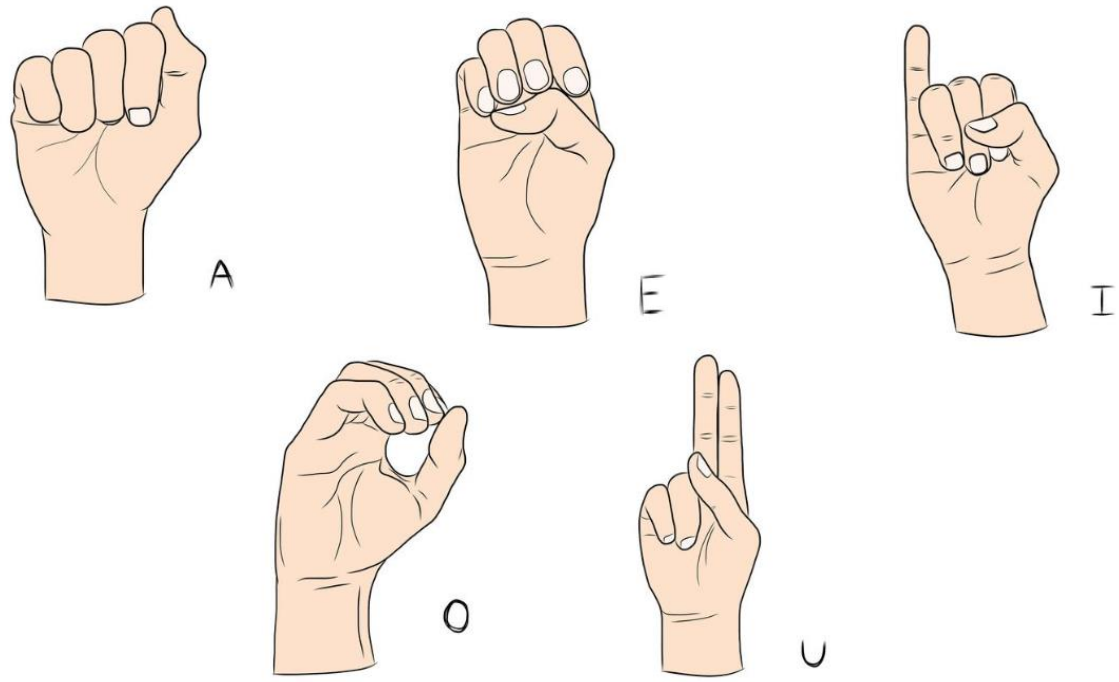
# 데이터 수집 예시

◆ 데이터 소스 공유 홈페이지 kaggle 이용

- https://www.kaggle.com/grassknoted/asl-alphabet

  : kaggle에서 알파벳을 나타내는 수화 이미지 데이터 15000장 수집 (각 알파벳 모음 당 3000장)



# 적용 기법 및 모델 설계

1 수집된 수화 이미지 dataset은 알파벳으로 이루어져 있다.

- 26가지의 알파벳 중 단어 형성의 기초가 되는 모음 5가지 a, e, i, o, u를 나타내는 수화 이미지를 알파벳으로 변환하는 것이 목표

2 5가지의 알파벳을 분류하기 위해 CNN 다중 분류 코드 사용

3 5가지 알파벳 각각에 대해 약 3000개의 Train data 적용

# 1 Code

```
import tensorflow as tf
from tensorflow.keras import layers, utils
import numpy as np
import pandas as pd
import os
import keras
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, Dropout, BatchNormalization
from keras.preprocessing.image import ImageDataGenerator, load_img
from keras.utils import to_categorical
from keras.models import Sequential
from keras import regularizers
from sklearn.model_selection import train_test_split
import cv2
import matplotlib.pyplot as plt
import seaborn as sns
print(os.listdir("C:₩₩Users₩₩User₩₩Desktop₩₩asl_alphabet"))      #경로 설정과 library 불러오기
fpath = "C:₩₩Users₩₩User₩₩Desktop₩₩asl_alphabet₩₩asl_alphabet_train"
categories = os.listdir(fpath)
print("No. of categories of images in the train set = ",len(categories))
train_dir = 'C:₩₩Users₩₩User₩₩Desktop₩₩asl_alphabet₩₩asl_alphabet_train'
test_dir = 'C:₩₩Users₩₩User₩₩Desktop₩₩asl_alphabet₩₩asl_alphabet_test'
```

→ 라이브러리 호출

# 카테고리 별 이미지 확인 및 출력

```
def load_unique():
    size_img = 64,64
    images_for_plot = []
    labels_for_plot = []
    for folder in os.listdir(train_dir):
        for file in os.listdir(train_dir + '/' + folder):
            filepath = train_dir + '/' + folder + '/' + file
            image = cv2.imread(filepath)
            final_img = cv2.resize(image, size_img)
            final_img = cv2.cvtColor(final_img, cv2.COLOR_BGR2RGB)
            images_for_plot.append(final_img)
            labels_for_plot.append(folder)
            break
    return images_for_plot, labels_for_plot

images_for_plot, labels_for_plot = load_unique()
print("unique_labels = ", labels_for_plot)
```

```
fig = plt.figure(figsize = (15,15))
def plot_images(fig, image, label, row, col, index):
    fig.add_subplot(row, col, index)
    plt.axis('off')
    plt.imshow(image)
    plt.title(label)
    return
```

이미지가 폴더에 잘 들어있는지 확인

```
image_index = 0
row = 2
col = 3
for i in range(1,(row*col)):
    plot_images(fig, images_for_plot[image_index], labels_for_plot[image_index], row, col, i)
    image_index = image_index + 1
plt.show()
```

# 카테고리 별 이미지 확인 및 출력

```
labels_dict = {'A':0,'E':1,'I':2,'O':3,'U':4}

def load_data():
    images = []
    labels = []
    size = 64,64
    print("LOADING DATA FROM : ",end = "")
    for folder in os.listdir(train_dir):
        print(folder, end = ' | ')
        for image in os.listdir(train_dir + "/" + folder):
            temp_img = cv2.imread(train_dir + '/' + folder + '/' + image)
            temp_img = cv2.resize(temp_img, size)
            images.append(temp_img)
            if folder == 'A':
                labels.append(labels_dict['A'])
            elif folder == 'E':
                labels.append(labels_dict['E'])
            elif folder == 'I':
                labels.append(labels_dict['I'])
            elif folder == 'O':
                labels.append(labels_dict['O'])
            elif folder == 'U':
                labels.append(labels_dict['U'])

    images = np.array(images)
    images = images.astype('float32')/255.0

    templabels = labels

    labels = keras.utils.to_categorical(labels)

    X_train, X_test, Y_train, Y_test = train_test_split(images, labels, test_size = 0.05)

    print()
    print('Loaded', len(X_train),'images for training,','Train data shape =',X_train.shape)
    print('Loaded', len(X_test),'images for testing','Test data shape =',X_test.shape)

    return X_train, X_test, Y_train, Y_test, templabels
```

→ 이미지 파일 Labelling

## 모델 구조 정의

```
def create_model():

    model = Sequential()
    model.add(Conv2D(16, kernel_size = [3,3], padding = 'same', activation = 'relu', input_shape = (64,64,3)))
    model.add(Conv2D(32, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(MaxPool2D(pool_size = [3,3]))

    model.add(Conv2D(32, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(Conv2D(64, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(MaxPool2D(pool_size = [3,3]))

    model.add(Conv2D(128, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(Conv2D(256, kernel_size = [3,3], padding = 'same', activation = 'relu'))
    model.add(MaxPool2D(pool_size = [3,3]))

    model.add(BatchNormalization())

    model.add(Flatten())
    model.add(Dropout(0.5))
    model.add(Dense(512, activation = 'relu'))
    model.add(Dense(5, activation = 'softmax')) #분류  class가 5개니까 마지막 출력노드는 5개

    model.compile(optimizer = 'adam', loss = keras.losses.categorical_crossentropy, metrics = ["accuracy"])

    print("MODEL CREATED")
    model.summary()

    return model
```

모델 구축하기

## 데이터 훈련

```
def fit_model():
    model_hist = model.fit(X_train, Y_train, batch_size = 64, epochs = 150, validation_split = 0.1)
    early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=5)
    return model_hist
```

## 모델 구조와 학습 과정 출력 모델 저장

```
model = create_model()
curr_model_hist = fit_model()
score = model.evaluate(X_test, Y_test, verbose=1)
print('test_loss:', score[0], ' , test_acc:', score[1])
model.save("finger.h5")
```

## Loss/Accuracy 그래프 출력

```
import matplotlib.pyplot as plt

fig, loss_ax = plt.subplots(figsize=(10, 5))
acc_ax = loss_ax.twinx()

loss_ax.plot(curr_model_hist.history['loss'], 'y', label='train loss')
loss_ax.plot(curr_model_hist.history['val_loss'], 'r', label='val loss')
acc_ax.plot(curr_model_hist.history['acc'], 'b', label='train acc')
acc_ax.plot(curr_model_hist.history['val_acc'], 'g', label='val acc')

loss_ax.set_xlabel('epoch')
loss_ax.set_ylabel('loss')
acc_ax.set_ylabel('accuray')
loss_ax.legend(loc='upper left')
acc_ax.legend(loc='lower left')

plt.show()
```

## Test data로 평가하기

```
evaluate_metrics = model.evaluate(X_test, Y_test)      #test해보기
print("₩nTest Accuracy = ", "{:.2f}%".format(evaluate_metrics[1]*100),"₩nTest loss = " ,"{:.6f}".format(evaluate_metrics[0]))
```

## Image Data Preprocessing

```
def load_test_data():
    images = []
    names = []
    size = 64,64
    for image in os.listdir(test_dir):
        temp = cv2.imread(test_dir + '/' + image)
        temp = cv2.resize(temp, size)
        images.append(temp)
        names.append(image)
    images = np.array(images)
    images = images.astype('float32')/255.0
    return images, names

test_images, test_img_names = load_test_data()
```

## 카테고리 예측하기

```
def give_predictions(test_data):
    predictions_classes = []
    for image in test_data:
        image = image.reshape(1,64,64,3)
        pred = model.predict_classes(image)
        predictions_classes.append(pred[0])
    return predictions_classes

predictions = give_predictions(test_images)
```

## 카테고리 예측

```
def get_labels_for_plot(predictions):
    predictions_labels = []
    for i in range(len(predictions)):
        for ins in labels_dict:
            if predictions[i] == labels_dict[ins]:
                predictions_labels.append(ins)
                break
    return predictions_labels

predictions_labels_plot = get_labels_for_plot(predictions)
```

## 예측한 결과값과 이미지 출력

```
predfigure = plt.figure(figsize = (13,13))
def plot_image_1(fig, image, label, prediction, predictions_label, row, col, index):
    fig.add_subplot(row, col, index)
    plt.axis('off')
    plt.imshow(image)
    title = "prediction : [" + str(predictions_label) + "] "+ "₩n" + label
    plt.title(title)
    return

image_index = 0
row = 2
col = 3
for i in range(1,(row*col)):
    plot_image_1(predfigure, test_images[image_index], test_img_names[image_index], predictions[image_index], predictions_labels_plot[image_index], row, col, i)
    image_index = image_index + 1
plt.show()
```

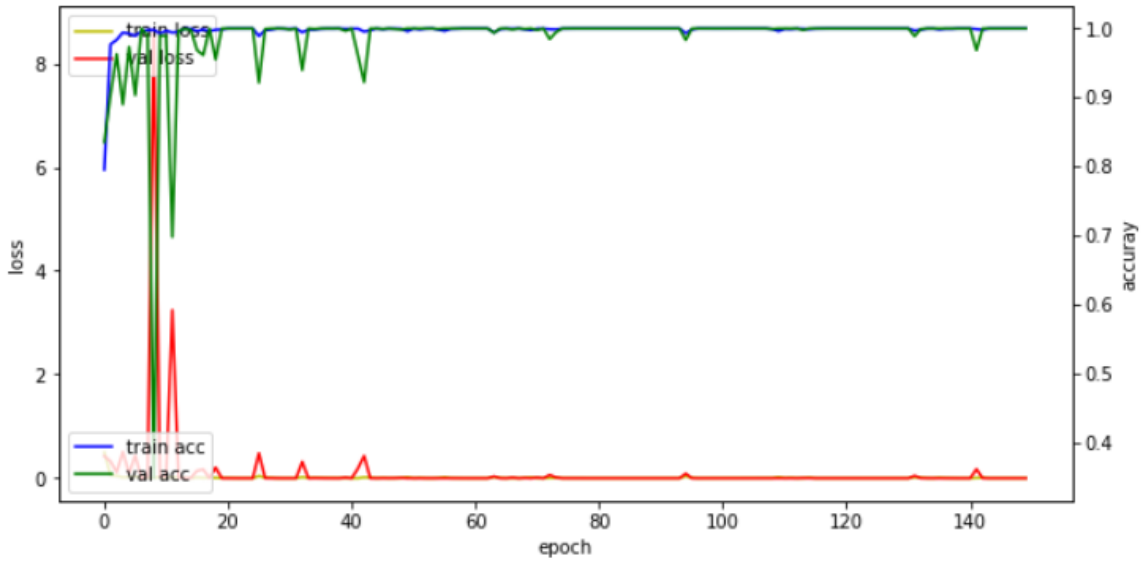
# 2 Parameter

```
MODEL CREATED
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 64, 64, 16)        448
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 21, 21, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 21, 21, 32)        9248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 21, 21, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 7, 128)         73856
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 7, 7, 256)         295168
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
batch_normalization_1 (Batch (None, 2, 2, 256)         1024
_________________________________________________________________
flatten_1 (Flatten)          (None, 1024)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_2 (Dense)              (None, 5)                 2565
=================================================================
Total params: 930,245
Trainable params: 929,733
Non-trainable params: 512
```

# 딥러닝 모형 실험

5개의 수화 이미지 데이터를 모델을 이용해 분류한 결과, 다음과 같이 분류가 되었다.

# 실험 결과 분석




(a) Train data acc/loss

```
12825/12825 [==============================] - 125s 10ms/step - loss: 1.5129e-05 - acc: 1.0000 - val_loss: 3.3158e-07 - val_acc: 1.0000
Epoch 143/150
12825/12825 [==============================] - 125s 10ms/step - loss: 1.5336e-06 - acc: 1.0000 - val_loss: 3.0895e-07 - val_acc: 1.0000
Epoch 144/150
12825/12825 [==============================] - 126s 10ms/step - loss: 7.1513e-04 - acc: 0.9996 - val_loss: 5.9401e-06 - val_acc: 1.0000
Epoch 145/150
12825/12825 [==============================] - 125s 10ms/step - loss: 6.8216e-05 - acc: 1.0000 - val_loss: 5.6566e-07 - val_acc: 1.0000
Epoch 146/150
12825/12825 [==============================] - 125s 10ms/step - loss: 7.7980e-05 - acc: 0.9999 - val_loss: 0.1539 - val_acc: 0.9719
Epoch 147/150
12825/12825 [==============================] - 125s 10ms/step - loss: 0.0155 - acc: 0.9980 - val_loss: 0.0180 - val_acc: 0.9979
Epoch 148/150
12825/12825 [==============================] - 125s 10ms/step - loss: 0.0070 - acc: 0.9982 - val_loss: 0.0043 - val_acc: 0.9986
Epoch 149/150
12825/12825 [==============================] - 126s 10ms/step - loss: 3.3416e-04 - acc: 0.9998 - val_loss: 1.1748e-06 - val_acc: 1.0000
Epoch 150/150
12825/12825 [==============================] - 126s 10ms/step - loss: 8.8318e-05 - acc: 1.0000 - val_loss: 3.7501e-07 - val_acc: 1.0000
750/750 [==============================] - 3s 3ms/step
test_loss: 3.6893165815854444e-07  , test_acc: 1.0
```

(b) Test data acc/loss

- 분석결과 그래프를 보면 train accuracy가 epoch 18 부터 대체적으로 1임을 볼 수 있다.
- Validation accuracy도 마찬가지로 epoch 40정도를 넘어가자 1의 값을 유지하고 있다.
- Test accuracy 또한 100%가 나왔으므로 accuracy 만을 기준으로 생각했을 때 최종적인 모델은 정확도가 100%인 모델이라고 볼 수 있다.
- 그러나, 적은 epoch에서 accuracy가 바로 1에 도달하는 것을 봤을 때, 과적합을 의심해볼 수도 있다.