ASL Alphabet Classification with CNN

```
from google.colab import drive
drive.mount('/content/gdrive')

     Mounted at /content/gdrive
```

```
import os
import cv2
import numpy as np
from time import time
from tensorflow.keras import utils
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D, BatchNorma
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib inline
```

```
!unzip gdrive/My\ Drive/asl_alphabet_train.zip
       inflating: asl_alphabet_train/A/A472.jpg
       inflating: asl_alphabet_train/A/A47.jpg
       inflating: asl_alphabet_train/A/A995.jpg
       inflating: asl_alphabet_train/A/A991.jpg
       inflating: asl_alphabet_train/A/A987.jpg
       inflating: asl_alphabet_train/A/A979.jpg
       inflating: asl_alphabet_train/A/A976.jpg
       inflating: asl_alphabet_train/A/A974.jpg
       inflating: asl_alphabet_train/A/A969.jpg
       inflating: asl_alphabet_train/A/A967.jpg
       inflating: asl_alphabet_train/A/A965.jpg
       inflating: asl_alphabet_train/A/A963.jpg
       inflating: asl_alphabet_train/A/A961.jpg
       inflating: asl_alphabet_train/A/A958.jpg
       inflating: asl_alphabet_train/A/A957.jpg
       inflating: asl_alphabet_train/A/A952.jpg
       inflating: asl_alphabet_train/A/A945.jpg
       inflating: asl_alphabet_train/A/A938.jpg
       inflating: asl_alphabet_train/A/A932.jpg
       inflating: asl_alphabet_train/A/A926.jpg
       inflating: asl_alphabet_train/A/A922.jpg
       inflating: asl_alphabet_train/A/A918.jpg
       inflating: asl_alphabet_train/A/A914.jpg
       inflating: asl_alphabet_train/A/A913.jpg
       inflating: asl_alphabet_train/A/A905.jpg
       inflating: asl_alphabet_train/A/A904.jpg
       inflating: asl_alphabet_train/A/A901.jpg
       inflating: asl_alphabet_train/A/A896.jpg
       inflating: asl_alphabet_train/A/A895.jpg
```

```
   inflating: asl_alphabet_train/A/A895.jpg
   inflating: asl_alphabet_train/A/A894.jpg
   inflating: asl_alphabet_train/A/A892.jpg
   inflating: asl_alphabet_train/A/A891.jpg
   inflating: asl_alphabet_train/A/A888.jpg
   inflating: asl_alphabet_train/A/A887.jpg
   inflating: asl_alphabet_train/A/A882.jpg
   inflating: asl_alphabet_train/A/A875.jpg
   inflating: asl_alphabet_train/A/A874.jpg
   inflating: asl_alphabet_train/A/A873.jpg
   inflating: asl_alphabet_train/A/A870.jpg
   inflating: asl_alphabet_train/A/A867.jpg
   inflating: asl_alphabet_train/A/A863.jpg
   inflating: asl_alphabet_train/A/A858.jpg
   inflating: asl_alphabet_train/A/A854.jpg
   inflating: asl_alphabet_train/A/A843.jpg
   inflating: asl_alphabet_train/A/A842.jpg
   inflating: asl_alphabet_train/A/A840.jpg
   inflating: asl_alphabet_train/A/A84.jpg
   inflating: asl_alphabet_train/A/A838.jpg
   inflating: asl_alphabet_train/A/A837.jpg
   inflating: asl_alphabet_train/A/A831.jpg
   inflating: asl_alphabet_train/A/A829.jpg
   inflating: asl_alphabet_train/A/A828.jpg
   inflating: asl_alphabet_train/A/A827.jpg
   inflating: asl_alphabet_train/A/A825.jpg
   inflating: asl_alphabet_train/A/A824.jpg
   inflating: asl_alphabet_train/A/A822.jpg
   inflating: asl_alphabet_train/A/A821.jpg
   inflating: asl_alphabet_train/A/A820.jpg
```

The data

```
test_dir = '/content/gdrive/My Drive/asl_alphabet_test'
```

```
os.listdir(test_dir)
```

```
['nothing_test.jpg',
 'Y_test.jpg',
 'space_test.jpg',
 'Z_test.jpg',
 'X_test.jpg',
 'W_test.jpg',
 'V_test.jpg',
 'U_test.jpg',
 'T_test.jpg',
 'S_test.jpg',
 'R_test.jpg',
 'Q_test.jpg',
 'P_test.jpg',
 'O_test.jpg',
 'N_test.jpg',
      'M_test.jpg',
      'L_test.jpg',
      'K_test.jpg',
      'J_test.jpg',
      'I_test.jpg',
      'H_test.jpg',
      'G_test.jpg',
      'F_test.jpg',
      'E_test.jpg',
      'D_test.jpg',
      'C_test.jpg',
      'B_test.jpg',
      'A_test.jpg']
```

```
train_dir = '/content/asl_alphabet_train'
test_dir = '/content/gdrive/My Drive/asl_alphabet_test'
classes = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',
           'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V',
           'W', 'X', 'Y', 'Z', 'nothing', 'space', 'del']
plt.figure(figsize=(11, 11))
for i in range (0,29):
    plt.subplot(7,7,i+1)
    plt.xticks([])
    plt.yticks([])
    path = train_dir + "/{0}/{0}1.jpg".format(classes[i])
    img = plt.imread(path)
    plt.imshow(img)
    plt.xlabel(classes[i])
```

Loading

```
def load_data(train_dir):
    images = []
    labels = []
    size = 32,32
    index = -1
    for folder in os.listdir(train_dir):
        index +=1
        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)
            labels.append(index)

    images = np.array(images)
    images = images.astype('float32')/255.0
    labels = utils.to_categorical(labels)
    x_train, x_test, y_train, y_test = train_test_split(images, labels, test_size = 0.1)

    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

start = time()
x_train, x_test, y_train, y_test = load_data(train_dir)
print('Loading:', time() - start)
```

```
     Loaded 78300 images for training, Train data shape = (78300, 32, 32, 3)
     Loaded 8700 images for testing Test data shape = (8700, 32, 32, 3)
     Loading: 70.09035587310791
```

Network

```
classes = 29
batch = 64
epochs = 5
learning_rate = 0.001

def results(model):
  adam = Adam(lr=learning_rate)

  model.compile(optimizer=adam, loss='categorical_crossentropy', metrics=['accuracy'])
  start = time()
  history = model.fit(x_train, y_train, batch_size=batch, epochs=epochs, validation_split=0.1
  train_time = time() - start

  model.summary()

  plt.figure(figsize=(12, 12))
  plt.subplot(3, 2, 1)
  plt.plot(history.history['accuracy'], label = 'train_accuracy')
  plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
  plt.xlabel('epoch')
  plt.ylabel('accuracy')
  plt.legend()
  plt.subplot(3, 2, 2)
  plt.plot(history.history['loss'], label = 'train_loss')
  plt.plot(history.history['val_loss'], label = 'val_loss')
  plt.xlabel('epoch')
  plt.ylabel('accuracy')
  plt.legend()
  plt.show()

  start = time()
  test_loss, test_acc = model.evaluate(x_test, y_test)
  test_time = time() - start
  print('\nTrain time: ', train_time)
  print('Test accuracy:', test_acc)
  print('Test loss:', test_loss)
  print('Test time: ', test_time)
```

Configuration

```
model = Sequential()

model.add(Conv2D(64, (3, 3), padding='same', input_shape=(32, 32, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3), padding='same', input_shape=(32, 32, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3), padding='same', input_shape=(32, 32, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(BatchNormalization())

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(1024, activation='sigmoid'))
model.add(Dense(classes, activation='softmax'))
```

```
results(model)
```

```
/usr/local/lib/python3.7/dist-packages/keras/optimizer_v2/adam.py:105: UserWarning: The
  super(Adam, self).__init__(name, **kwargs)
Epoch 1/5
1102/1102 [==============================] - 526s 475ms/step - loss: 0.5877 - accuracy:
Epoch 2/5
1102/1102 [==============================] - 522s 473ms/step - loss: 0.0728 - accuracy:
Epoch 3/5
1102/1102 [==============================] - 525s 477ms/step - loss: 0.0422 - accuracy:
Epoch 4/5
1102/1102 [==============================] - 522s 474ms/step - loss: 0.0321 - accuracy:
Epoch 5/5
1102/1102 [==============================] - 522s 473ms/step - loss: 0.0303 - accuracy:
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
```

```
model.save('best_model_data.h5')
```

```
 conv2d_1 (Conv2D)           (None, 16, 16, 128)       73856

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 128)        0
 2D)

 conv2d_2 (Conv2D)           (None, 8, 8, 256)         295168

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 256)        0
 2D)

 batch_normalization (BatchN  (None, 4, 4, 256)        1024
 ormalization)

 flatten (Flatten)           (None, 4096)              0

 dropout (Dropout)           (None, 4096)              0

 dense (Dense)               (None, 1024)              4195328

 dense_1 (Dense)             (None, 29)                29725

=================================================================
Total params: 4,596,893
Trainable params: 4,596,381
Non-trainable params: 512
```

✓ 0s completed at 22:04