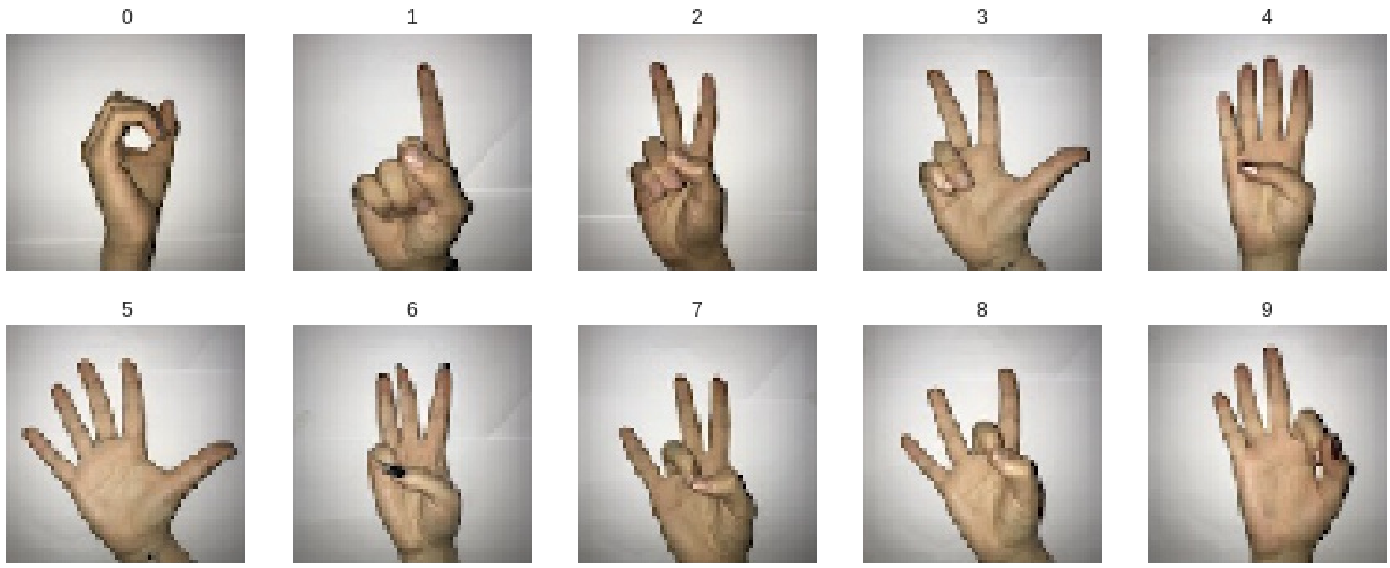
In [ ]:

```
import os, cv2, math
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Flatten
from keras.layers import Dense
from keras.layers import Dropout
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import plot_model
import numpy as np
from keras.preprocessing import image
from sklearn.model_selection import train_test_split
from shutil import copyfile
from tqdm import tqdm
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
```

In [ ]:

```
nrow, ncol = 2, 5
plt.rcParams['figure.figsize'] = (ncol*3, nrow*3)
for row in range(nrow):
    for col in range(ncol):
        img_index = row*ncol+col
        # load image
        img = image.load_img('Sign-Language-Digits-Dataset/Examples/example_' + str(img_
index) + '.JPG', target_size = (64, 64))
        plt.subplot(nrow, ncol, img_index + 1)
        plt.imshow(img)
        plt.title(img_index)
        plt.axis('off')
```



This project aims to create a classifier that can intepret sign language for number 0 to 9. The image of the sign language for respective number is shown above.

Inside 'Sign-Language-Digits-Dataset/Dataset/', the images of different sign language are organised according to their labels(i.e. 0 to 9). The dataset will first be divided into 3 sets : training_set, validation_set, and test_set. The size of training set, validation set, and test set are 70%, 15% and 15% of the whole dataset respectively.

To accomodate for the requirement for flow_from_directory method from keras, I will reorganise the images in the following structure :

- **Training set : 'Sign-Language-Digits-Dataset/Dataset/training_set/class_00/image_file'**
- **Validation set : 'Sign-Language-Digits-Dataset/Dataset/validation_set/class_00/image_file'**

- **Test set : 'Sign-Language-Digits-Dataset/Dataset/test_set/class_00/image_file'**

In [ ]:

```
DATASET_PATH = 'Sign-Language-Digits-Dataset/Dataset/'
```

In [ ]:

```
# Creating a list of filename for training set, validation set, and test set
train_set = {}
validation_set = {}
test_set = {}
for cat in os.listdir(DATASET_PATH):
  cat_dir = os.path.join(DATASET_PATH, cat) # e.g. DATASET_PATH/'0'
  cat_files = os.listdir(cat_dir)
  # Training set's size is 70% of the data
  train_list , test_list = train_test_split(cat_files, test_size = 0.3)
  # Validation set's and Test set's size are both 15% of the data
  validation_list, test_list = train_test_split(test_list, test_size = 0.5)
  train_set[cat] = train_list
  validation_set[cat] = validation_list
  test_set[cat] = test_list
```

In [ ]:

```
for cat in tqdm(train_set.keys()):
  cat_dir = os.path.join(DATASET_PATH, 'training_set', 'class_0' + str(cat))
  os.makedirs(cat_dir)
  for file in train_set[cat]:
    # src path is DATASET_PATH/'0'/file
    src = os.path.join(DATASET_PATH, cat, file)
    # dest path is DATASET_PATH/'training_set'/'class_00'
    # to accomodate for the directory format required by flow_from_directory method in ke
ras
    dest = os.path.join(cat_dir, file)
    copyfile(src, dest)
```

```
100%|██████████| 10/10 [00:00<00:00, 45.75it/s]
```

In [ ]:

```
for cat in tqdm(validation_set.keys()):
  cat_dir = os.path.join(DATASET_PATH, 'validation_set', 'class_0' + str(cat))
  os.makedirs(cat_dir)
  for file in validation_set[cat]:
    # src path is DATASET_PATH/'0'/file
    src = os.path.join(DATASET_PATH, cat, file)
    # dest path is DATASET_PATH/'validation_set'/'class_00'
    # to accomodate for the directory format required by flow_from_directory method in ke
ras
    dest = os.path.join(cat_dir, file)
    copyfile(src, dest)
```

```
100%|██████████| 10/10 [00:00<00:00, 210.44it/s]
```

In [ ]:

```
for cat in tqdm(test_set.keys()):
  cat_dir = os.path.join(DATASET_PATH, 'test_set', 'class_0' + str(cat))
  os.makedirs(cat_dir)
  for file in test_set[cat]:
    # src path is DATASET_PATH/'0'/file
    src = os.path.join(DATASET_PATH, cat, file)
    # dest path is DATASET_PATH/'test_set'/'class_00'
    # to accomodate for the directory format required by flow_from_directory method in ke
ras
    dest = os.path.join(cat_dir, file)
    copyfile(src, dest)
```

```
100%|██████████| 10/10 [00:00<00:00, 208.02it/s]
```

In [ ]:

```
for i in range(10):
  train_size = len(train_set[str(i)])
  validation_size = len(validation_set[str(i)])
  test_size = len(test_set[str(i)])
  print("0{} : Training size({}) Validation size({}) Test size({})".format(i, train_size
, validation_size, test_size))
```

```
00 : Training size(143) Validation size(31) Test size(31)
01 : Training size(144) Validation size(31) Test size(31)
02 : Training size(144) Validation size(31) Test size(31)
03 : Training size(144) Validation size(31) Test size(31)
04 : Training size(144) Validation size(31) Test size(32)
05 : Training size(144) Validation size(31) Test size(32)
06 : Training size(144) Validation size(31) Test size(32)
07 : Training size(144) Validation size(31) Test size(31)
08 : Training size(145) Validation size(31) Test size(32)
09 : Training size(142) Validation size(31) Test size(31)
```

Data augmentation is performed on the training set images so that the classifier can learn any changes with respect to scaling, horizontal_flip, or others.

In [ ]:

```
# Performing data augmentation on training dataset
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

# For validation dataset, only rescale the pictures
validation_datagen = ImageDataGenerator(rescale = 1./255)

# For test dataset, only rescale the pictures
test_datagen = ImageDataGenerator(rescale = 1./255)

training_data = train_datagen.flow_from_directory(os.path.join(DATASET_PATH, 'training_se
t'),
                                                  target_size = (64, 64),
                                                  batch_size = 32,
                                                  class_mode = 'categorical')

validation_data = validation_datagen.flow_from_directory(os.path.join(DATASET_PATH, 'vali
dation_set'),
                                                  target_size = (64, 64),
                                                  batch_size = 32,
                                                  class_mode = 'categorical')

test_data = test_datagen.flow_from_directory(os.path.join(DATASET_PATH, 'test_set'),
                                             target_size = (64, 64),
                                             batch_size = 32,
                                             class_mode = 'categorical')
```

```
Found 1438 images belonging to 10 classes.
Found 310 images belonging to 10 classes.
Found 314 images belonging to 10 classes.
```

In [ ]:

```
# Initialising the CNN
classifier = Sequential()

# Adding first convolutional layer, followed by pooling, and dropout
classifier.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Dropout(0.25))

# Adding second convolutional layer, followed by pooling, and dropout
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
```

```
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Dropout(0.25))

# Adding third convolutional layer, followed by pooling, and dropout
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))
classifier.add(Dropout(0.25))

# Flattening
classifier.add(Flatten())

# Full connection
classifier.add(Dense(units = 128, activation = 'relu'))
classifier.add(Dense(units = 10, activation = 'softmax'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['ac
curacy'])
```

In [ ]:

```
# Train the data with training set, and check the result with validation accuracy
history = classifier.fit_generator(training_data,
                         steps_per_epoch = math.ceil(training_data.n / training_data.bat
ch_size),
                         epochs = 100,
                         validation_data = validation_data,
                         validation_steps = math.ceil(validation_data.n / validation_dat
a.batch_size))
```

```
Epoch 1/100
45/45 [==============================] - 4s 99ms/step - loss: 2.3166 - acc: 0.1078 - val_
loss: 2.2999 - val_acc: 0.2000
Epoch 2/100
45/45 [==============================] - 4s 89ms/step - loss: 2.2987 - acc: 0.1100 - val_
loss: 2.2814 - val_acc: 0.2032
Epoch 3/100
45/45 [==============================] - 4s 86ms/step - loss: 2.1714 - acc: 0.2350 - val_
loss: 1.7885 - val_acc: 0.4677
Epoch 4/100
45/45 [==============================] - 4s 84ms/step - loss: 1.6709 - acc: 0.4214 - val_
loss: 1.3537 - val_acc: 0.5968
Epoch 5/100
45/45 [==============================] - 4s 86ms/step - loss: 1.3323 - acc: 0.5522 - val_
loss: 1.1175 - val_acc: 0.6548
Epoch 6/100
38/45 [========================>.....] - ETA: 0s - loss: 1.1436 - acc: 0.605545/45 [=====
=========================] - 4s 88ms/step - loss: 1.1228 - acc: 0.6079 - val_loss: 0.7856
- val_acc: 0.7548
Epoch 7/100
45/45 [==============================] - 4s 83ms/step - loss: 0.9682 - acc: 0.6934 - val_
loss: 0.7076 - val_acc: 0.7645
Epoch 8/100
45/45 [==============================] - 4s 88ms/step - loss: 0.8472 - acc: 0.7121 - val_
loss: 0.6002 - val_acc: 0.8161
Epoch 9/100
45/45 [==============================] - 4s 84ms/step - loss: 0.7240 - acc: 0.7608 - val_
loss: 0.5330 - val_acc: 0.8323
Epoch 10/100
45/45 [==============================] - 4s 89ms/step - loss: 0.6620 - acc: 0.7824 - val_
loss: 0.4652 - val_acc: 0.8387
Epoch 11/100
45/45 [==============================] - 4s 87ms/step - loss: 0.5775 - acc: 0.8130 - val_
loss: 0.4387 - val_acc: 0.8581
Epoch 12/100
 1/45 [..............................] - ETA: 1s - loss: 0.6897 - acc: 0.750045/45 [=====
=========================] - 4s 85ms/step - loss: 0.5160 - acc: 0.8338 - val_loss: 0.3950
- val_acc: 0.8742
Epoch 13/100
45/45 [==============================] - 4s 88ms/step - loss: 0.4625 - acc: 0.8519 - val_
loss: 0.3751 - val_acc: 0.8710
Epoch 14/100
```

```
45/45 [==============================] - 4s 87ms/step - loss: 0.4323 - acc: 0.8566 - val_
loss: 0.3411 - val_acc: 0.8871
Epoch 15/100
45/45 [==============================] - 4s 87ms/step - loss: 0.3560 - acc: 0.8789 - val_
loss: 0.3274 - val_acc: 0.8968
Epoch 16/100
45/45 [==============================] - 4s 86ms/step - loss: 0.3471 - acc: 0.8861 - val_
loss: 0.2720 - val_acc: 0.9097
Epoch 17/100
44/45 [============================>.] - ETA: 0s - loss: 0.3077 - acc: 0.889245/45 [=====
=========================] - 4s 84ms/step - loss: 0.3105 - acc: 0.8875 - val_loss: 0.2789
- val_acc: 0.9194
Epoch 18/100
45/45 [==============================] - 4s 88ms/step - loss: 0.3541 - acc: 0.8969 - val_
loss: 0.2637 - val_acc: 0.9129
Epoch 19/100
45/45 [==============================] - 4s 86ms/step - loss: 0.2914 - acc: 0.9062 - val_
loss: 0.2287 - val_acc: 0.9194
Epoch 20/100
45/45 [==============================] - 4s 85ms/step - loss: 0.2834 - acc: 0.9012 - val_
loss: 0.2743 - val_acc: 0.9065
Epoch 21/100
45/45 [==============================] - 4s 88ms/step - loss: 0.2999 - acc: 0.9033 - val_
loss: 0.3163 - val_acc: 0.9032
Epoch 22/100
45/45 [==============================] - 4s 88ms/step - loss: 0.2368 - acc: 0.9263 - val_
loss: 0.2428 - val_acc: 0.9194
Epoch 23/100
45/45 [==============================] - 4s 86ms/step - loss: 0.2326 - acc: 0.9291 - val_
loss: 0.2146 - val_acc: 0.9355
Epoch 24/100
45/45 [==============================] - 4s 87ms/step - loss: 0.2355 - acc: 0.9200 - val_
loss: 0.2397 - val_acc: 0.9226
Epoch 25/100
45/45 [==============================] - 4s 85ms/step - loss: 0.2090 - acc: 0.9285 - val_
loss: 0.1783 - val_acc: 0.9387
Epoch 26/100
45/45 [==============================] - 4s 88ms/step - loss: 0.2007 - acc: 0.9257 - val_
loss: 0.1967 - val_acc: 0.9290
Epoch 27/100
45/45 [==============================] - 4s 87ms/step - loss: 0.2032 - acc: 0.9270 - val_
loss: 0.2683 - val_acc: 0.9065
Epoch 28/100
35/45 [======================>.......] - ETA: 0s - loss: 0.1885 - acc: 0.931345/45 [=====
=========================] - 4s 87ms/step - loss: 0.1918 - acc: 0.9361 - val_loss: 0.1989
- val_acc: 0.9452
Epoch 29/100
45/45 [==============================] - 4s 86ms/step - loss: 0.1742 - acc: 0.9417 - val_
loss: 0.2098 - val_acc: 0.9290
Epoch 30/100
45/45 [==============================] - 4s 83ms/step - loss: 0.1646 - acc: 0.9422 - val_
loss: 0.1798 - val_acc: 0.9484
Epoch 31/100
45/45 [==============================] - 4s 86ms/step - loss: 0.1541 - acc: 0.9479 - val_
loss: 0.2427 - val_acc: 0.9226
Epoch 32/100
45/45 [==============================] - 4s 88ms/step - loss: 0.1681 - acc: 0.9470 - val_
loss: 0.2270 - val_acc: 0.9387
Epoch 33/100
45/45 [==============================] - 4s 86ms/step - loss: 0.1512 - acc: 0.9463 - val_
loss: 0.2692 - val_acc: 0.9226
Epoch 34/100
45/45 [==============================] - 4s 86ms/step - loss: 0.1594 - acc: 0.9478 - val_
loss: 0.2032 - val_acc: 0.9419
Epoch 35/100
45/45 [==============================] - 4s 86ms/step - loss: 0.1471 - acc: 0.9513 - val_
loss: 0.1822 - val_acc: 0.9516
Epoch 36/100
45/45 [==============================] - 4s 86ms/step - loss: 0.1314 - acc: 0.9527 - val_
loss: 0.1751 - val_acc: 0.9452
Epoch 37/100
45/45 [==============================] - 4s 86ms/step - loss: 0.1201 - acc: 0.9597 - val_
```

```
loss: 0.1665 - val_acc: 0.9484
Epoch 38/100
45/45 [==============================] - 4s 86ms/step - loss: 0.1155 - acc: 0.9644 - val_
loss: 0.1777 - val_acc: 0.9548
Epoch 39/100
41/45 [==========================>...] - ETA: 0s - loss: 0.1349 - acc: 0.958845/45 [=====
=========================] - 4s 86ms/step - loss: 0.1319 - acc: 0.9590 - val_loss: 0.1698
- val_acc: 0.9710
Epoch 40/100
45/45 [==============================] - 4s 87ms/step - loss: 0.1395 - acc: 0.9548 - val_
loss: 0.1491 - val_acc: 0.9581
Epoch 41/100
45/45 [==============================] - 4s 85ms/step - loss: 0.1270 - acc: 0.9618 - val_
loss: 0.1482 - val_acc: 0.9645
Epoch 42/100
45/45 [==============================] - 4s 88ms/step - loss: 0.1096 - acc: 0.9666 - val_
loss: 0.2051 - val_acc: 0.9290
Epoch 43/100
45/45 [==============================] - 4s 87ms/step - loss: 0.1121 - acc: 0.9639 - val_
loss: 0.1704 - val_acc: 0.9516
Epoch 44/100
45/45 [==============================] - 4s 87ms/step - loss: 0.1186 - acc: 0.9582 - val_
loss: 0.2136 - val_acc: 0.9323
Epoch 45/100
 4/45 [=>............................] - ETA: 0s - loss: 0.1298 - acc: 0.945345/45 [=====
=========================] - 4s 85ms/step - loss: 0.0954 - acc: 0.9681 - val_loss: 0.2170
- val_acc: 0.9387
Epoch 46/100
45/45 [==============================] - 4s 89ms/step - loss: 0.0926 - acc: 0.9715 - val_
loss: 0.1994 - val_acc: 0.9355
Epoch 47/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0916 - acc: 0.9638 - val_
loss: 0.2190 - val_acc: 0.9484
Epoch 48/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0962 - acc: 0.9645 - val_
loss: 0.1633 - val_acc: 0.9548
Epoch 49/100
45/45 [==============================] - 4s 87ms/step - loss: 0.1014 - acc: 0.9680 - val_
loss: 0.1364 - val_acc: 0.9613
Epoch 50/100
42/45 [===========================>..] - ETA: 0s - loss: 0.0655 - acc: 0.981245/45 [=====
=========================] - 4s 82ms/step - loss: 0.0660 - acc: 0.9797 - val_loss: 0.1739
- val_acc: 0.9419
Epoch 51/100
45/45 [==============================] - 4s 88ms/step - loss: 0.0969 - acc: 0.9645 - val_
loss: 0.1538 - val_acc: 0.9774
Epoch 52/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0785 - acc: 0.9757 - val_
loss: 0.2437 - val_acc: 0.9323
Epoch 53/100
45/45 [==============================] - 4s 84ms/step - loss: 0.0973 - acc: 0.9681 - val_
loss: 0.2176 - val_acc: 0.9484
Epoch 54/100
45/45 [==============================] - 4s 89ms/step - loss: 0.0871 - acc: 0.9679 - val_
loss: 0.2447 - val_acc: 0.9258
Epoch 55/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0650 - acc: 0.9784 - val_
loss: 0.1845 - val_acc: 0.9677
Epoch 56/100
 1/45 [..............................] - ETA: 1s - loss: 0.0228 - acc: 1.000045/45 [=====
=========================] - 4s 86ms/step - loss: 0.0786 - acc: 0.9681 - val_loss: 0.1556
- val_acc: 0.9677
Epoch 57/100
45/45 [==============================] - 4s 87ms/step - loss: 0.0889 - acc: 0.9750 - val_
loss: 0.1540 - val_acc: 0.9645
Epoch 58/100
45/45 [==============================] - 4s 87ms/step - loss: 0.0808 - acc: 0.9756 - val_
loss: 0.2367 - val_acc: 0.9452
Epoch 59/100
45/45 [==============================] - 4s 84ms/step - loss: 0.0952 - acc: 0.9652 - val_
loss: 0.2098 - val_acc: 0.9516
Epoch 60/100
```

```
45/45 [==============================] - 4s 88ms/step - loss: 0.0735 - acc: 0.9743 - val_
loss: 0.1907 - val_acc: 0.9548
Epoch 61/100
43/45 [===========================>..] - ETA: 0s - loss: 0.0872 - acc: 0.969445/45 [=====
=========================] - 4s 87ms/step - loss: 0.0942 - acc: 0.9666 - val_loss: 0.2337
- val_acc: 0.9387
Epoch 62/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0816 - acc: 0.9750 - val_
loss: 0.1878 - val_acc: 0.9613
Epoch 63/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0804 - acc: 0.9729 - val_
loss: 0.2179 - val_acc: 0.9452
Epoch 64/100
45/45 [==============================] - 4s 87ms/step - loss: 0.0704 - acc: 0.9778 - val_
loss: 0.1301 - val_acc: 0.9742
Epoch 65/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0646 - acc: 0.9757 - val_
loss: 0.1568 - val_acc: 0.9613
Epoch 66/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0628 - acc: 0.9748 - val_
loss: 0.1773 - val_acc: 0.9710
Epoch 67/100
 6/45 [===>..........................] - ETA: 1s - loss: 0.0278 - acc: 0.989645/45 [=====
=========================] - 4s 87ms/step - loss: 0.0618 - acc: 0.9778 - val_loss: 0.1486
- val_acc: 0.9774
Epoch 68/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0745 - acc: 0.9791 - val_
loss: 0.1406 - val_acc: 0.9613
Epoch 69/100
45/45 [==============================] - 4s 85ms/step - loss: 0.0515 - acc: 0.9833 - val_
loss: 0.1386 - val_acc: 0.9677
Epoch 70/100
45/45 [==============================] - 4s 83ms/step - loss: 0.0406 - acc: 0.9861 - val_
loss: 0.2163 - val_acc: 0.9452
Epoch 71/100
45/45 [==============================] - 4s 88ms/step - loss: 0.0618 - acc: 0.9785 - val_
loss: 0.1571 - val_acc: 0.9645
Epoch 72/100
40/45 [=========================>....] - ETA: 0s - loss: 0.0628 - acc: 0.980545/45 [=====
=========================] - 4s 86ms/step - loss: 0.0594 - acc: 0.9812 - val_loss: 0.1517
- val_acc: 0.9742
Epoch 73/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0535 - acc: 0.9854 - val_
loss: 0.1564 - val_acc: 0.9645
Epoch 74/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0599 - acc: 0.9812 - val_
loss: 0.1113 - val_acc: 0.9677
Epoch 75/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0529 - acc: 0.9847 - val_
loss: 0.1691 - val_acc: 0.9710
Epoch 76/100
45/45 [==============================] - 4s 87ms/step - loss: 0.0651 - acc: 0.9799 - val_
loss: 0.1485 - val_acc: 0.9581
Epoch 77/100
45/45 [==============================] - 4s 87ms/step - loss: 0.0547 - acc: 0.9776 - val_
loss: 0.1324 - val_acc: 0.9742
Epoch 78/100
 6/45 [===>..........................] - ETA: 1s - loss: 0.0139 - acc: 0.994845/45 [=====
=========================] - 4s 87ms/step - loss: 0.0663 - acc: 0.9764 - val_loss: 0.1138
- val_acc: 0.9613
Epoch 79/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0481 - acc: 0.9806 - val_
loss: 0.1371 - val_acc: 0.9774
Epoch 80/100
45/45 [==============================] - 4s 88ms/step - loss: 0.0517 - acc: 0.9826 - val_
loss: 0.1531 - val_acc: 0.9645
Epoch 81/100
45/45 [==============================] - 4s 87ms/step - loss: 0.0506 - acc: 0.9833 - val_
loss: 0.1479 - val_acc: 0.9484
Epoch 82/100
45/45 [==============================] - 4s 84ms/step - loss: 0.0619 - acc: 0.9791 - val_
loss: 0.1519 - val_acc: 0.9742
```

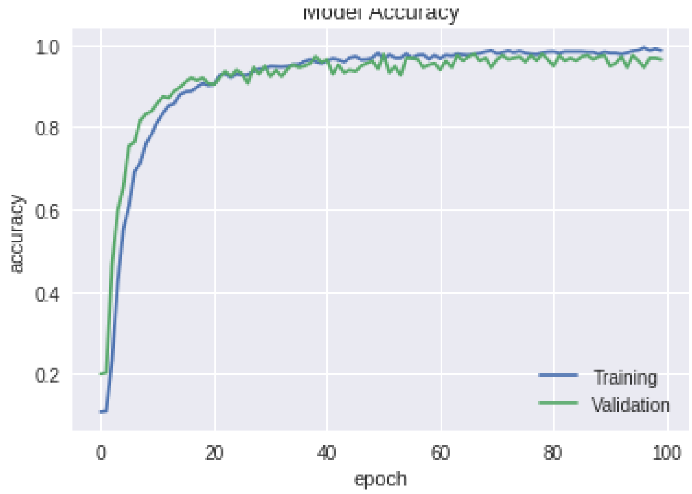
```
Epoch 83/100
43/45 [===========================>..] - ETA: 0s - loss: 0.0466 - acc: 0.984745/45 [=====
=========================] - 4s 88ms/step - loss: 0.0481 - acc: 0.9833 - val_loss: 0.1572
- val_acc: 0.9613
Epoch 84/100
45/45 [==============================] - 4s 89ms/step - loss: 0.0494 - acc: 0.9833 - val_
loss: 0.1799 - val_acc: 0.9677
Epoch 85/100
45/45 [==============================] - 4s 91ms/step - loss: 0.0601 - acc: 0.9833 - val_
loss: 0.1530 - val_acc: 0.9613
Epoch 86/100
45/45 [==============================] - 4s 90ms/step - loss: 0.0486 - acc: 0.9833 - val_
loss: 0.1274 - val_acc: 0.9710
Epoch 87/100
45/45 [==============================] - 4s 85ms/step - loss: 0.0499 - acc: 0.9819 - val_
loss: 0.1294 - val_acc: 0.9742
Epoch 88/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0580 - acc: 0.9819 - val_
loss: 0.1713 - val_acc: 0.9645
Epoch 89/100
 4/45 [=>............................] - ETA: 0s - loss: 0.0368 - acc: 0.976645/45 [=====
=========================] - 4s 86ms/step - loss: 0.0572 - acc: 0.9771 - val_loss: 0.1225
- val_acc: 0.9710
Epoch 90/100
45/45 [==============================] - 4s 87ms/step - loss: 0.0531 - acc: 0.9826 - val_
loss: 0.1285 - val_acc: 0.9742
Epoch 91/100
45/45 [==============================] - 4s 87ms/step - loss: 0.0631 - acc: 0.9798 - val_
loss: 0.1770 - val_acc: 0.9484
Epoch 92/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0613 - acc: 0.9799 - val_
loss: 0.1877 - val_acc: 0.9548
Epoch 93/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0549 - acc: 0.9771 - val_
loss: 0.1339 - val_acc: 0.9710
Epoch 94/100
43/45 [===========================>..] - ETA: 0s - loss: 0.0565 - acc: 0.979745/45 [=====
=========================] - 4s 86ms/step - loss: 0.0546 - acc: 0.9806 - val_loss: 0.1603
- val_acc: 0.9452
Epoch 95/100
45/45 [==============================] - 4s 84ms/step - loss: 0.0472 - acc: 0.9847 - val_
loss: 0.1371 - val_acc: 0.9742
Epoch 96/100
45/45 [==============================] - 4s 88ms/step - loss: 0.0417 - acc: 0.9868 - val_
loss: 0.1387 - val_acc: 0.9613
Epoch 97/100
45/45 [==============================] - 4s 86ms/step - loss: 0.0265 - acc: 0.9931 - val_
loss: 0.2287 - val_acc: 0.9452
Epoch 98/100
45/45 [==============================] - 4s 84ms/step - loss: 0.0395 - acc: 0.9861 - val_
loss: 0.1480 - val_acc: 0.9677
Epoch 99/100
45/45 [==============================] - 4s 90ms/step - loss: 0.0378 - acc: 0.9903 - val_
loss: 0.1255 - val_acc: 0.9677
Epoch 100/100
 1/45 [..............................] - ETA: 1s - loss: 0.0529 - acc: 0.968845/45 [=====
=========================] - 4s 85ms/step - loss: 0.0398 - acc: 0.9861 - val_loss: 0.1645
- val_acc: 0.9645
```

In [ ]:

```
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Training', 'Validation'])
```

Out[ ]:

```
<matplotlib.legend.Legend at 0x7f0bb8f88748>
```



From the accuracy plot, the validation accuracy differs from the training accuracy by a small extent indicating absence of overfitting. When the classifier is evaluated on the test set, it obtains a relatively high accuracy.

In [ ]:

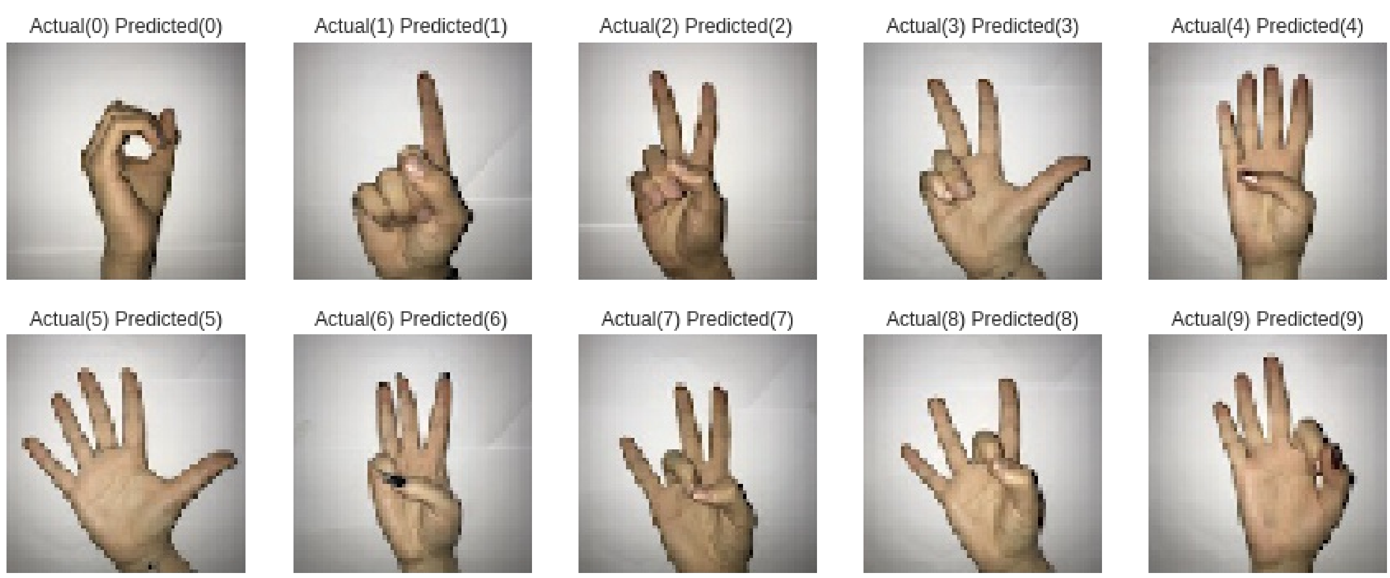
```
# Accuracy of the classifier when evaluated based on the test set
test_loss, test_accuracy = classifier.evaluate_generator(test_data, math.ceil(test_data.n
/ test_data.batch_size))
print("Accuracy on test set : {}".format(test_accuracy))
```

```
Accuracy on test set : 0.974522290715746
```

To visualise the performance of the classifier, the classifier will be used to predict all the example images for 0 to 9.

In [ ]:

```
nrow, ncol = 2, 5
plt.rcParams['figure.figsize'] = (ncol*3, nrow*3)
for row in range(nrow):
    for col in range(ncol):
        img_index = row*ncol+col
        # load image
        img = image.load_img('Sign-Language-Digits-Dataset/Examples/example_' + str(img_
index) + '.JPG', target_size = (64, 64))
        # convert image into array for prediction
        test_image = image.img_to_array(img)
        test_image = np.expand_dims(test_image, axis = 0)
        # predict image using classifier
        result = classifier.predict(test_image).argmax()
        plt.subplot(nrow, ncol, img_index + 1)
        plt.imshow(img)
        plt.title("Actual({}) Predicted({})".format(img_index, result))
        plt.axis('off')
```



It can be seen from above output that the classifier is able to classify all the images correctly. To save the model for future use, simply run the code below.

In [ ]:

```
# save the models and weight for future purposes
# serialize model to JSON
model_json = classifier.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
classifier.save_weights("model.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

Reference

- https://github.com/ardamavi/Sign-Language-Digits-Dataset
- https://www.superdatascience.com/deep-learning/