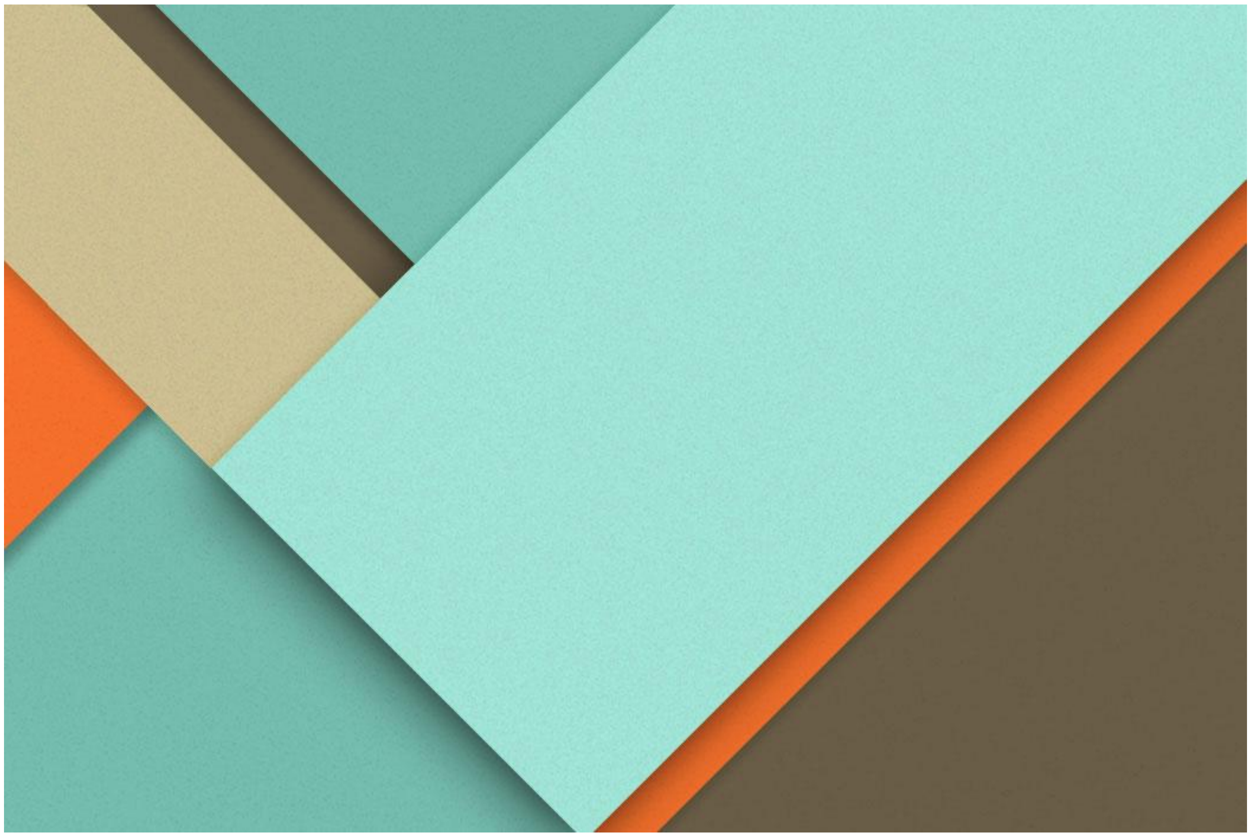


Arabic Hand Gesture Recognition

30.11.2020

—

Your Name
Your Company
123 Your Street
Your City, ST 12345

Introduction

The project mainly belongs to deep learning(subdomain of machine learning) involving neural networks to develop a predictive model trained on the collected dataset. The domain of this project is related to sign language used by the deaf and the mute people. The system will use the visual hand dataset based on an Arabic Sign Language and interpret this visual data in textual information. While running the system, each meaningful text is translated using its corresponding hand gesture as an input. So, before giving an insight of the dataset and the deep learning methods used, it is useful to talk about the problem statement described in the succeeding section.

Problem statement

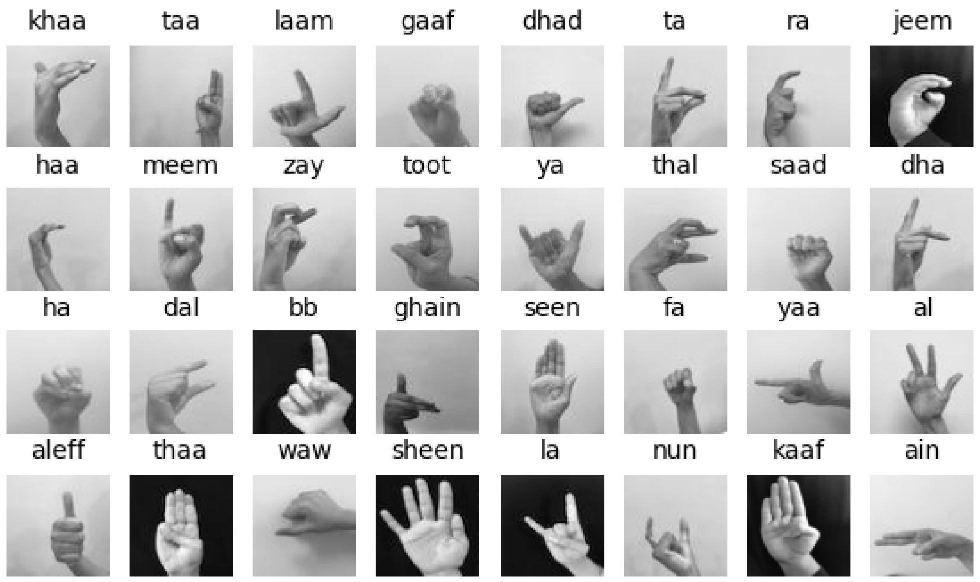
Sign language is very important for the deaf and the mute people to communicate both with the normal people and with themselves. We as normal people tend to ignore the importance of sign language which is the mere source of communication for the deaf and the mute communities. These people are facing major downfalls in their lives because of these disabilities or impairments leading to their unemployment, severe depression, and several other symptoms. One of the services they are using for communication or for us to talk to them are the sign language interpreters. But, hiring these interpreters is very costly and therefore, a cheap solution is required for resolving this big problem of these massive unfortunate people. A thorough analysis and survey to find a way to make these disabled communicable with themselves and the normal people have led to the breakthrough of the Sign Language Recognition System. The system targets to recognize the sign language and translate it into text for some meaningful communication.

Objective

Our aim is to develop a sign language recognition system capable enough to translate the most commonly expressed hand gestures used by deaf or a dumb person into textual data. To make these disabled people communicable is our prime objective.

Dataset description

The provided dataset consists of 54049 images of American sign language alphabets performed by more than 40 people for 32 standard Arabic signs and alphabets. The number of images per class differs from one class to another. Each distinct hand gesture indicates some meaningful information. Estimatedly, there are around 1500 images per class and each class represents a different meaning by its hand gesture or sign. Pictorially, the sample image of each class along with its label is represented in the figure below.



For some storage schemes, 32 folders are created and each folder consists of around 1500 images incorporating differently aged people's hand gestures in different environments. The directories containing these folders are treated as training and validation datasets for the model which will be explained later in this section.

Now, before talking about the model used, it is mandatory to undergo data preprocessing to make the dataset more consistent and compatible to the model as an input. So, how the data preprocessing is done is elaborated in the next section.

Data preprocessing

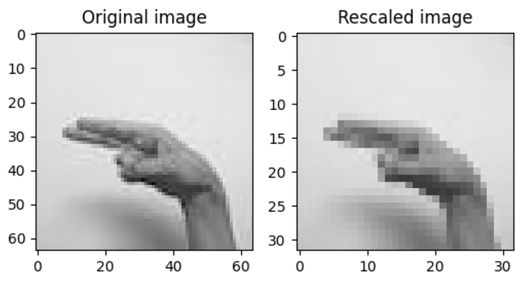
The data preprocessing involves the transformation applied to the data before feeding it to the model for training/testing. So, what changes are performed on the dataset is described below.

Removing data imbalancement

As already mentioned, the number of images per class differs from each other. This imbalancement among the classes may degrade the training performance of the model. Thus, to avoid this imbalancement, there must be an equal number of images among all classes. This imbalancement is removed by looping over each class folder to get the list of filenames of all the images per class. During each iteration, 1000 images are picked randomly from the current class folder and the rest are removed. Resultantly, a total 32000 images are filtered by summing up 1000 images of all the classes.

Data rescaling

The images contained in each class have the dimensions of (64x64). In order to keep the computations while training less complex and fast, the images can be rescaled into (32x32) following the same ratio of dimensionality. Rescaling is explained pictorially by the following figure.



Data augmentation

The data augmentation technique is widely used to increase the size of the training dataset by generating the artificial modified versions of the original images from the training dataset.

The technique results in a more diverse and consistent sequence of images and this may further result in creating more generalized and skillful deep learning models. The technique helps to avoid overfitting and underfitting of the model by applying several optional modifications to the training images. In this case, the following augmented changes are performed on the training images through ImageDataGenerator provided by the keras API.

Width shift range

This augmentation technique includes the horizontal shifting of the object to the left or right upto to the defined limit.

Height shift range

This steps includes the vertical shifting of the targeted object to up and down upto a certain limit.

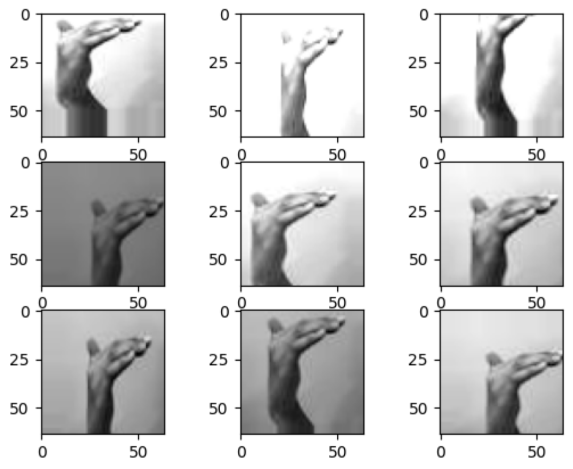
Brightness range

This augmentation technique involves the random darkening and brightening the images upto to a certain limit.

Zoom range

This augmentation technique randomly removes or add the pixels into the images for zoom in or zoom out upto the provided limitation.

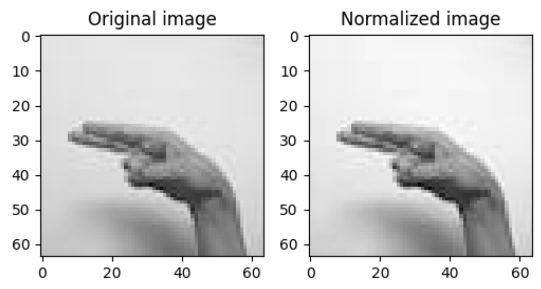
All the above-mentioned augmentation techniques are performed by passing parameters with their limitations to the ImageDataGenerator class provided by keras API. The transformations of the original image can be seen in the following figure.



Considering the preceding figure, it includes various augmented images generated from the one original image belonging to class 'khaa'. These images are then converted into normalized images and this normalization process is explained in the next section.

Data normalization

This step performs the normalization process on each image of the dataset. Usually, the pixel values in the image range from 0 to 255. But, these values must be rescaled before providing these images to the model as an input. So, the normalization will rescale these pixel values in the range of (0, 1). This rescaling will keep the model easy to learn and train in a fast way. So, this process is explained by the figure below.



Considering the above figure, there is some contrast difference between the two images. Normalized image is a bit more clear and bright than the original image. So, normalized images are more adaptable and easy to learn for the model to train.

Data splitting

The data used to build the model comes from multiple types of datasets. There are three different purposeful datasets for any computer vision project to analyze, make some comparisons and improve the performance of the model. In particular, these three different types of datasets are used in different stages of the creation of any machine learning model. These three distinct datasets are stated below:

Training dataset

The training set is a dataset on which the model is trained for learning weights or features. Initially, the model is fitted on the training dataset and in our case specifically, 80 percent of the whole dataset is used for the training dataset which is approximately 25600 images.

Validation dataset

The model is fitted on the validation dataset for the unbiased evaluation of itself during training. It validates the performance of the model based on how well the model is learning its weights before it is used for the real-time testing on the testing dataset. In our scenario of sign language recognition, 20 percent of the dataset is used which is equivalent to 6400 images.

Test dataset

After the completion of training and validation phenomena, the test dataset is used for testing the model and to measure the goodness of how well the model is trained. For this, 960 samples are used for the test dataset since there are 30 test images for each sign alphabet. This self-generated test set is created in order to measure the model's ability to generalize. More importantly, this test set is not collected from the 32000 images.

Model selection

After the dataset is fully preprocessed, it is then fed to the model network as a compatible input fashion for training. But before starting this time-consuming process, it is necessary to ensure the best possible selection of the deep neural network considering the problem domain. So, to ensure the best possible fit, there are several pretrained models available in the keras library. These pretrained models are trained at the ImageNet dataset to provide state of the art results in the domain of image classification. What is ImageNet dataset and what classes the ImageNet dataset constitutes is explained briefly in the next section

ImageNet dataset

ImageNet is a large collection of annotated images publicly made available for computer vision research. This large-scale collection of images is a critical resource for analyzing, training and testing the machine learning algorithms. There are around 14 million images and 1000 categories in this dataset and this dataset is also used for large-scale visual recognition challenge competitions. The pretrained models provided by the keras.applications python package are also called complex functional models because these functional models are trained on the ImageNet dataset. These pre-trained models are capable enough to classify any image that falls into these categories of images.

Keras pretrained models

As mentioned before briefly, keras applications constitute several pre-trained deep learning models available in its repository. The pretrained weights are also available alongside these models. So, these models are further used for custom object detection, image classification etc. But considering the domain problem, image classifiers are filtered from these pre-trained models and not the object detectors because the case requires performing the image classification. The selection is made considering the complexity and the nature of the dataset. The selected pretrained models with their results are mentioned in the below table.

Model	**Size(MB)**	**Top-1 acc**	**Top-5 acc**	**Parameters**	**Depth**
Xception	88	0.790	0.945	22,910,480	126
VGG16	528	0.713	0.901	138,357,544	23
ResNet50	98	0.749	0.921	25,636,712	-
InceptionV3	92	0.779	0.937	23,851,7841	159

MobileNet	16	0.704	0.895	4,253,864	88
EfficientNet	29	0.810	0.922	19,466,823	-

Considering the above table, it is important to know that all the models are trained at the ImageNet dataset. Every model has its own size, accuracy and number of parameters along with the architecture depth. These models are retrained further on the arabic hand gesture classification dataset comprising 32 classes. After training, the best possible fit is considered for the case. So, the final selection is made after custom-training these models on the given dataset. Finally, the custom training begins in the succeeding section.

Model training

This section includes the detailed analysis about how to perform the complex training process to produce state of the art results. To custom train the selected keras pretrained models, transfer learning is the only choice. So, what is transfer learning and how to perform it is explained in the below section.

Transfer learning

Transfer learning refers to the situation when the knowledge learned in one task or domain is reused to improve the generalization in another domain. From machine learning's perspective, it can be defined as reusing the saved weights of any pre-trained model to improve the accuracy or to custom-train your own model. In order to use the weights of any pre-trained model e.g. VGG16, EfficientNet, some modifications have to be made to make the model compatible to train on another dataset. The modifications performed on the neural network are elaborated in the next part.

Modifications in pretrained models

Basically, three modifications are made to make the model ready to train in the given case. Those modifications are briefly explained below.

Input layer modification

Input layer is changed considering the dimensions and size of the input images. In the current case, the images are of the size (64, 64) and the ImageDataGenerator class receives the input shape parameter to automatically prepare the input layer of the model to initialize the training process.

Output layer modification

The output layer is modified depending upon the number of classes. Number of neurons is equivalent to the number of classes at the output layer.

Addition of layers

Some dense(fully-connected) layers are added at the bottom of these ready-made architectures just before the output layer to make the model more effective and suitable for use in accordance with the complexity and the format of the dataset