الجامعة المصرية
للتعلم الإلكتروني الأهلية

EELU
THE EGYPTIAN E-LEARNING UNIVERSITY

Silent Conversations

Sign Language Recognition in Deafness and Hearing Loss Using Artificial Intelligence

2023 - 2024

Supervisors

Dr.Ahmed Abdelraheem

Lecturer, Department of Information Technology at EELU

Eng.Aya Magdy Youssef

Teacher Assistant, Department of Information Technology at EELU

Team Members

Amira Ahmed
2001213

Sama Alaa
2001081

Sara Ahmed
2000987

Rawan Hamada
2001021

Amira Hashem
2000739

Mona Hassan
2000769

Zyad Medhat
2001083

Agenda

01

Introduction

Deaf and hearing-impaired people face many challenges and problems in their daily lives. They suffer from difficulty communicating with people who do not master sign language.

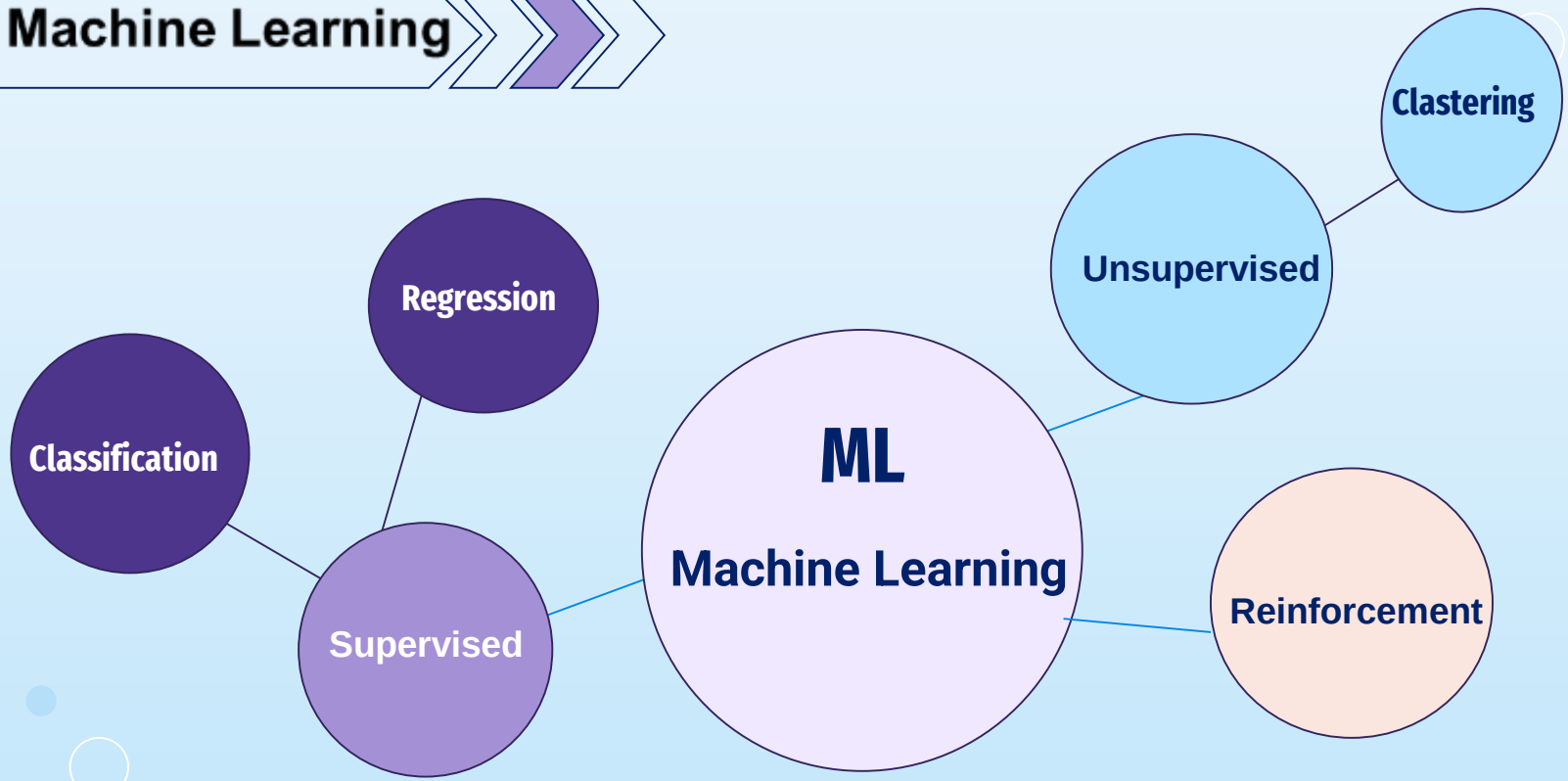
It is difficult for them to exchange thoughts, feelings, and daily needs. They may feel isolated and cut off from society. This social isolation can negatively affect mental health. And emotional to them.

Therefore, the idea of our project is to create a mobile application that is a silent conversation to learn sign language for individuals who suffer from deafness or hearing loss, to bridge the communication gap between the deaf community and the hearing world by enabling simultaneous translation of sign language gestures using all algorithms and computer vision techniques.

02

Background

Machine Learning

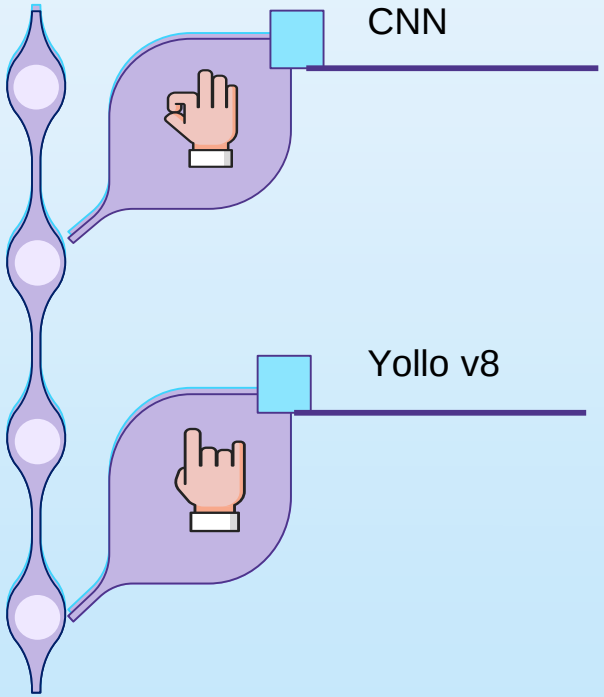


Deep Learning

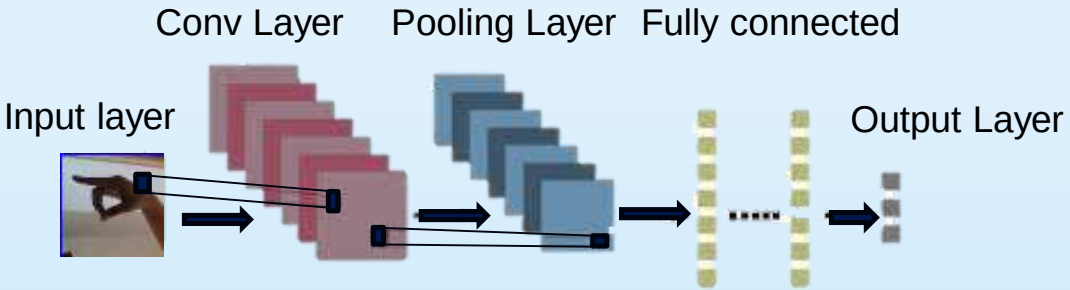
DL

Is a new field of research that deals with finding algorithms that allow the machine to learn on its own.
techniques enable this automatic learning through the absorption of huge amounts of unstructured data such as text, images or video.

Algorithms



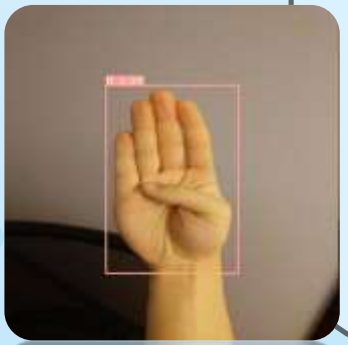
Convolutional Neural Networks – CNN Algorithm



We will use the CNN model, which is a type of neural network that is used in computer vision, and it is one of the very important fields in deep learning. It processes and recognizes data (images and videos). It is designed to deal with this type of two- and three-dimensional data and extract information from it. This is done through number of layers that make up the model, which enable it to better recognize images and classify them correctly. It can deal with a huge amount of data. The more numerous and diverse data is available, the better its work.

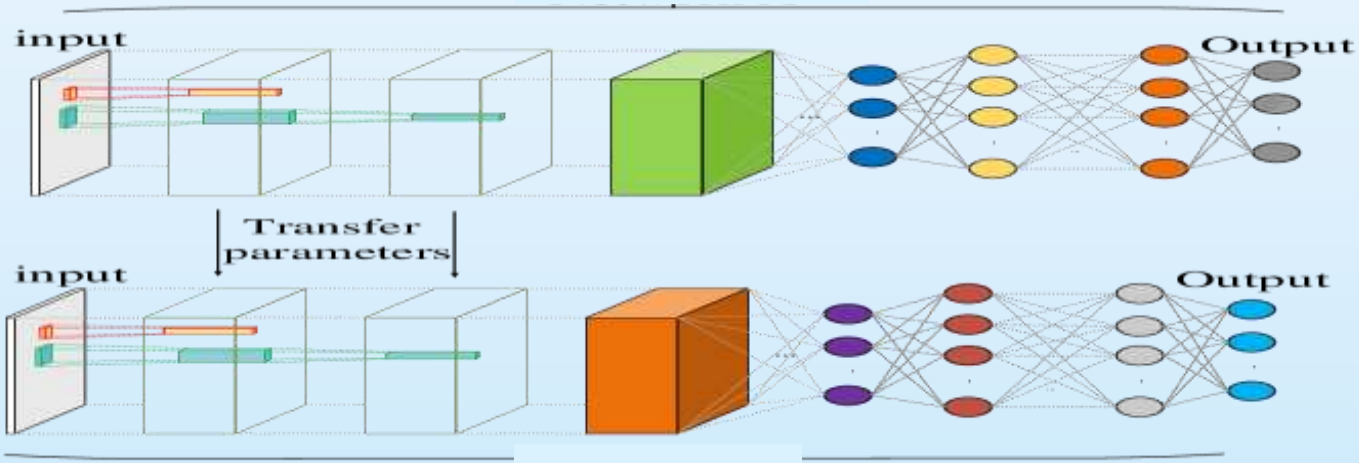
Yolo v8 Algorithm

is an object detection model that uses deep neural networks. YOLO takes an innovative approach to the object detection process, dividing the image into a grid of cells or several square sections, such as (3*3), and each square examines the image to determine whether it contains the required part or not and then analysing the image as a whole in one go, making it more efficient and faster than traditional models that divide the image into small sectors and analyse them separately



Pre-trained Models

Pre – train model



CNN new task

A pretrained model is a deep learning model someone else built that’s trained on large datasets to accomplish a specific task and solve some problem , and it can be used as is or customized to suit application requirements across multiple industries.

03

Literature Review (Related Work)

	Name	Year	Algorithm	Dataset	Accuracy
1	A New Benchmark on American Sign Language Recognition using Convolutional Neural Network	2019	CNN	Synthetic ASL Alphabet	99.96%
2	Sign Language Alphabet Recognition Using Convolution Neural Network	2021	CNN	Sign Language (ENG Alphabet)	99.63%
3	CNN Model for American Sign Language Recognition	2020	CNN	Sign Language MNIST	99.30%
4	Hyper tuned Deep Convolutional Neural Network for Sign Language Recognition	2023	CNN	Sign Language MNIST	99%

	Name	Year	Algorithm	Dataset	Accuracy
5	Sign Language to Sentence Interpreter Using Convolutional Neural Network in Real Time	2022	CNN	ASL(American Sign Language) Alphabet Dataset	98.7%
6	Classification of Sign Language Characters by Applying a Deep Convolutional Neural Network	2020	CNN	Sign Language MNIST	97.62%
7	Sign language recognition system for communicating to people with disabilities	2023	CNN	ASL Handsign Dataset (Grayscale & Thresholder)	96.3%

	Name	Year	Algorithm	Dataset	Accuracy
8	A Translator for Sign Language to Multilingual Text and Speech	2023	CNN	Beginner to Intermediate NLP Tutorial	95.20%
9	Reconstruction of CNN for Sign Language Recognition	2020	CNN	ASL Fingerspelling Recognition w/ TensorFlow	92.21%
10	Arabic Sign Language Recognition and Generating Arabic Speech Using Convolutional Neural Network	2020	CNN	RGB Arabic Alphabets Sign Language Dataset	90%

04

System Methodology

Steps of creating a CNN model

01 Data Collection

02 Data Splitting

03 Data Preprocessing

04 CNN Model

05 Model Training

06 Model Evaluation

07 Improve Performance

08 Final Model Test

09 Integrate the model into an application

Step 01

Data collectionIt
collects a large set of classified data that contains images and videos of sign language and then loads it.

Step 02

Data splitting
The data will be divided into a group for training and a group for testing, and the division depends on its size.

Step 03

Data Preprocessing
In this step, we need to handle missing Values, Outlier Detection and Handling, Data normalization, encoding categories, reduce dimensions(resize), remove redundancy, data augmentation and others, as required by the data, to make it more homogeneous.

Step 04

CNN model

Here, the CNN model is built so that it consists of number of layers that are appropriate for the model to analyze data (images and videos), and it consists of number of hyperparameters that are set and modified until we achieve the highest accuracy of the model.

Step 05

Model training

Use the training set to train the model it and adjust the hyperparameters based on the model's performance on the test set

Step 06

Model evaluation

In it, the trained model is applied to the test data, and its accuracy is measured to determine the extent to which it works correctly.

Step 07

Improve performance

The hyperparameters are adjusted if the accuracy of the model is low and there is an Overfitting or Bias until we reach the highest accuracy.

Step 08

Final model test

Test the model on new data to ensure its ability to recognize sign language and function correctly

Step 09

Integrate the model into an application

After verifying the model's performance and accuracy, we integrate it with a mobile application.

Dataset of

- → English
- → Arabic
- → Spanish
- → India

The Dataset (Done or not)

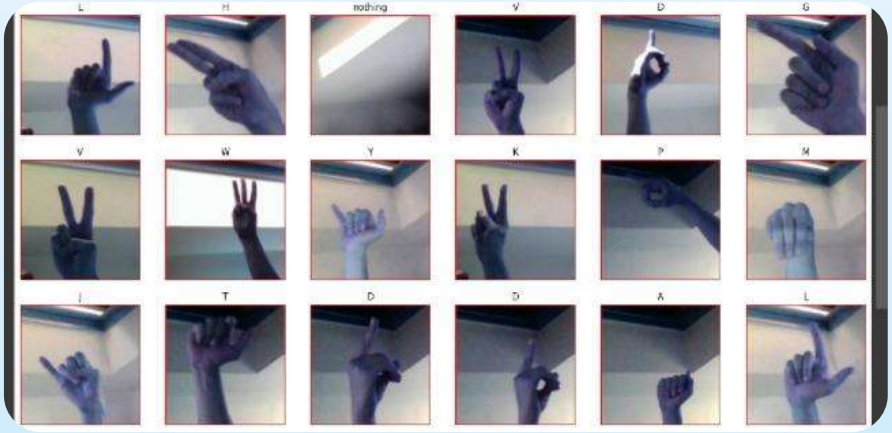
------	English	Arabic	Spanish	India
Alphapts	✓	✓	✓	✓
Words	✓	✓	✓	✓
Sentences	✓	✓	✗	✓

American Alphabet dataset

The dataset is About:

The dataset is a collection of images of alphabets from the American Sign Language, separated in 29 folders of 26 are for the letters A-Z and 3 classes for SPACE, DELETE and NOTHING which are 200x200 pixels.

samples from the dataset



Split Dataset

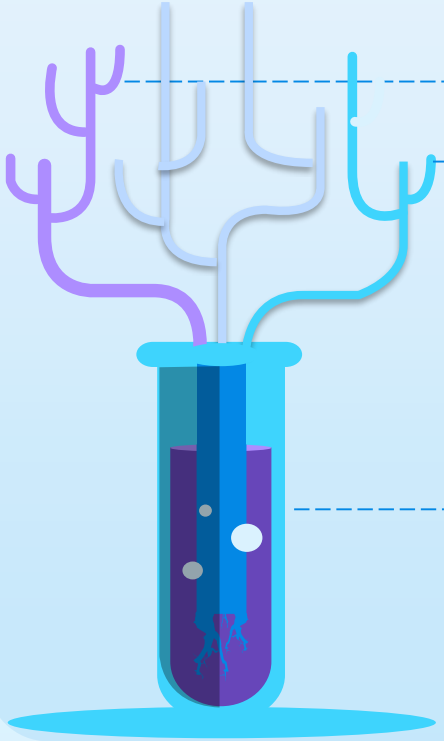
X_Train
X-validation
87000 items

Y_Train
Y-validation
87000 items

Test
28 items

Split=%20:80
shuffle=True,
random_state=100

Preprocessing



- Normalization (/255)
- Scaling
- Resizing (100*100)

Create a CNN model on the Dataset

Layer type	Layer type Description	Size
Input Layer	Input Image	3×100×100
Conv 1	Convolutional ReLU MaxPooling	32 kernel, 3x3 Window size 2×2
Conv 2	Convolutional ReLU MaxPooling	64 kernel, 5x5 Window size 2×2
Conv 3	Convolutional ReLU MaxPooling	128 kernel, 5x5 Window size 2×2
Conv 4	Convolutional ReLU MaxPooling	256 kernel, 3x3 Window size 2×2

Layer type	Layer type Description	Size
Flatten	Flatten	
Fully Connected Layer1	Dense ReLU	512
Fully Connected Layer2	Dense ReLU	265
Fully Connected Layer3	Dense ReLU	182
Output Layer	Dense SoftMax	29

Epochs' Number

1

2

```
epochs = 25
ThisModel = KerasModel.fit(X_train, y_train,
                           epochs=epochs,
                           batch_size=64,
                           validation_data=(X_val, y_val),
                           verbose=1,
                           shuffle=False)
```

```
1088/1088 [==============================] - 26s 24ms/step - loss: 0.0411 - accuracy: 0.9908 - val_loss: 0.0291 - val_accuracy: 0.9934
Epoch 15/25
1088/1088 [==============================] - 26s 24ms/step - loss: 0.0439 - accuracy: 0.9909 - val_loss: 0.1504 - val_accuracy: 0.9721
Epoch 16/25
1088/1088 [==============================] - 26s 24ms/step - loss: 0.0388 - accuracy: 0.9922 - val_loss: 0.0386 - val_accuracy: 0.9921
Epoch 17/25
1088/1088 [==============================] - 26s 24ms/step - loss: 0.0318 - accuracy: 0.9933 - val_loss: 0.0758 - val_accuracy: 0.9859
Epoch 18/25
1088/1088 [==============================] - 26s 24ms/step - loss: 0.0419 - accuracy: 0.9924 - val_loss: 0.0216 - val_accuracy: 0.9964
Epoch 19/25
1088/1088 [==============================] - 26s 24ms/step - loss: 0.0236 - accuracy: 0.9956 - val_loss: 0.0118 - val_accuracy: 0.9973
Epoch 20/25
1088/1088 [==============================] - 26s 23ms/step - loss: 0.0400 - accuracy: 0.9933 - val_loss: 0.0561 - val_accuracy: 0.9874
Epoch 21/25
1088/1088 [==============================] - 26s 24ms/step - loss: 0.0396 - accuracy: 0.9923 - val_loss: 0.0353 - val_accuracy: 0.9927
Epoch 22/25
1088/1088 [==============================] - 25s 23ms/step - loss: 0.0281 - accuracy: 0.9947 - val_loss: 0.0606 - val_accuracy: 0.9851
Epoch 23/25
1088/1088 [==============================] - 26s 24ms/step - loss: 0.0325 - accuracy: 0.9940 - val_loss: 0.0306 - val_accuracy: 0.9934
Epoch 24/25
1088/1088 [==============================] - 25s 23ms/step - loss: 0.0389 - accuracy: 0.9936 - val_loss: 0.0113 - val_accuracy: 0.9975
Epoch 25/25
1088/1088 [==============================] - 25s 23ms/step - loss: 0.0351 - accuracy: 0.9945 - val_loss: 0.0224 - val_accuracy: 0.9958
```

Evaluation

1

```python
# Evaluate the model on the train data
final_train_loss = ThisModel.history['loss'][-1]
final_train_accuracy = ThisModel.history['accuracy'][-1]

print("Final Train Loss:", final_train_loss)
print("Final Train Accuracy:", final_train_accuracy)
```

```
Final Train Loss: 0.035144366323947906
Final Train Accuracy: 0.9945258498191833
```

2

```python
# Evaluate the model on the validation data
final_val_loss = ThisModel.history['val_loss'][-1]
final_val_accuracy = ThisModel.history['val_accuracy'][-1]

print("Final Validation Loss:", final_val_loss)
print("Final Validation Accuracy:", final_val_accuracy)
```

```
Final Validation Loss: 0.022375954315066338
Final Validation Accuracy: 0.9958046078681946
```

ASL_and_Words

The dataset is About:

The dataset is a collection of images of Letters (a, b, c, …, z), Words (Help, Like, Brother, Baby, …) and Numbers (1, 2, 3, …, 9) from the American Sign Language, separated 203000 images in 51 folders, folder have 4,000 image, Common images in data sets are 200 x 200 pixels, and there are images 100 x 100 and other images that differ in their dimensions. which represent the various classes.

Samples from the dataset

Split Dataset

X_Train

X-validation
203000 items

Y_Train

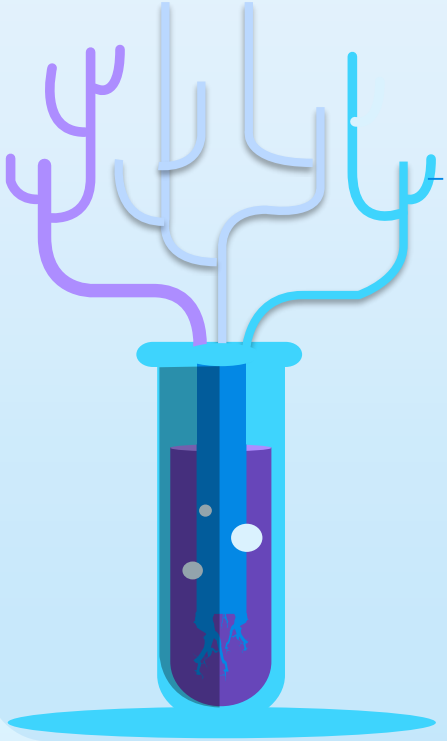
y-validation
203000 items

Test

51 items

Split=%20:80
shuffle=True,
random_state=100

Preprocessing



Resizing (64*64)

Create a CNN model on the Dataset

Layer type	Layer type Description	Size
Input Layer	Input Image	3×64×64
Conv 1	Convolutional ReLU MaxPooling	32 kernel, 3x3 Window size 2×2
Conv 2	Convolutional ReLU MaxPooling	64 kernel, 5x5 Window size 2×2
Conv 3	Convolutional ReLU MaxPooling	128 kernel, 5x5 Window size 2×2
Conv 4	Convolutional ReLU MaxPooling	256 kernel, 3x3 Window size 2×2

Layer type	Layer type Description	Size
Flatten	Flatten	
Fully Connected Layer1	Dense ReLU	512
Fully Connected Layer2	Dense ReLU	256
Fully Connected Layer3	Dense ReLU	128
Output Layer	Dense SoftMax	51

Epochs' Number

1

2

```
epochs = 15
ThisModel = KerasModel.fit(X_train, y_train,
                           epochs=epochs,
                           batch_size=64,
                           validation_data=(X_val, y_val),
                           verbose=1,
                           shuffle=False)
```

```
Epoch 4/15
2538/2538 [==============================] - 30s 12ms/step - loss: 0.2488 - accuracy: 0.9323 - val_loss: 0.2932 - val_accuracy: 0.9217
Epoch 5/15
2538/2538 [==============================] - 30s 12ms/step - loss: 0.2219 - accuracy: 0.9405 - val_loss: 0.2644 - val_accuracy: 0.9326
Epoch 6/15
2538/2538 [==============================] - 30s 12ms/step - loss: 0.2052 - accuracy: 0.9472 - val_loss: 0.2107 - val_accuracy: 0.9462
Epoch 7/15
2538/2538 [==============================] - 29s 12ms/step - loss: 0.1896 - accuracy: 0.9518 - val_loss: 0.2811 - val_accuracy: 0.9326
Epoch 8/15
2538/2538 [==============================] - 30s 12ms/step - loss: 0.1885 - accuracy: 0.9540 - val_loss: 0.2241 - val_accuracy: 0.9543
Epoch 9/15
2538/2538 [==============================] - 29s 12ms/step - loss: 0.1911 - accuracy: 0.9552 - val_loss: 0.2373 - val_accuracy: 0.9490
Epoch 10/15
2538/2538 [==============================] - 32s 13ms/step - loss: 0.1885 - accuracy: 0.9562 - val_loss: 0.2155 - val_accuracy: 0.9540
Epoch 11/15
2538/2538 [==============================] - 30s 12ms/step - loss: 0.1751 - accuracy: 0.9602 - val_loss: 0.2950 - val_accuracy: 0.9408
Epoch 12/15
2538/2538 [==============================] - 30s 12ms/step - loss: 0.1989 - accuracy: 0.9568 - val_loss: 0.1709 - val_accuracy: 0.9646
Epoch 13/15
2538/2538 [==============================] - 29s 12ms/step - loss: 0.1983 - accuracy: 0.9581 - val_loss: 0.2235 - val_accuracy: 0.9556
Epoch 14/15
2538/2538 [==============================] - 32s 13ms/step - loss: 0.2153 - accuracy: 0.9562 - val_loss: 0.3286 - val_accuracy: 0.9365
Epoch 15/15
2538/2538 [==============================] - 30s 12ms/step - loss: 0.1789 - accuracy: 0.9623 - val_loss: 0.2015 - val_accuracy: 0.9605
```

Evaluation

1

```
# Evaluate the model on the train data
final_train_loss = ThisModel.history['loss'][-1]
final_train_accuracy = ThisModel.history['accuracy'][-1]

print("Final Train Loss:", final_train_loss)
print("Final Train Accuracy:", final_train_accuracy)
```

```
Final Train Loss: 0.17890425026416779
Final Train Accuracy: 0.9622783064842224
```

2

```
# Evaluate the model on the validation data
final_val_loss, final_val_accuracy = KerasModel.evaluate(X_val, y_val)

print("Final Validation Loss:", final_val_loss)
print("Final Validation Accuracy:", final_val_accuracy)
```

```
1269/1269 [==============================] - 5s 4ms/step - loss: 0.2015 - accuracy: 0.9605
Final Validation Loss: 0.20148353278636932
Final Validation Accuracy: 0.9604679942131042
```

PreTrain Model: Split Dataset

X_Train

X-validation
87000 items

Y_Train

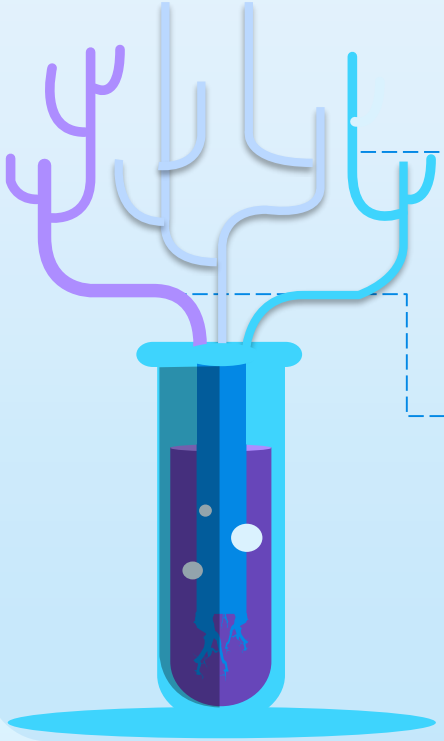
Y-validation
87000 items

Test

28 items

Split=10:90%
shuffle=True,
random_state=13

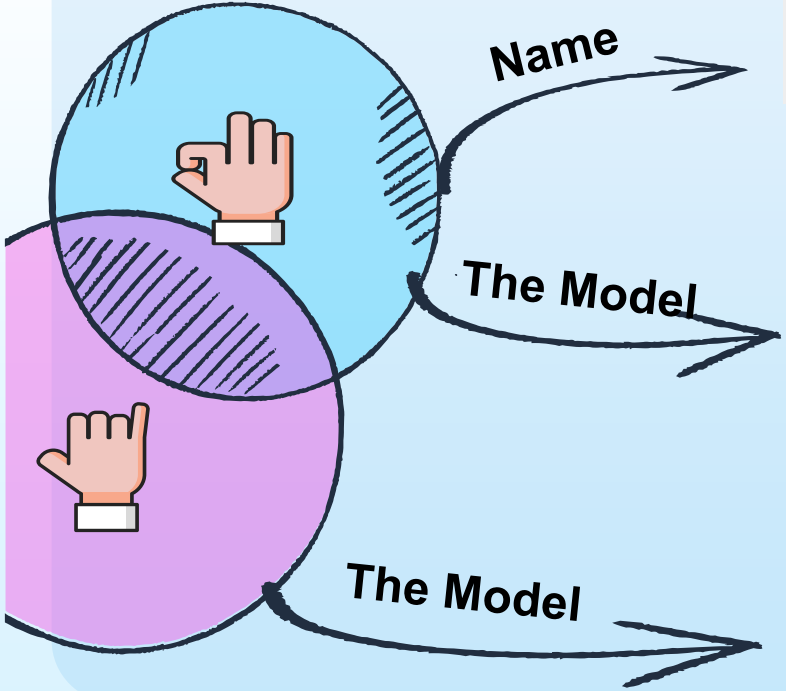
Preprocessing



Resizing (200*200)

Augmentation

Pre Train Model

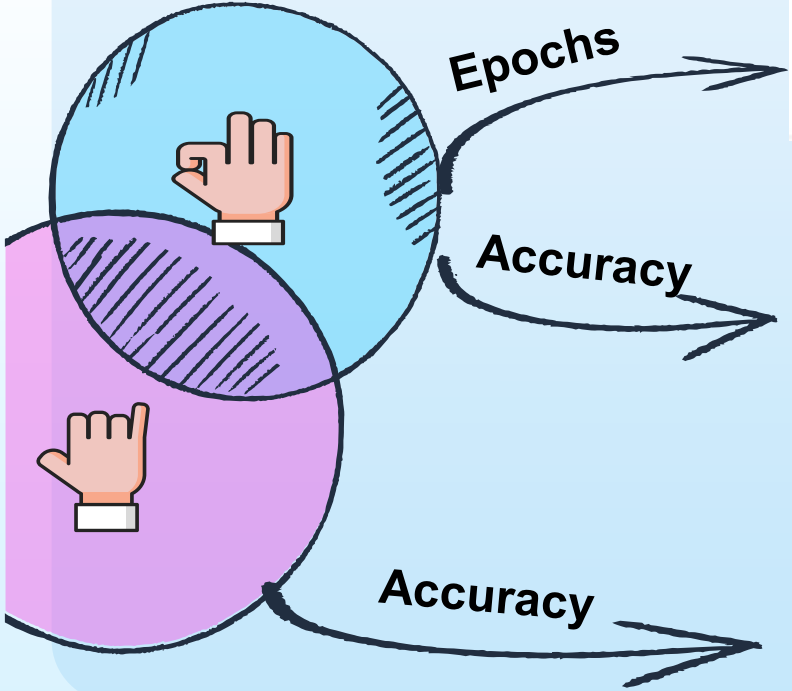



```python
WEIGHTS_FILE = './inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5'

inception_v3_model = keras.applications.inception_v3.InceptionV3(
    input_shape = (200, 200, 3),
    include_top = False,
    weights = 'imagenet'
)

inception_v3_model.summary()
```

```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
input_1 (InputLayer)            [(None, 200, 200, 3)] 0          []

conv2d (Conv2D)                 (None, 99, 99, 32)   864        ['input_1[0][0]']

batch_normalization (BatchNorm  (None, 99, 99, 32)   96         ['conv2d[0][0]']
alization)

activation (Activation)         (None, 99, 99, 32)   0          ['batch_normalization[0][0]']

conv2d_1 (Conv2D)               (None, 97, 97, 32)   9216       ['activation[0][0]']

batch_normalization_1 (Batch    (None, 97, 97, 32)   96         ['conv2d_1[0][0]']
Normalization)

activation_1 (Activation)       (None, 97, 97, 32)   0          ['batch_normalization_1[0][0]']

conv2d_2 (Conv2D)               (None, 97, 97, 64)   18432      ['activation_1[0][0]']

batch_normalization_2 (Batch    (None, 97, 97, 64)   192        ['conv2d_2[0][0]']
Normalization)

activation_2 (Activation)       (None, 97, 97, 64)   0          ['batch_normalization_2[0][0]']

max_pooling2d (MaxPooling2D     (None, 48, 48, 64)   0          ['activation_2[0][0]']
)

conv2d_3 (Conv2D)               (None, 48, 48, 80)   5120       ['max_pooling2d[0][0]']

batch_normalization_3 (Batch    (None, 48, 48, 80)   240        ['conv2d_3[0][0]']
Normalization)

activation_3 (Activation)       (None, 48, 48, 80)   0          ['batch_normalization_3[0][0]']

conv2d_4 (Conv2D)               (None, 46, 46, 192)  138240     ['activation_3[0][0]']

batch_normalization_4 (Batch    (None, 46, 46, 192)  576        ['conv2d_4[0][0]']
Normalization)
```

```python
from tensorflow.keras import layers
x = layers.GlobalAveragePooling2D()(inception_output)
x = layers.Dense(1024, activation='relu')(x)
x = layers.Dense(29, activation='softmax')(x)

model = Model(inception_v3_model.input, x)

model.compile(
    # optimizer=SGD(lr=0.0001, momentum=0.9),
    optimizer=SGD(learning_rate=0.0001, momentum=0.9),
    loss='categorical_crossentropy',
    metrics=['acc']
)
for layer in model.layers[:249]:
    layer.trainable = False
for layer in model.layers[249:]:
    layer.trainable = True
```

Pre Train Model



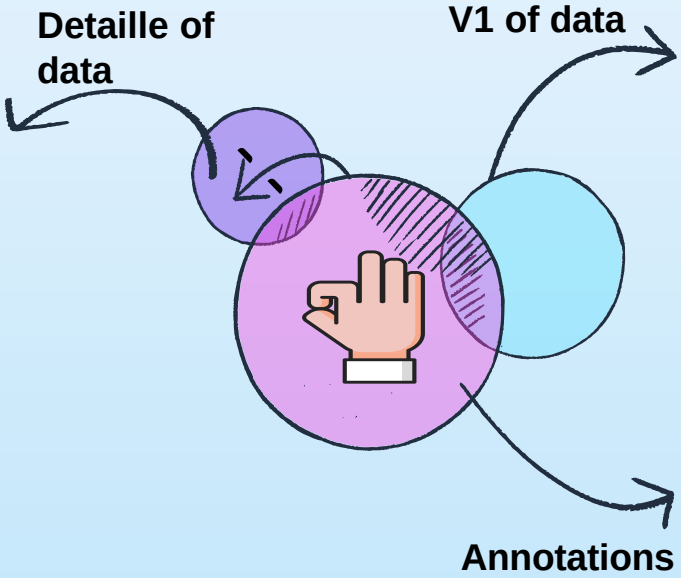

```
Epoch 8/25
200/200 [==============================] - 217s 1s/step - loss: 0.3284 - acc: 0.9575 - val_loss: 0.4094 - val_acc: 0.9284
Epoch 9/25
200/200 [==============================] - 217s 1s/step - loss: 0.2360 - acc: 0.9688 - val_loss: 0.3204 - val_acc: 0.9397
Epoch 10/25
200/200 [==============================] - 217s 1s/step - loss: 0.1851 - acc: 0.9778 - val_loss: 0.2504 - val_acc: 0.9578
Epoch 11/25
200/200 [==============================] - 219s 1s/step - loss: 0.1449 - acc: 0.9817 - val_loss: 0.2153 - val_acc: 0.9631
Epoch 12/25
200/200 [==============================] - ETA: 0s - loss: 0.1185 - acc: 0.9852
Reached 95.# accuracy, Stopping!
200/200 [==============================] - 215s 1s/step - loss: 0.1185 - acc: 0.9852 - val_loss: 0.1993 - val_acc: 0.9553
```

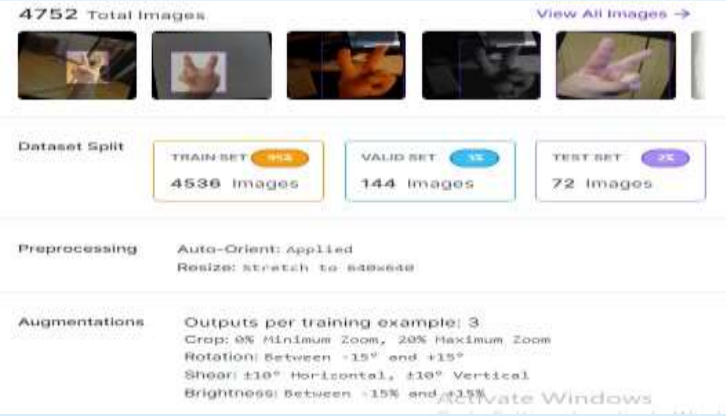
```
[17] # Evaluate the model on the train data
     final_train_loss = history.history['loss'][-1]
     final_train_accuracy = history.history['acc'][-1]

     print("Final Train Loss:", final_train_loss)
     print("Final Train Accuracy:", final_train_accuracy)

     Final Train Loss: 0.11854083091020584
     Final Train Accuracy: 0.9852343797683716
```

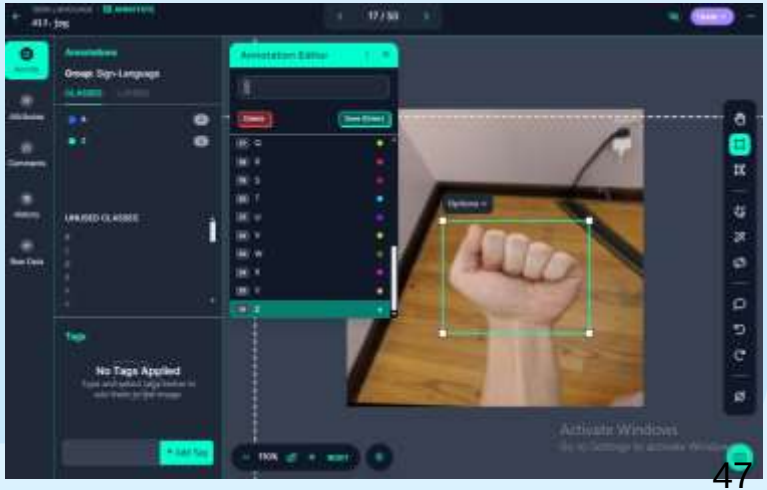
```
# Evaluate the model on the validation data
final_val_loss = history.history['val_loss'][-1]
final_val_accuracy = history.history['val_acc'][-1]

print("Final Validation Loss:", final_val_loss)
print("Final Validation Accuracy:", final_val_accuracy)

Final Validation Loss: 0.19925682246685028
Final Validation Accuracy: 0.9553124904632568
```

Roboflow

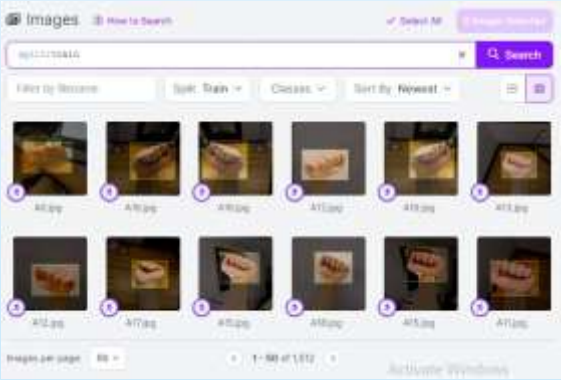

SIGN LANGUAGE
Sign Language
Object Detection
Data
Classes 26
Upload Data
Assign Images
Annotate
Dataset 1728
Health Check

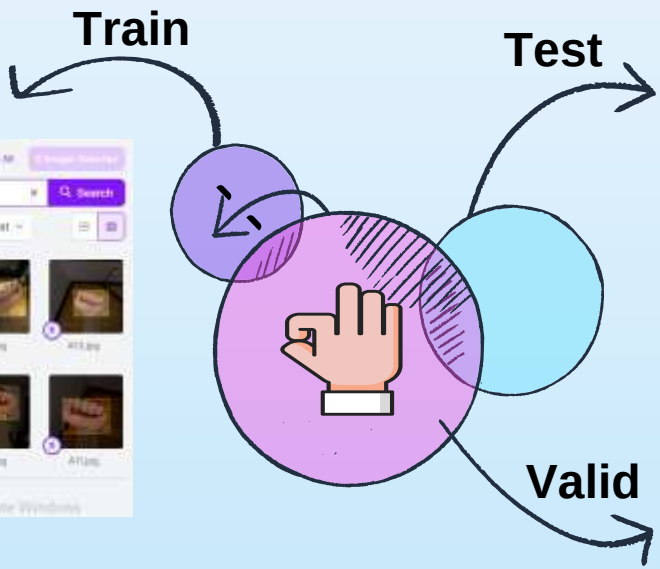

Detaille of data
V1 of data
Annotations

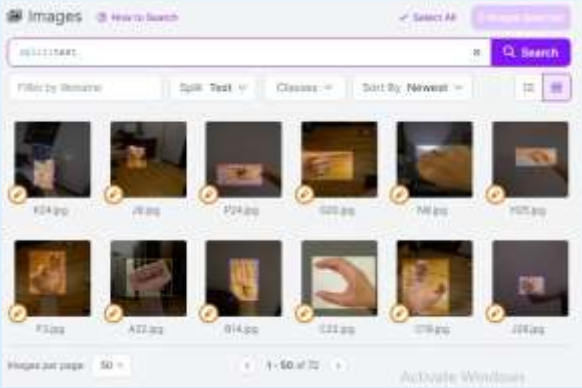

4752 Total Images
View All Images →
Dataset Split
TRAIN SET
4536 Images
VALID SET
144 Images
TEST SET
72 Images
Preprocessing
Auto-Orient: Applied
Resize: Stretch to 640x640
Augmentations
Outputs per training example: 3
Crop: 0% Minimum Zoom, 20% Maximum Zoom
Rotation: Between -15° and +15°
Shear: ±10° Horizontal, ±10° Vertical
Brightness: Between -15% and +15%

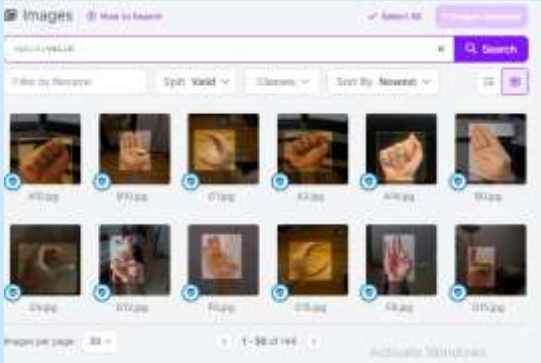

Annotations
Group: Sign-Language
Annotation Editor
Tags
No Tags Applied

Roboflow

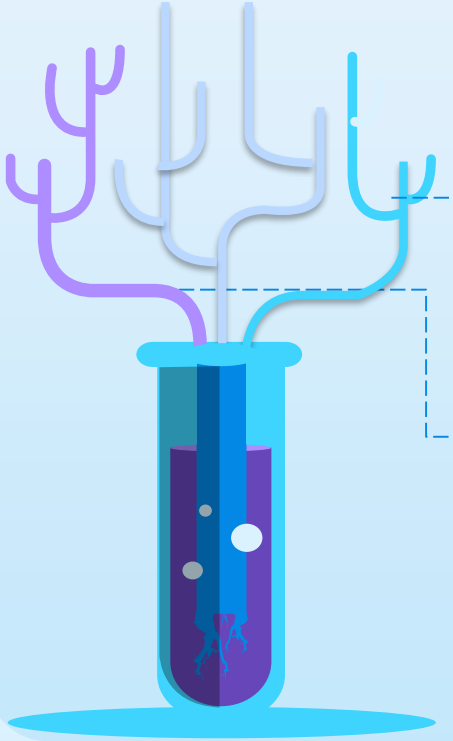

Images
Search
Split: Train


Train
Test
Valid


Images
Search
Split: Test


Images
Search
Split: Valid

Yolov8 Preprocessing



Resizing (640*640)

Augmentation

Yolov8 Model: Split Dataset

Train

87%

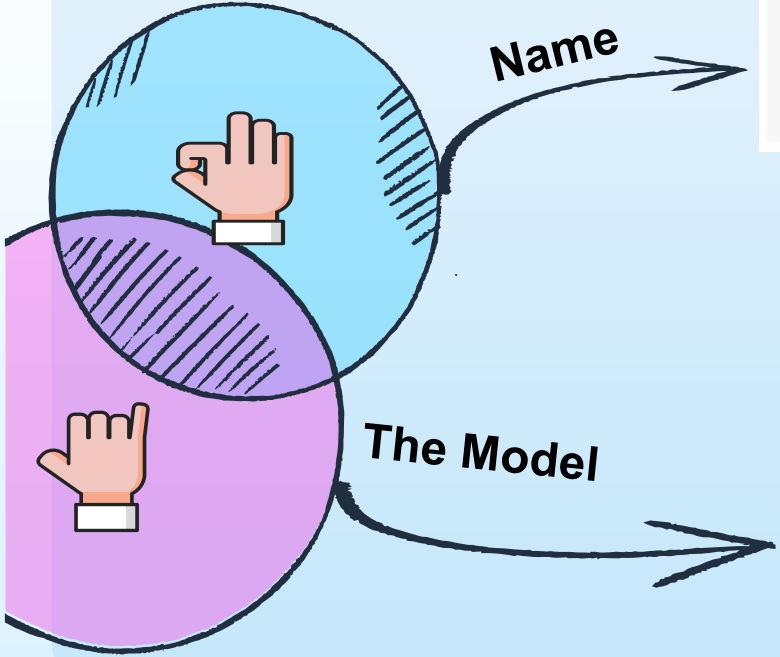
Validation

8%

Test

4%

Yolov8 Model

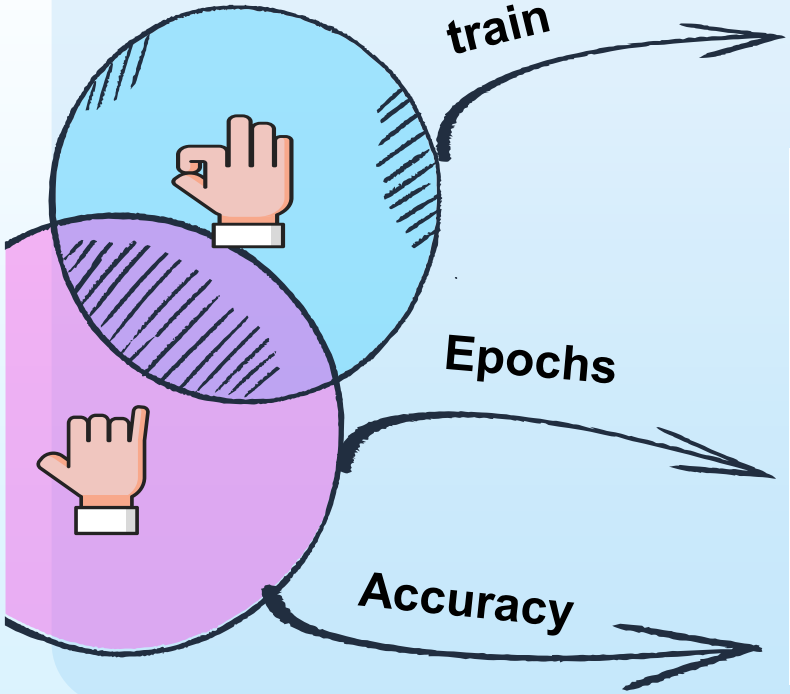



```python
from roboflow import Roboflow
rf = Roboflow(api_key="xFNaby4ELDIoNzLMlxT3")
project = rf.workspace("amira-hashem-8cyvn").project("sign-language-bdxtw")
version = project.version(2)
dataset = version.download("yolov8")
```

```python
# Import the YOLO module from ultralytics
from ultralytics import YOLO
```

```python
# Load a pretrained Yolov8 model
model = YOLO('yolov8n.pt')
```

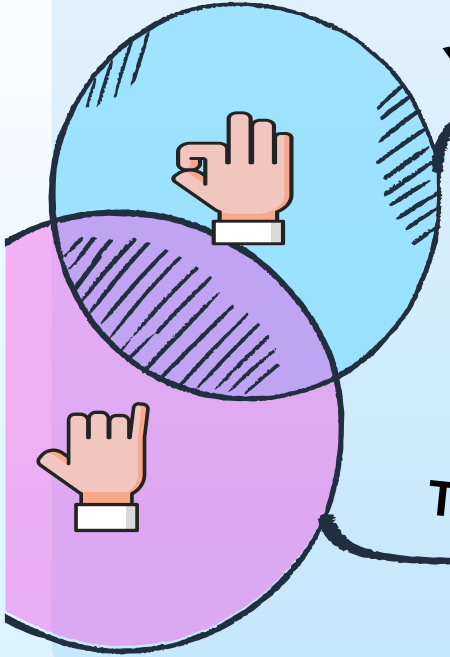
Yolov8 Model




```
!yolo task = detect mode = train model = yolov8n.pt data = {dataset.location}/data.yaml epochs = 20 imgsz=640
```

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
14/20      2.30G     0.6188     0.6537      1.207          6        640: 100% 226/226 [01:10<00:00,  3.21it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 11/11 [00:03<00:00,  3.01it/s]
             all        348        348      0.631      0.638      0.712       0.42

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
15/20      2.30G     0.5918     0.6131      1.188          6        640: 100% 226/226 [01:10<00:00,  3.20it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 11/11 [00:03<00:00,  2.82it/s]
             all        348        348      0.681      0.715      0.742      0.447

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
16/20      2.30G     0.582     0.5806      1.174          6        640: 100% 226/226 [01:11<00:00,  3.16it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 11/11 [00:03<00:00,  2.97it/s]
             all        348        348      0.652      0.632      0.726      0.464

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
17/20      2.30G     0.5523     0.5432      1.145          6        640: 100% 226/226 [01:11<00:00,  3.18it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 11/11 [00:03<00:00,  2.86it/s]
             all        348        348      0.655      0.698      0.768        0.7

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
18/20      2.30G     0.5406     0.5198       1.13          6        640: 100% 226/226 [01:09<00:00,  3.25it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 11/11 [00:04<00:00,  2.25it/s]
             all        348        348      0.652      0.723      0.764      0.692

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
19/20      2.30G     0.5192      0.499      1.118          6        640: 100% 226/226 [01:10<00:00,  3.20it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 11/11 [00:04<00:00,  2.30it/s]
             all        348        348      0.701      0.699      0.764      0.701

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
20/20      2.30G     0.5047     0.4858      1.102          6        640: 100% 226/226 [01:10<00:00,  3.20it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 11/11 [00:03<00:00,  2.93it/s]
             all        348        348      0.726      0.655      0.766      0.705
```

Yolov8 Model



validation

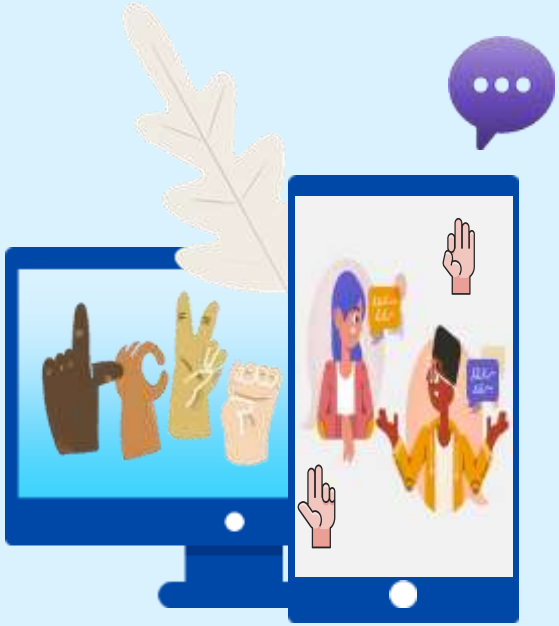
```
# Model validation
!yolo task = detect mode = val model = /content/runs/detect/train/weights/best.pt  data = {dataset.location}/data.yaml
```

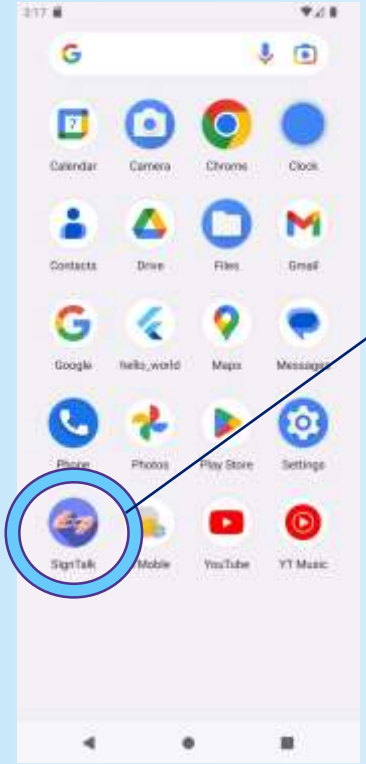
Test

```
# Model Test
!yolo task = detect mode = predict model =/content/runs/detect/train/weights/best.pt  data = {dataset.location}/data.yaml source = {dataset.location}/test/images sa
```

05

Prototype, Tools, and Technologies + Framework



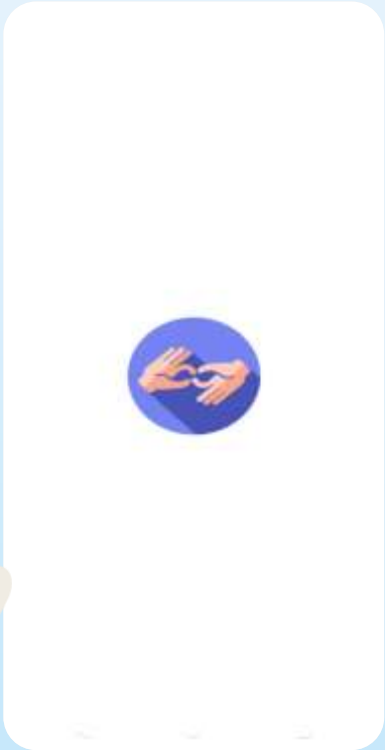


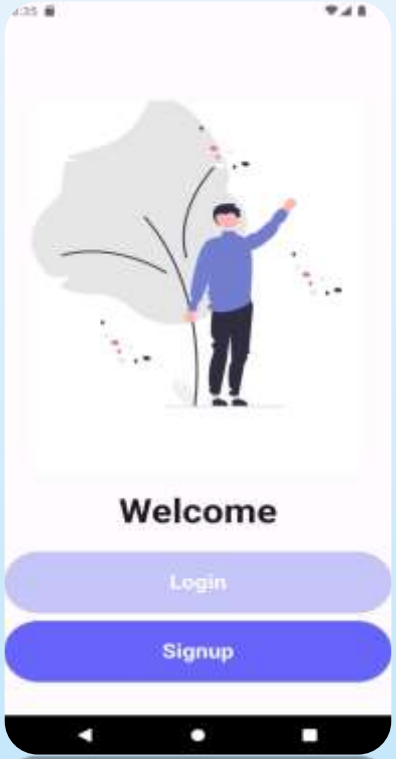
This is the SignTalk application.

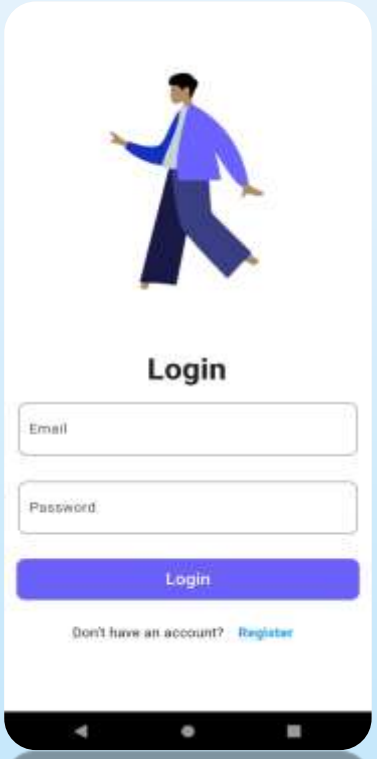
A new application designed to bridge the communication gap between the deaf and hearing communities.

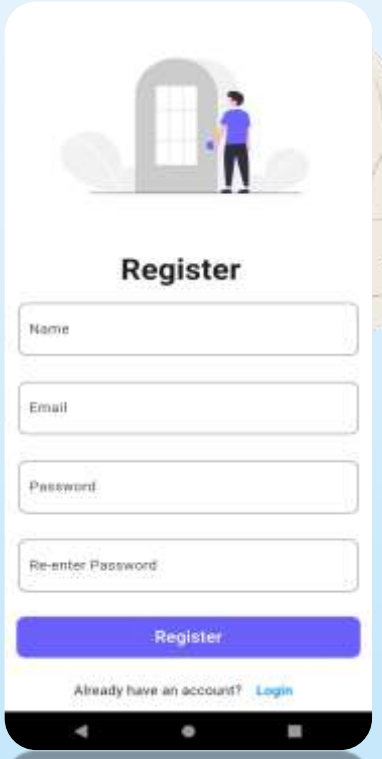
Many deaf and hard of hearing individuals face barriers due to their inability to hear.

SignTalk aims to break down these barriers through real-time sign language translation.




Welcome
Login
Signup


Login
Email
Password
Login
Don't have an account? Register

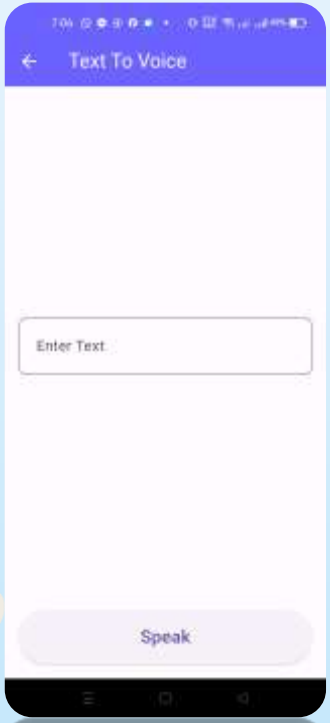

Register
Name
Email
Password
Re-enter Password
Register
Already have an account? Login


Everything is fine
I do not know
Everything is fine
I like it
Choose The Way You Want To Translate:
Convert Sign To Word (Image)
Convert Sign To Word (Video)
Voice to Text
Text to Voice
About
Learn A-Z


Convert Sign To Word
Upload Your image Here
Camera
Gallery


Sign To Word Video


Voice To Text
Speech Is Not Available

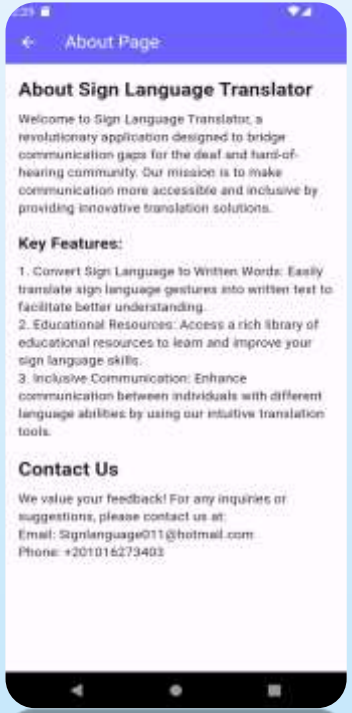

Text To Voice
Enter Text
Speak


Learn A-Z
Alphabets...
Numbers...
1 2 3 4 5
6 7 8 9 10
Words...


Learn A-Z
Words...
Words...


Learn A-Z
Sentences...


Learn A-Z


About Page
About Sign Language Translator
Welcome to Sign Language Translator, a revolutionary application designed to bridge communication gaps for the deaf and hard-of-hearing community. Our mission is to make communication more accessible and inclusive by providing innovative translation solutions.
Key Features:
1. Convert Sign Language to Written Words: Easily translate sign language gestures into written text to facilitate better understanding.
2. Educational Resources: Access a rich library of educational resources to learn and improve your sign language skills.
3. Inclusive Communication: Enhance communication between individuals with different language abilities by using our intuitive translation tools.
Contact Us
We value your feedback! For any inquiries or suggestions, please contact us at:
Email: Signlanguage011@hotmail.com
Phone: +201016273403

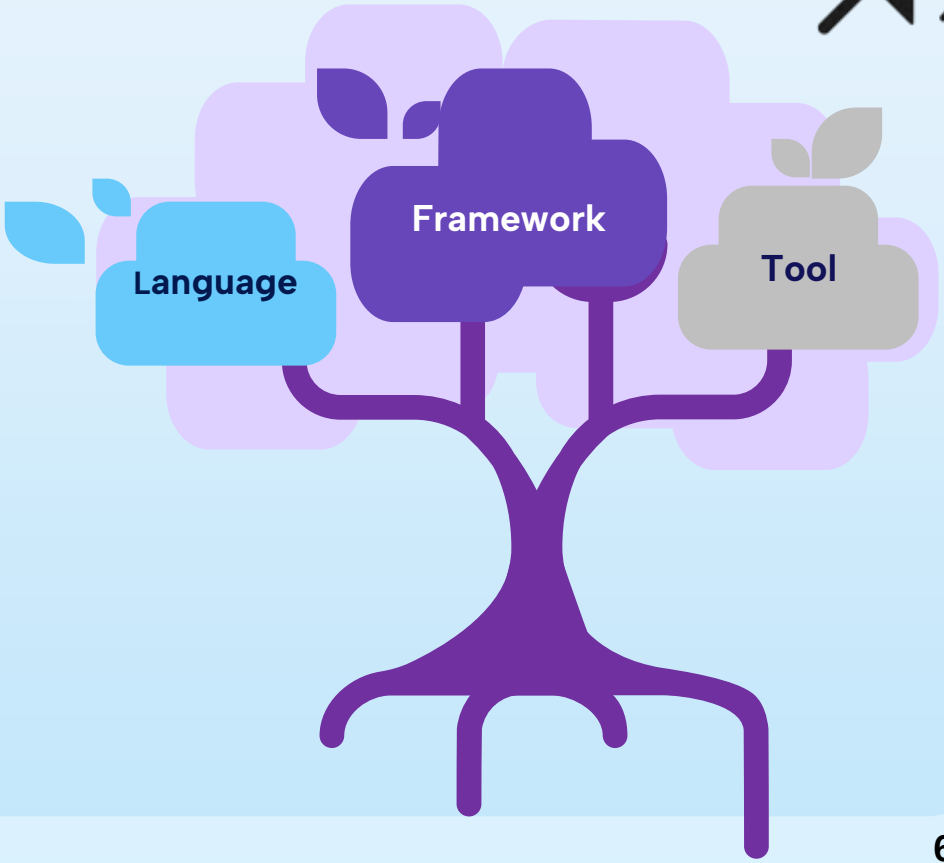

Sign
Talk
TEXT
HERE

Mobile App

Dart

Flutter

Figma

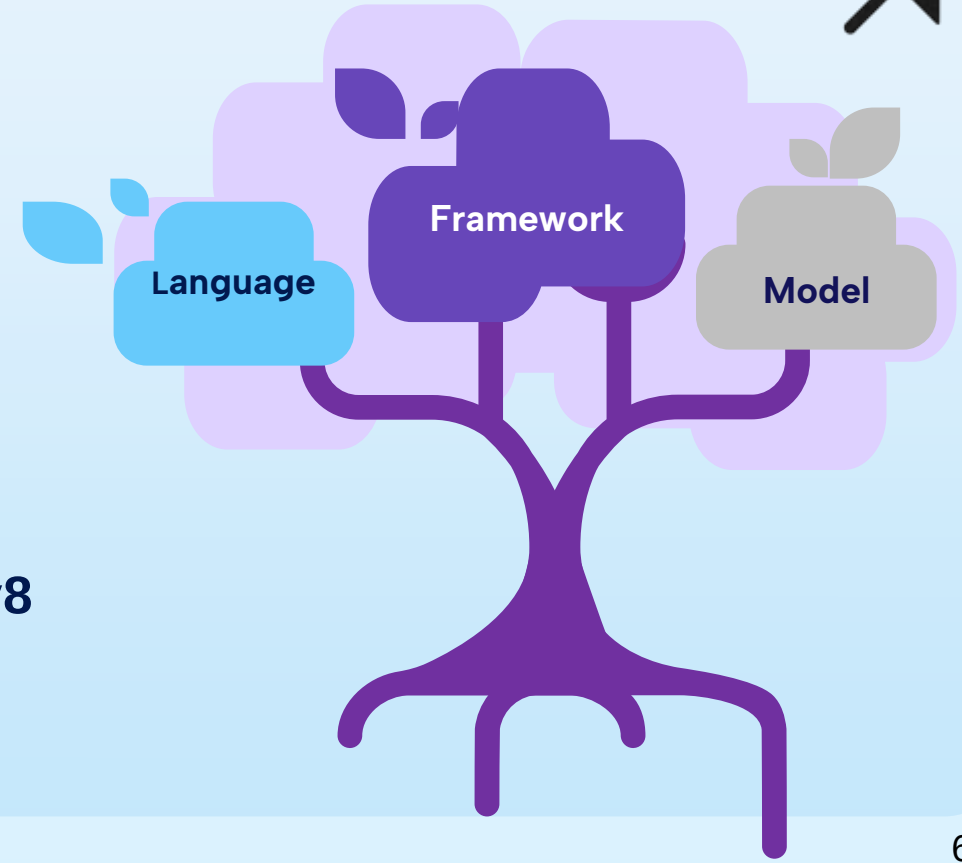


Deep Learning

Python

Google Colab

CNN – Yollo v8



06

Project Scope

Dependances

- Planning and organization
- Effective communication
- Dataset
- Algorithm is suitable
- Evaluation

Risk and Mitigation

The user have more than one language for the deaf.	We collected a set of data from 4 different languages, this data was studied, and then we began to choose the language most commonly spoken among people.
It is possible that the user will choose a word that is not found in the data set.	We combined two data sets.
It is possible that the user may not be proficient in writing to communicate with a deaf person.	Therefore, we added a microphone that converts speech into written text, making communication faster and easier.

Project developments

01

1- We will enhance the application for faster and easier communication by converting speech into sign language, enabling deaf individuals to understand. This makes the application bidirectional.

02

We will add a dataset containing sentences and more phrases to cover a larger portion of the language.

03

We will incorporate multiple languages, allowing any user to use the application in any country and serve a broader audience.

04

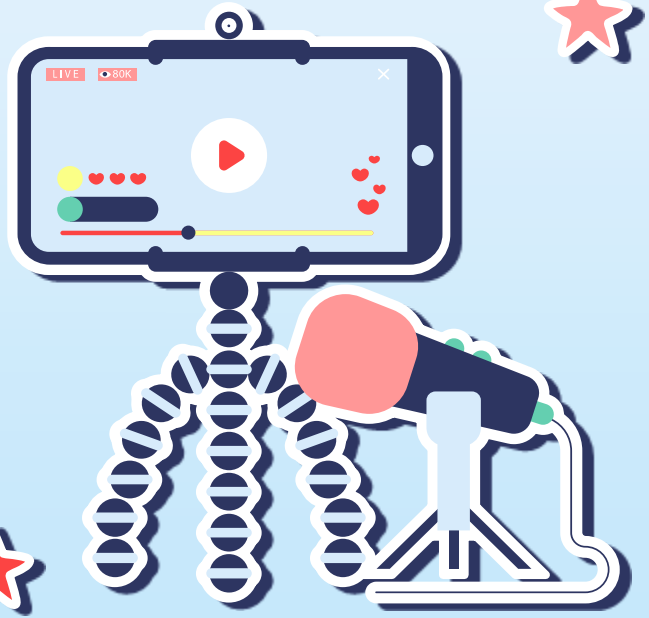
we will convert text into sign language as well.

07

Conclusion

Using deep learning, we created a model that recognizes hand gestures in sign language and translates them into expressive texts that express their needs and facilitate communication. This is done using convolutional neural networks that are trained on a set of properly labeled data and tested to reach the highest accuracy of the model. We linked the model to a mobile phone to help them communicate in daily life.

Demo



Time Plan

Month	Activity	Notes
Jul - Aug - Sep	Study Courses	• AI - Flutter
Oct - Nov	Paper Dataset Model Flutter	• Search of Paper. • Search of dataset and dataset Processing. • Build Model (alphabet) and (Numbers). • Making protoype in figma. • Making home page and nav between all pages.
Dec - Jan	Model Flutter	• Build Model (Words). • Model prediction and Pre train model. • Making login and signup page in add to link it with firebase and making the documentation. • Making a video to explain the app.
Feb - mar	Model Flutter	• Build Model object detection (image) (video) • Making Convert from voice to text
April - may	Flutter	• Connect the model to the mobile app • Make Book

References

[1].**Alphapet_Dataset**

[1]English

- https://www.kaggle.com/datasets/lexset/synthetic-asl-alphabet/data,"
- https://www.kaggle.com/datasets/debashishsau/aslamerican-sign-language-aplhabet-dataset
- https://www.kaggle.com/datasets/datamunge/sign-language-mnist/data
- https://www.kaggle.com/datasets/grassknoted/asl-alphabet/code

[2]Arabic

- https://www.kaggle.com/datasets/sabribelmadoui/arabic-sign-language-unaugmented-dataset
- https://universe.roboflow.com/omdena-pemas/arabic-sl

[3]Spanish

- https://www.kaggle.com/datasets/kirlelea/spanish-sign-language-alphabet-static"

[4]india

- https://universe.roboflow.com/niladri-basu-roy-qnrm4/indian-sign-language-detection
- https://www.kaggle.com/datasets/kartik2112/indian-sign-language-translation-letters-n-digits
- https://www.kaggle.com/datasets/dodiyaparth/indian-sign-language

[2].**Word_Dataset**

[1]English

- "https://www.kaggle.com/datasets/risangbaskoro/wlasl-processed,"
- https://www.kaggle.com/datasets/kshitij192/action-recognition
- https://www.kaggle.com/datasets/belalelwikel/asl-and-some-words/data

[2]Arabic

- https://universe.roboflow.com/rania-hamada-xcozs/arabic-sign-language-words-detection
- https://www.kaggle.com/datasets/mahmoudmsaafan/arabic-sign-language-dataset

[3]Spanish

- https://www.kaggle.com/datasets/mguiralc03/spanishsignlanguagerecognitiontfg”

[4]india

- https://www.kaggle.com/datasets/kshitij192/action-recognition
- https://www.kaggle.com/datasets/daskoushik/include
- https://www.kaggle.com/datasets/soumyakushwaha/indian-sign-language-dataset

[1].**Sentence_Dataset**

[1]English

- “https://www.kaggle.com/datasets/tasinalnahiankhan/phrase-level-asl-converted-in-numpy-array,”

[2]Arabic

- https://www.kaggle.com/datasets/mohamedlotfy50/arabic-sign-language

[4]india

- https://data.mendeley.com/datasets/kcmpdxky7p/1

- **Paper_Link**

- https://sci-hub.se/10.1109/sti47673.2019.9067974
- https://sci-hub.se/10.1109/iciccs51141.2021.9432296
- https://link.springer.com/chapter/10.1007/978-981-15-7961-5_6
- https://www.hindawi.com/journals/cin/2022/1450822/
- https://link.springer.com/chapter/10.1007/978-981-99-3734-9_32
- https://ieeexplore.ieee.org/abstract/document/9491897
- https://www.sciencedirect.com/science/article/pii/S1877050922021846/pdf?
- https://scholarworks.calstate.edu/downloads/dj52wb92k
- https://sci-hub.se/https://ieeexplore.ieee.org/abstract/document/9179356/
- https://www.hindawi.com/journals/wcmc/2020/3685614/

الجامعة المصرية
للتعلم الإلكتروني الأهلية

EELU
THE EGYPTIAN E-LEARNING UNIVERSITY

Thanks!