Computer Vision

Sign language to text

Team 37
Enas Ikram Girgis Ibrahim 162021072 CS1
Marley Amged Hamdy Thabet 162021252 CS4
Abram Ashraf Abd El-Sayed Shehata 162021003 CS1

April 10th, 2024

Assiut University

Overview

1. Task Description
2. Demo
3. Contribution
4. Data
5. Project Architecture
6. Methods
7. Results

Task Description

Brief description

Recognize American Sign Language (ASL) gestures in real-time and converting them to text.

Overview

1. Task Description
2. Demo
3. Contribution
4. Data
5. Project Architecture
6. Methods
7. Results

Demo

Github Repository

https://github.com/Mollyamged/Sign_Language_to_text.git

Overview

1. Task Description
2. Demo
3. Contribution
4. Data
5. Project Architecture
6. Methods
7. Results

Contribution

1. Collected and added the data by ourselves.
2. Augmented the percentage of data designated for testing purposes.
3. Added a layer from flatten type
4. Substituted the initial three LSTM layers with Dense layers.

Overview

1. Task Description
2. Demo
3. Contribution
4. Data
5. Project Architecture
6. Methods
7. Results

Data

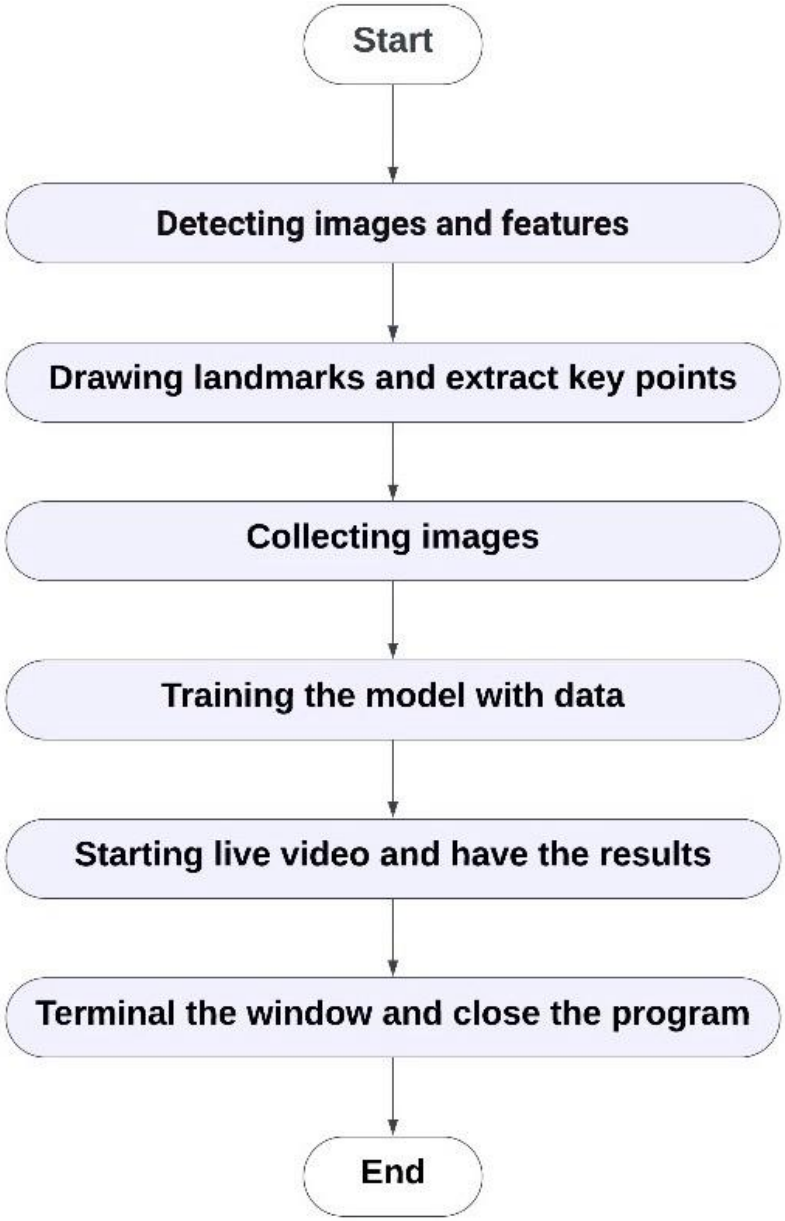
Dataset:

- The dataset consists images for sign language gestures for nine different words.
- Each word is represented by 30 images, resulting in a total of 270 images in the dataset.
- The images are divided into two parts: training set and testing set.10% of the total images are reserved for testing, while the remaining 90% are used for training.

Overview

1. Task Description
2. Demo
3. Contribution
4. Data
5. Project Architecture
6. Methods
7. Results

Project Architecture



Overview

1. Task Description
2. Demo
3. Contribution
4. Data
5. Project Architecture
6. Methods
7. Results

Methods

1. **Input Layer (Flatten):**
   - It converts the input data into a 1-dimensional array.
   - input data with dimensions 30x126.
2. **Dense layer(1):**
   - 64 neurons
   - ReLU activation function
3. **Dropout layer(1):**
   - dropout rate of 0.2
   - That mean 20% of the neurons will be randomly dropped during training to prevent overfitting
4. **Dense layer(2):**
   - 128 neurons
   - ReLU activation function
5. **Dropout layer(2):**
   - dropout rate of 0.2

Methods

6. **Dense layer(3):**
   - 64 neurons
   - ReLU activation function
7. **Dropout layer(3):**
   - dropout rate of 0.2
8. **Dense layer(4):**
   - 64 neurons
   - ReLU activation function
9. **Dropout layer(4):**
   - dropout rate of 0.2
10. **Dense layer(5):**
    - 32 neurons
    - ReLU activation function
11. **Dropout layer(5):**
    - dropout rate of 0.2

Methods

12. **Dense layer(Output Layer):**
    - 9 neurons
    - softmax activation function

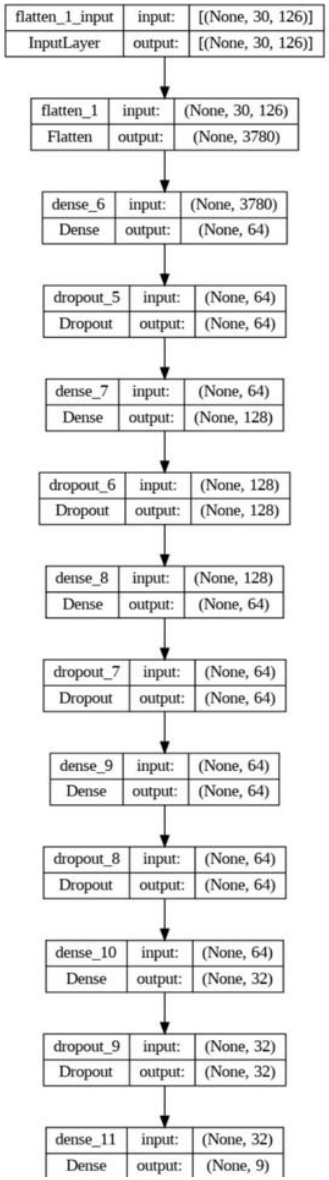
Model Compilation:

- **loss**= 'categorical_crossentropy'
- **optimizer**='adam'
- **metrics**='accuracy'

Model Training:

- **Epochs**=2000
- **batch_size**=32
- **callbacks**=[early_stopping]

Methods


flatten_1_input | input: | [(None, 30, 126)]
InputLayer | output: | [(None, 30, 126)]
flatten_1 | input: | (None, 30, 126)
Flatten | output: | (None, 3780)
dense_6 | input: | (None, 3780)
Dense | output: | (None, 64)
dropout_5 | input: | (None, 64)
Dropout | output: | (None, 64)
dense_7 | input: | (None, 64)
Dense | output: | (None, 128)
dropout_6 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_8 | input: | (None, 128)
Dense | output: | (None, 64)
dropout_7 | input: | (None, 64)
Dropout | output: | (None, 64)
dense_9 | input: | (None, 64)
Dense | output: | (None, 64)
dropout_8 | input: | (None, 64)
Dropout | output: | (None, 64)
dense_10 | input: | (None, 64)
Dense | output: | (None, 32)
dropout_9 | input: | (None, 32)
Dropout | output: | (None, 32)
dense_11 | input: | (None, 32)
Dense | output: | (None, 9)

Overview

1. Task Description
2. Demo
3. Contribution
4. Data
5. Project Architecture
6. Methods
7. Results

Results

1. **Accuracy: 96.3%**
2. **Loss: 14.89%**

```
[40]: loss, accuracy = model.evaluate(X_test, y_test)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.9630 - loss: 0.1489
```