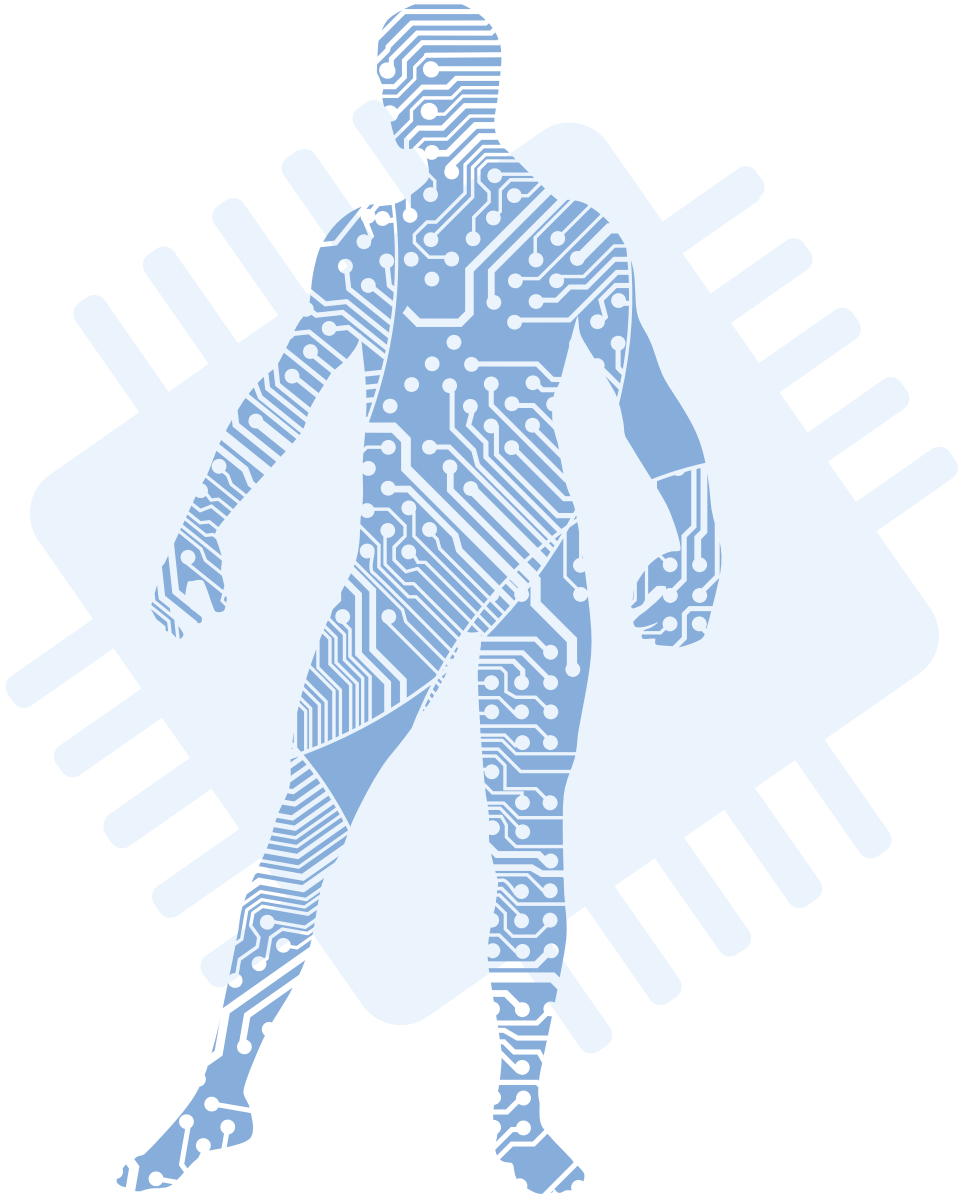
American Sign Language Classifier

Content



Introduction

Introduction

American Sign Language (ASL) is a natural language that servers as the predominant sign language of deaf communities in the United States and most of Anglophone Canada. ASL is a complete and organized visual language that is expressed by employing both manual and nonmanual features.

The use of the hands to represent individual letters of a written alphabet is called “fingerspelling”. It’s an important tool that helps signers manually spell out names of people, places and things that don’t have an established sign.

This project is focused on creating a machine learning model that is capable of classification of letters from the ASL alphabet and in this case to classify the letters into two classes ‘A’ and ‘B’ by taking as input and image of the hand symbol and outputs the class.

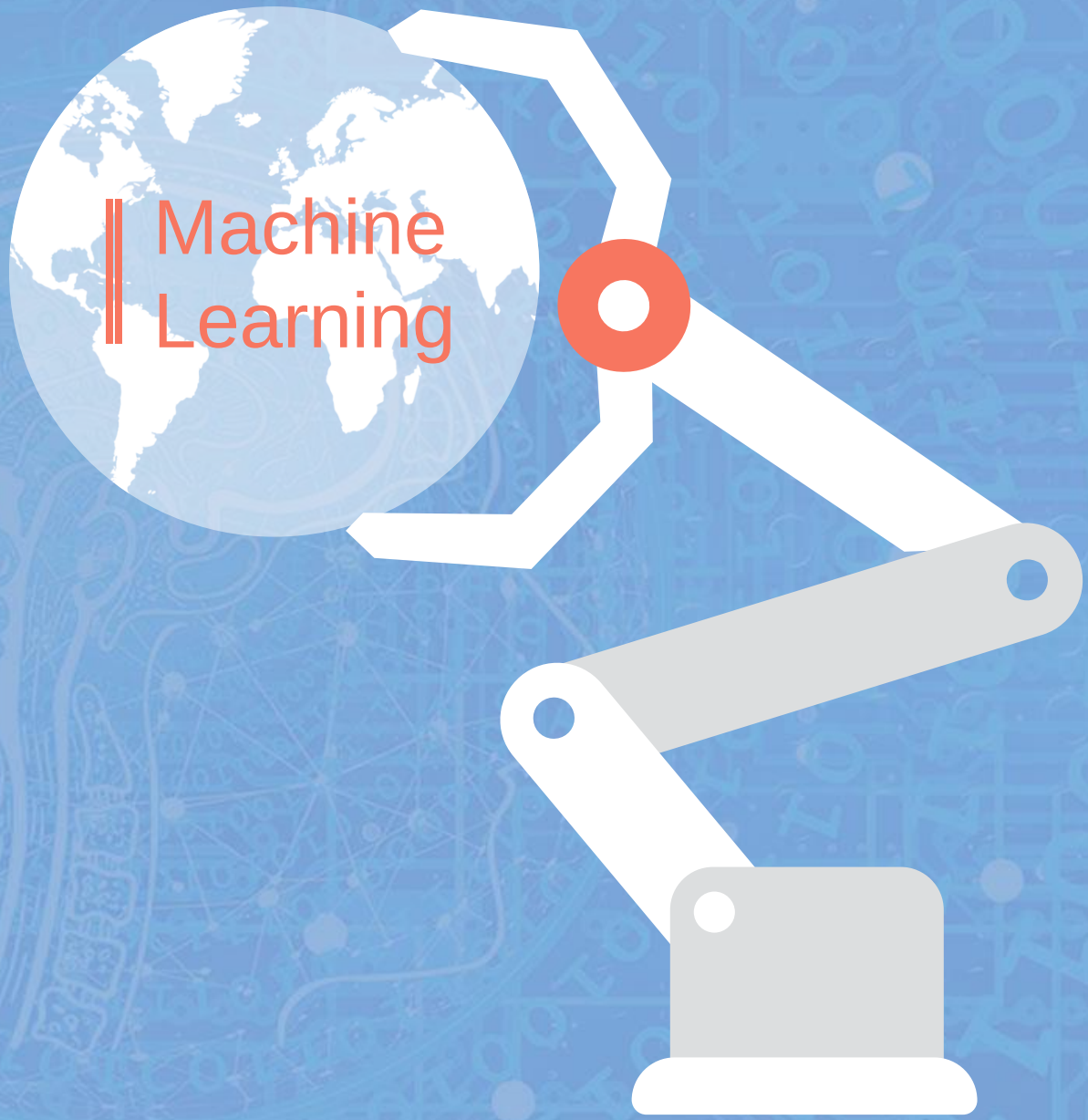
Related work

Related Work

Multiple studies regarding sign language classification were done over the years. The range from using machine learning algorithm to using convolutional neural networks.
Two methods that use ML are:

One study done by S. Shrenika and M. Madhu Bala titled “Sign Language Recognition Using Template Matching Technique” describes a of classification of numbers. The dataset contains hand gestures signifying numbers from 1 – 5 that are classified K-NN on features extracted using contour-based segmentation.

Another one done by M. Quinn and J. I. Olszewska titled “British Sign Language Recognition in the Wild Based on multi-class SVM” approached the problem by using SVM for classification and for features HOG.

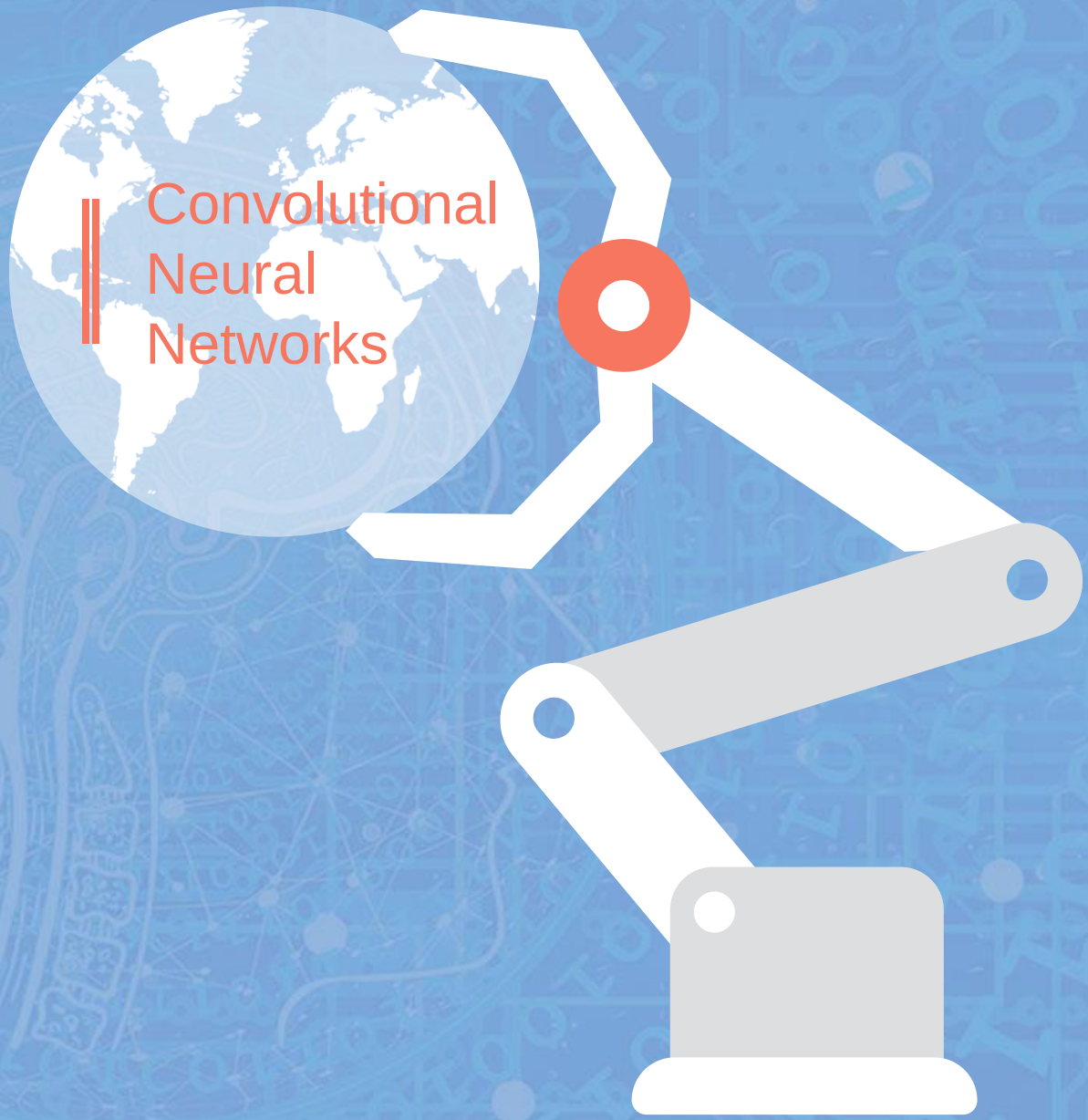


Related Work

Some studies that use CNN instead of ML include:

The study "Static sign language recognition using deep learning,“ by L. K. S. Tolentino, R. 0. Serfa Juan, A. C. Thio-ac, M. A. B. Pamahoy, J. R. R. Forteza, and X. J. 0. Garcia employ a method of classification using a CNN model which has been trained using Keras.

Another study by S. Sharma and K. Kumar, in "ASL-3DCNN: American sign language recognition technique using 3-D convolutional neural networks,“ use a technique called ASL-3DCNN for recognizing American Sign Language using 3D convolutional networks.



Proposed Method

Proposed method

My approach to the problem of classification consists of using AdaBoost for classification and for features using Histogram of Oriented Gradients (HOG).

The dataset consist of images of hands representing the letters “A” and “B”. For each letter, the dataset consists of 70 images divided into groups of 5 similar images that have slight differences in hand and fingers positioning.

The dataset is split into training and test images. For training the first 4 images from each group are taken, 40 of them are chosen and for testing the last image from all the groups is taken, all 14 of them are used.

The implementation consists of a small pipeline that is done to achieve the result of the classification.

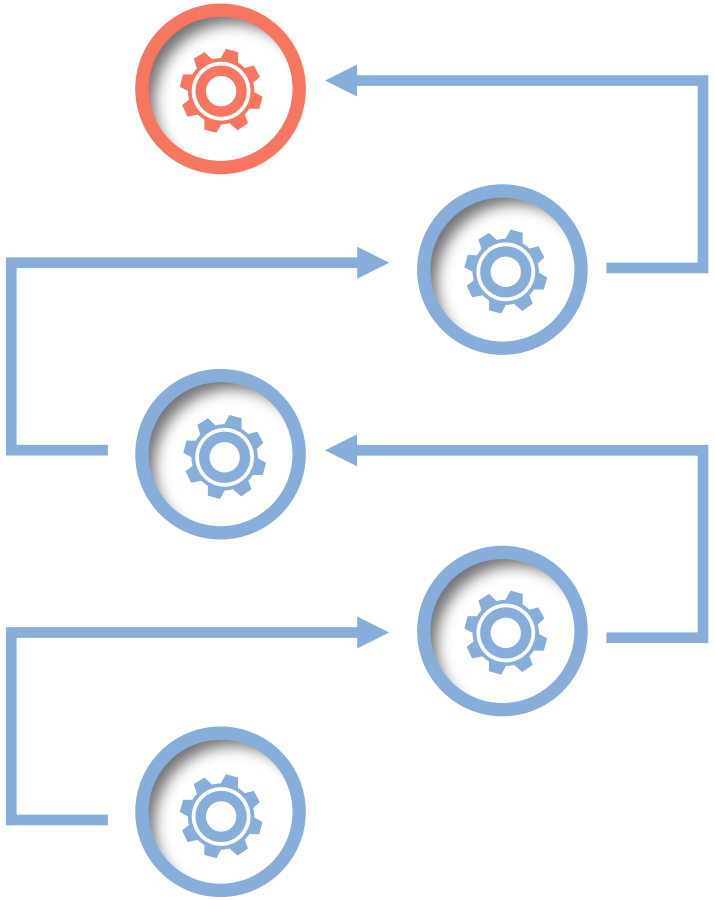
Classification pipeline

Training

The AdaBoost algorithm is trained on the dataset that is represented as a 80x8160 array (X) containing the features and a 80x1 array (y) containing the classes.

Image Preparation

The input image is loaded in color and than it is converted into grayscale because color is not important here. After that it is blurred to remove potential noise and is resized to 128x128px.



Testing

Now the trained model can be used to predict the class of a given image. To assess the performance of the model a test dataset is used similar to the training one, and the model is used to predict the classes for data from X and the result is compared with the corresponding classes from y.

Features Extraction

Each image goes through a HOG algorithm that outputs the histogram in the form of a 1-dimensional array containing 8100 features that will represent a row in the dataset.

Dataset preparation

This is a function that is used to prepare the input images before extracting the features from them

```
function process_image(image):
    gray_image = convert image to grayscale (eliminate color information)
    blurred_image = apply Gaussian blur with a 3x3 kernel
    resized_image = resize blurred_image to dimensions (128, 128)
    return resized_image
```

Dataset preparation

This is the function that is used to extract features from the input image:

function HOG(image, blockSize, cellSize, nbins)
    compute the horizontal gradient (gx) using convolution with a kernel [-1, 0, 1]
    compute the vertical gradient (gy) using convolution with a kernel [[-1], [0], [1]]
    calculate the gradient magnitude: magnitude = sqrt(gx^2 + gy^2)
    calculate the gradient angle (in degrees, range [0, 180)): angle = atan2(gy, gx) * (180 / π) mod 180
    extract cell dimensions (cW, cH) from cellSize
    initialize a 3D histogram array of zeros: hist = array of size (h / cH, w / cW, nbins), initialized to zeros
    for each cell in the image
        for each pixel within the current cell
            determine which bin the gradient angle belongs to
            compute contributions for left and right bins
            update the histogram with weighted contributions

    extract block dimensions (bW, bH) from blockSize
    initialize a list to store normalized histograms

Dataset preparation

```
for each block in the image
    initialize an empty array to store the block's concatenated histogram
    for each cell within the current block
        append the histogram of the current cell to norm
    normalize the block's concatenated histogram
    append the normalized histogram to the list of normalized histograms

return flattened histogram and flattened normalized histogram
```

Dataset preparation

```
function load_data(dir, limit)
    initialize an empty list X (features)
    initialize an empty list y (labels)
    for each class in CLASSES_LIST:
        set the path to the current category folder
        initialize empty lists X_category (features) and y_category (labels)
        for each image in the category folder
            read the image from the file path and process it using process_image(image)
            extract features from the processed image using HOG function
            append the extracted features to X_category
            if the current class is the first in CLASSES_LIST:
                append 1 to y_category (positive label)
            else:
                append -1 to y_category (negative label)
        combine X_category and y_category into a single list of tuples
        shuffle the combined data randomly and split the shuffled data back into X_shuffled and y_shuffled
        append the first `limit` features from X_shuffled to X and from y_shuffled to y
        return X and y
```

Training and testing

X, y = load data from train folder with a limit of 40 images per class by calling load_data() function
classifier = use AdaBoost to find a strong classifier
save the classifier for future use

Xtest, ytest = load data from test folder with a limit of 14 images per class (in this case all images from folder) by calling load_data() function

run the classifier on the test set
compute the confusion matrix based on the predictions and the real labels
using the confusion matrix, compute accuracy, precision, recall, f1

Implementation

Implementation

The implementation consists of multiple algorithm implemented from scratch by me. This includes:

- Apply convolution algorithm
- HOG algorithm
- AdaBoost algorithm for finding the strongest classifier
- Algorithm for computing the confusion matrix and the metrics

The libraries that I used are:

- OpenCV
  - Functions for image manipulation such as blurring the image, converting it to grayscale and rescaling
  - Reading the image from PC
- Matplotlib
  - Function for displaying the image
- Seaborn
  - Function for creating and saving a heatmap that represents the confusion matrix
- Sklearn
  - AdaBoost implementation used for comparison
- Joblib
  - Used for saving and loading the trained model
- Numpy
  - For storing data in matrices and matrix operations
- Math
  - For mathematical operations

Implementation

The features, as mentioned earlier, are represented by histograms of oriented gradients, each one of them containing 8100 features. To achieve this, the HOG function that constructs the histogram uses a cell size of 8x8, a block size of 16x16 and 9 bins.

The image before being fed to the HOG function it is processed by converting it to grayscale, it is blurred using Gausian blur with a 3x3 kernel and resized to 128x128.

The classification algorithm used is AdaBoost. It was trained on 40 random images from 56 that are in total. It uses 300 weak learners that are decision stumps. The result of the prediction is a 1 or a -1, 1 if the image is from the first class and -1 if it is in the second class.

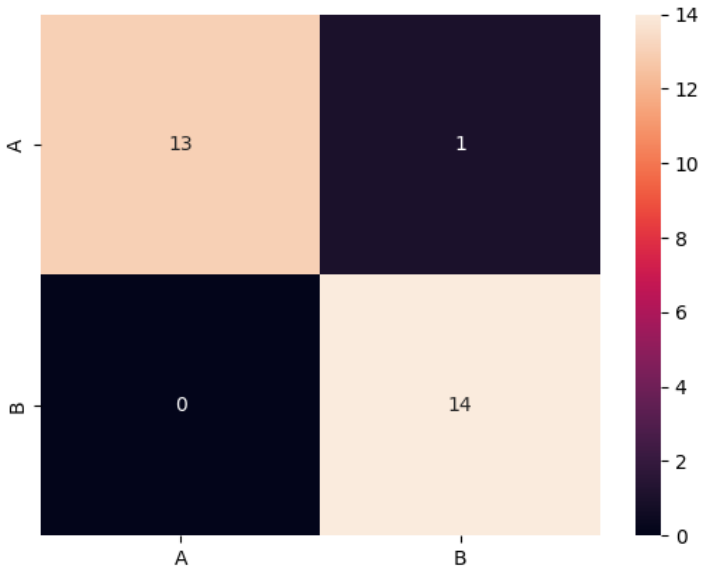
For evaluation, the confusion matrix was used and metrics such as accuracy, precision, recall and f1 were computed.

Regarding the hardware the model was trained on an intel core i7-because and it took approximately 243 minutes to finish training, which may be because it was written in python.

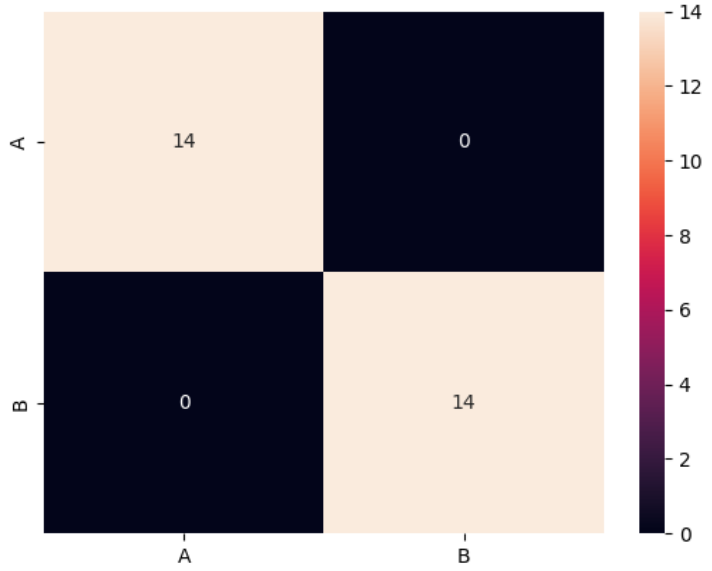
Results

Results

The model was tested on a training set composed of 28 images, 14 from each class and the confusion matrix together with metrics such as accuracy, precision, recoil and f1 were computed. To put in perspective the results a AdaBoost model from scikit learn was trained on the same dataset and the same metrics were computed. The results:



Own AdaBoost implementation



Scikit Learn AdaBoost implementation

Results

	Own AdaBoost	Scikit Learn AdaBoost
Accuracy	96.42%	100.00%
Precision	92.85%	100.00%
Recoil	100.00%	100.00%
F1	96.29%	100.00%

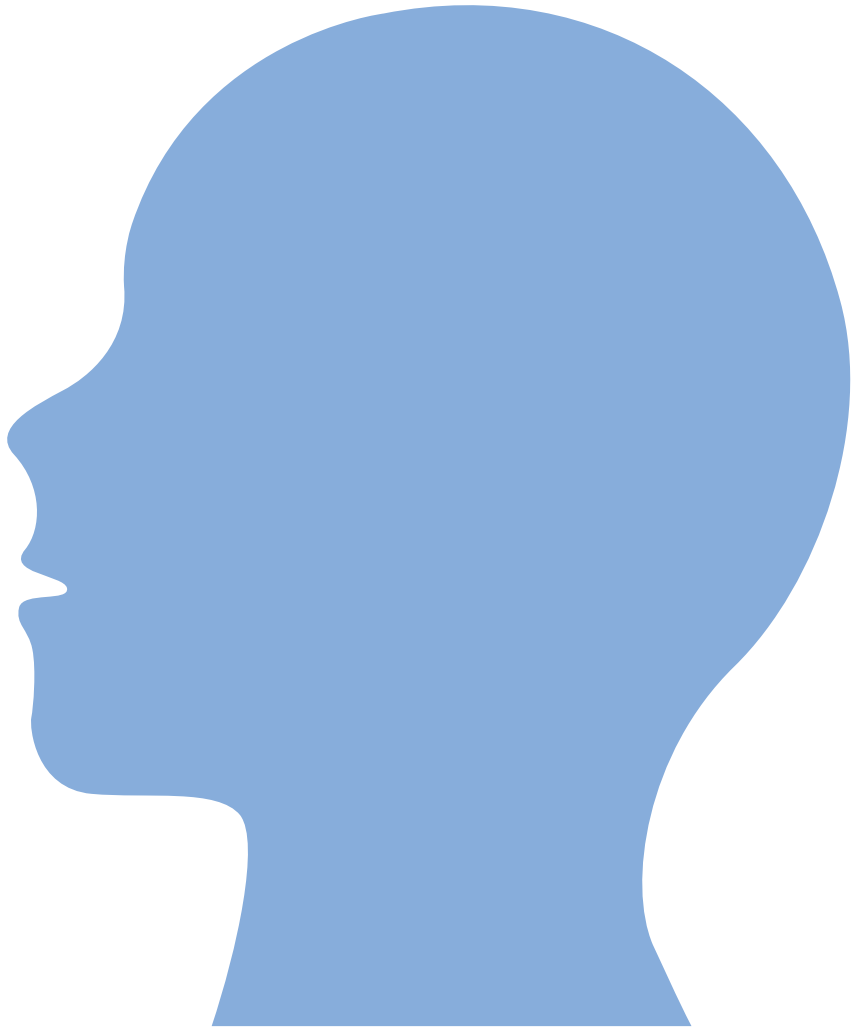
Given the results the Scikit Learn model is better the mine, but the high results can also be a sign of overfitting on both models, as a consequence of training on a small dataset.

Conclusion

Conclusion

What works:
The implementation works as intended, the performance of the model according to the statistics is good and in almost all the cases it will correctly predict the class of the image, but as mentioned earlier the model probably overfitted on the input data.

What doesn’t:
Using other images that don’t have the hand extracted and the background black may lead to unexpected results because the image processing algorithm expects to receive an image of this kind.
Also, other letters beside “A” and “B” can’t be classified because the model was trained only on those two letters.




Thank You