Extraction of a Table from an Image using Efficient Image Processing and ML Algorithms

Prakhar,Roopkatha,Mahesh,Prasanna

Indian Institute of Science

September 17, 2022

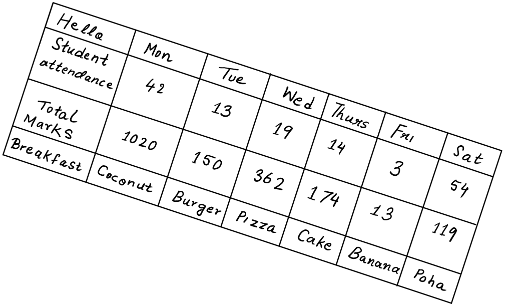
Overview

Goal of the Project

Goal

To convert an image taken from a digital camera of a table hand drawn/written in A4 sheet and then convert it into a digital table using image processing and ML algorithms

Overview of the Project



Hello	Mon	Tue	Wed	Thurs	Fri	Sat
Student attendance	42	13	19	14	3	54
Total Marks	1020	150	362	174	13	119
Breakfast	Coconut	Burger	Pizza	Cake	Banana	Poha

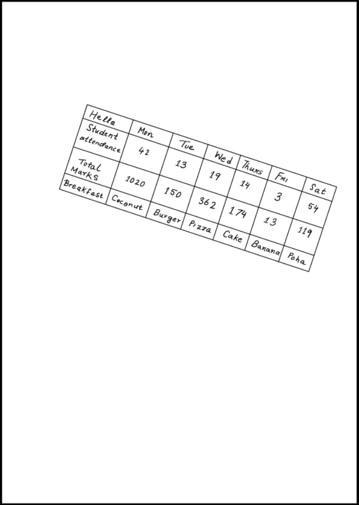
Overview of the Project

- Detect the table and extract each cell using OpenCV.
- Extract the characters from the word and pass it to the classifier
- Classify the character and join back to form the word
- Write the word in the proper cell location in the digital table

Table cell detection with OpenCV

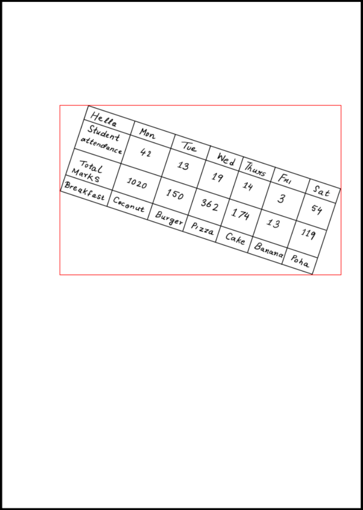
- Start with a picture of an A4 sheet of white paper with the data table hand drawn/written on it.
- Use thresholding to turn image into a binary image.
- Fix the table orientation of the table so that it is vertical.
- Use morphological transformation we keep only the vertical lines and then only the horizontal lines
- Construct a new image from these. Detect the contours of this image.
- Extract the cells from the original image using this.

Original Image



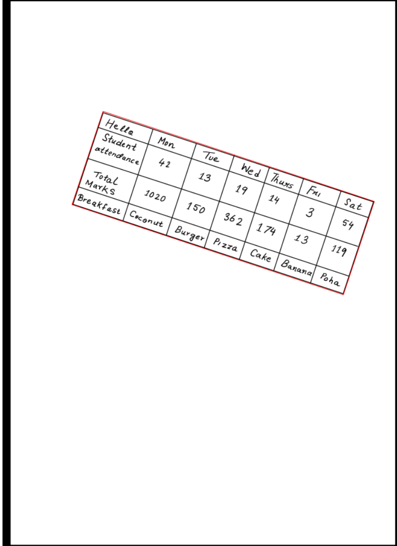
Hello	Mon	Tue	Wed	Thurs	Fri	Sat
Student attendance	42	13	19	14	3	54
Total Marks	1020	150	362	174	13	119
Breakfast	Coconut	Burger	Pizza	Cake	Banana	Poha

Contour with largest area



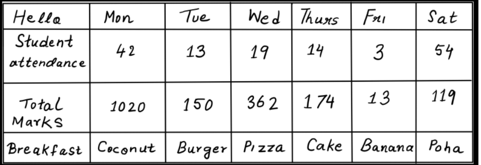
Hello	Mon	Tue	Wed	Thurs	Fri	Sat
Student attendance	42	13	19	14	3	54
Total Marks	1020	150	362	174	13	119
Breakfast	Coconut	Burger	Pizza	Cake	Banana	Poha

Using minAreaRect() on contour with minimum area



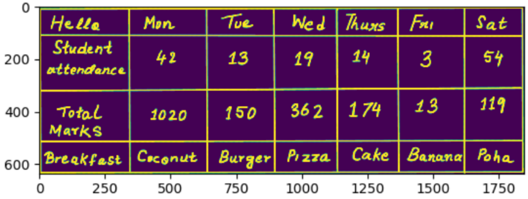
Hello	Mon	Tue	Wed	Thurs	Fri	Sat
Student attendance	42	13	19	14	3	54
Total Marks	1020	150	362	174	13	119
Breakfast	Coconut	Burger	Pizza	Cake	Banana	Poha

Rotate and crop table



Hello	Mon	Tue	Wed	Thurs	Fri	Sat
Student attendance	42	13	19	14	3	54
Total Marks	1020	150	362	174	13	119
Breakfast	Coconut	Burger	Pizza	Cake	Banana	Poha

Thresholded table


0
200
400
600
0
250
500
750
1000
1250
1500
1750

Hello	Mon	Tue	Wed	Thurs	Fri	Sat
Student attendance	42	13	19	14	3	54
Total Marks	1020	150	362	174	13	119
Breakfast	Coconut	Burger	Pizza	Cake	Banana	Poha

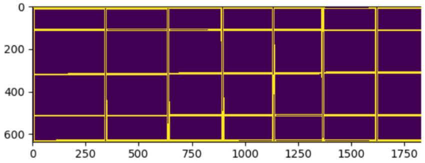


Table lines extracted with erotion and dilation

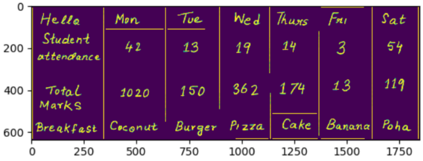
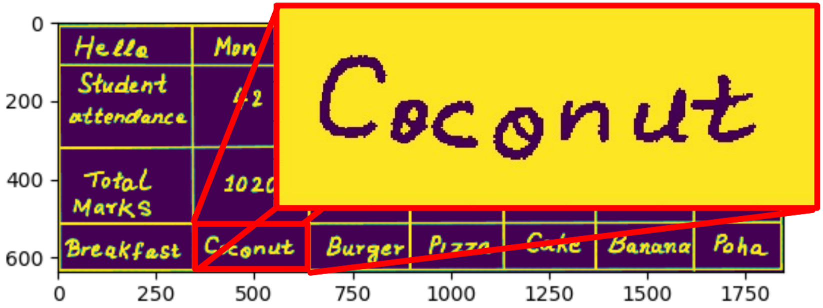

0
200
400
600
0
250
500
750
1000
1250
1500
1750

Table lines removed by drawing coutours

Et voilà!


Coconut
0
200
400
600
0
250
500
750
1000
1250
1500
1750
Hello
Mon
Student attendance
42
Total Marks
1020
Breakfast
Coconut
Burger
Pizza
Cake
Banana
Poha

Assumptions for words in cell

- We assume that the sequence of words are written in a horizontal manner such that each sentence is separable by other sentence by a horizontal line
- To extract a sentence we parse the image row wise and check for ink that is enclosed within two straight horizontal blank lines. This forms the sentence on which the further processing is done

Statement Extraction

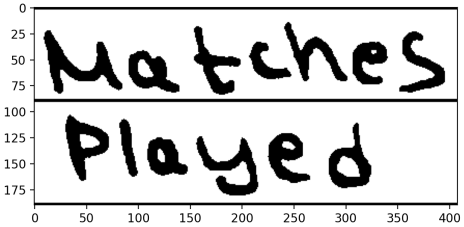


Figure: Statement Segregation

Word Segregation

- Once we have a sentence ready, next step is to extract words out of the sentence
- For that first steps is to calculate amount of whitespace one expects between two words, which is then used as separator
- Once we have the separation width, we use this to separate words just like we extracted statement

Word Segregation

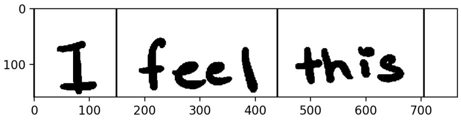


Figure: Word Segregation

Separation Width Calculation

- We iterate over the characters in the sentence and calculate the enclosing box to calculate average
- This estimate of separation width is used as minimum whitespace to segregate the characters

Character extraction steps

- Do a scan from left to right, as soon as "ink" is hit, do a BFS search to extract the full character
- Write the character to separate blank canvas, resize the image and pass it to the classifier
- To handle characters such as i and j, before doing a BFS, a lock in period is defined where as soon as ink is hit, everything within 30% of the separation width is said to be belonging to single character.

Data Set

- The data set that was used is the EMNIST By Class Data .
- This data consists of 47 classes
- Compared to 62 classes, these 47 classes classify 'c' 'C' , 's' 'S' , 'u' 'U' in the same class.
- 112800 samples of train data and 18000 test data .
- Link- https://www.kaggle.com/crawford/emnist/

Models That were Used

- Single Decision Tree and Random Forest.
- Support Vector Machine (SVM)
- K Nearest Neighbour (KNN)
- Voting Classifier
- Multilayer Perceptron
- Convolutional Neural Network

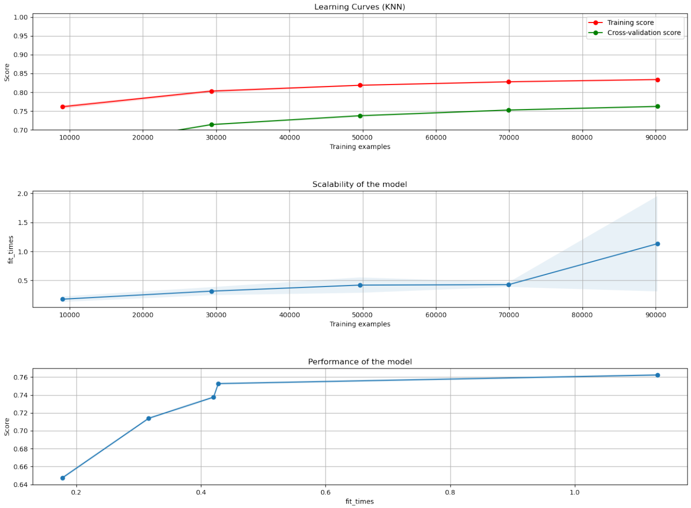
Trial using Random Forest

- 28x28 size image was converted to a linear array of 784 features for each sample
- It achieved a score of 79.8% on the test data when a single decision tree was giving 56.4%
- We are using 100 decision trees. Grid Search on random forest did not change the accuracy much and default values were sufficient.

Trial using K nearest neighbour

- 28x28 size image was converted to a linear array of 784 features for each sample
- The number of points that was used was 5 . Tests were conducted for optimal k but lower values showed better accuracy
- The distance between the points was calculated using the Euclidean distance.
- The model showed around 84 % training accuracy and 76 % and validation accuracy

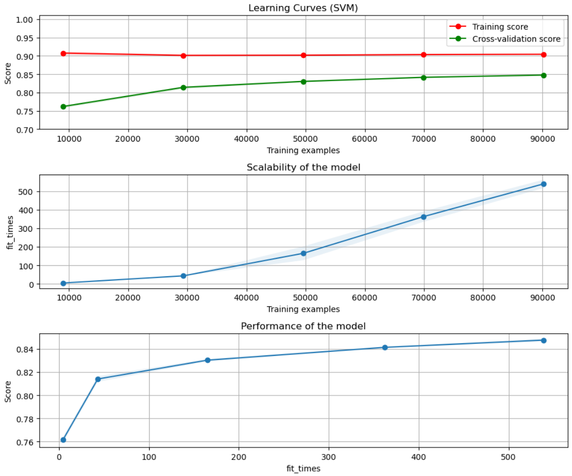
Trial using K nearest neighbour



Trial using SVM

- SVMs take a long time for training for large size data
- PCA was used to reduce the number of components from 784 to 200.
- The model showed around 90.6 % training accuracy and 85 % test accuracy
- We used the rbf kernel for SVM

Trial using SVM



Trial using Voting Classifier

- Voting Classifier function is provided by sklearn where we can combine outputs of models
- We have combined KNN and Random Forest to see if we could achieve some improvement .
- KNN was performing well on some characters like "H" so combining them we tried to see if there ws any improvement in the accuracy
- The testing accuracy was improved to 79.2% compared to the 76 % to the KNN but slightly lower than the Random Forest which was 79.8%

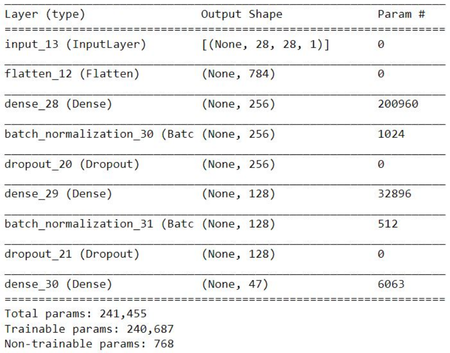
Multi Layer Perceptron for Character Recognition

- Link- https://www.kaggle.com/crawford/emnist/
- Total 131,600 images for 47 classes including 10 digits and 37 classes for lower and upper case alphabets
- Original dimension of each image is 1x784

Multi Layer Perceptron for Character Recognition

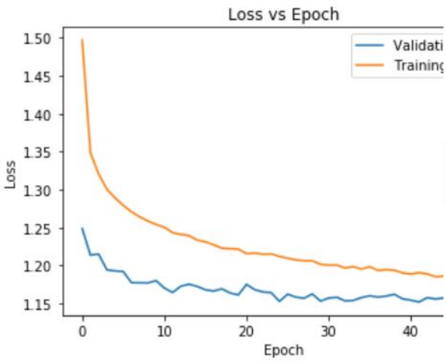
- Training Accuracy is 66.79%
- Validation Accuracy is 68.30%

Multi Layer Perceptron for Character Recognition

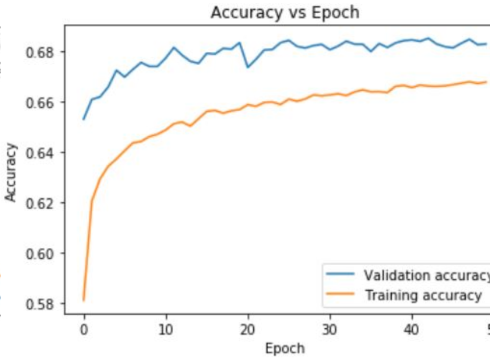


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_13 (InputLayer)        [(None, 28, 28, 1)]       0
_________________________________________________________________
flatten_12 (Flatten)         (None, 784)               0
_________________________________________________________________
dense_28 (Dense)             (None, 256)               200960
_________________________________________________________________
batch_normalization_30 (Batc (None, 256)               1024
_________________________________________________________________
dropout_20 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_29 (Dense)             (None, 128)               32896
_________________________________________________________________
batch_normalization_31 (Batc (None, 128)               512
_________________________________________________________________
dropout_21 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_30 (Dense)             (None, 47)                6063
=================================================================
Total params: 241,455
Trainable params: 240,687
Non-trainable params: 768
```

Training



Loss vs Epoch
Validati
Training
Loss
Epoch
1.50
1.45
1.40
1.35
1.30
1.25
1.20
1.15
0
10
20
30
40



Accuracy vs Epoch
Validation accuracy
Training accuracy
Accuracy
Epoch
0.68
0.66
0.64
0.62
0.60
0.58
0
10
20
30
40

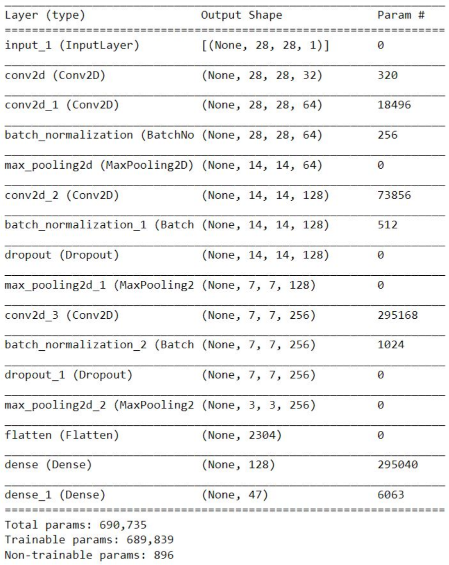
Convolutional Neural Network for Character Recognition

- Link- https://www.kaggle.com/crawford/emnist/
- Total 131,600 images for 47 classes including 10 digits and 37 classes for lower and upper case alphabets
- Original dimension of each image is 1x784

Convolutional Neural Network for Character Recognition

- Convert each image from 1x784 to 28x28x1
- Binarize the image in both train and test data
- X_train = 112799x28x28; X_test = 18799x28x28
- Use one-hot encoding on the labels
    - Y_train = 112799x47; Y_test = 18799x47

CNN Architecture using Keras

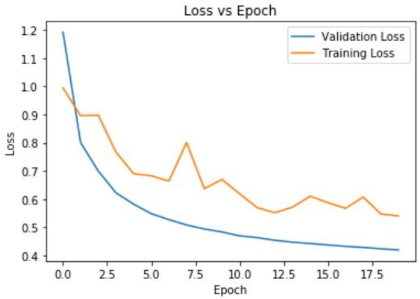


```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         [(None, 28, 28, 1)]       0         
_________________________________________________________________
conv2d (Conv2D)              (None, 28, 28, 32)        320       
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 64)        18496     
_________________________________________________________________
batch_normalization (BatchNo (None, 28, 28, 64)        256       
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 128)       73856     
_________________________________________________________________
batch_normalization_1 (Batch (None, 14, 14, 128)       512       
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 128)       0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 128)         0         
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 7, 7, 256)         295168    
_________________________________________________________________
batch_normalization_2 (Batch (None, 7, 7, 256)         1024      
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 256)         0         
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 256)         0         
_________________________________________________________________
flatten (Flatten)            (None, 2304)              0         
_________________________________________________________________
dense (Dense)                (None, 128)               295040    
_________________________________________________________________
dense_1 (Dense)              (None, 47)                6063      
=================================================================
Total params: 690,735
Trainable params: 689,839
Non-trainable params: 896
```

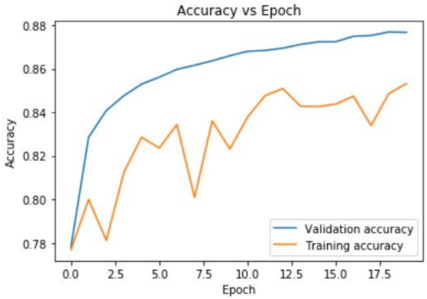
CNN Architecture using Keras

- Training Accuracy is 87.67%
- Validation Accuracy is 85.31%

Training



Loss vs Epoch
Validation Loss
Training Loss
Loss
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch

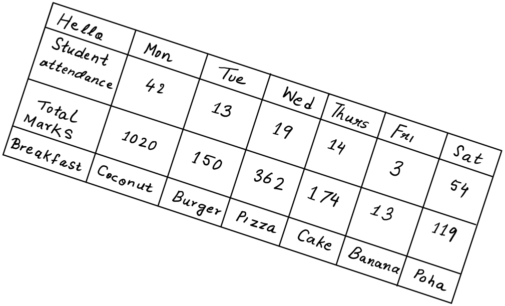


Accuracy vs Epoch
Validation accuracy
Training accuracy
Accuracy
0.88
0.86
0.84
0.82
0.80
0.78
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
Epoch

Final model

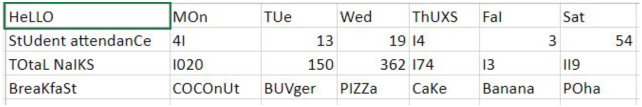
- Convolutional Neural Network is the best classifier for this problem
- So, the final model will use CNN for classification after character extraction step using BFS

Final model



Hello	Mon	Tue	Wed	Thurs	Fri	Sat
Student attendance	42	13	19	14	3	54
Total Marks	1020	150	362	174	13	119
Breakfast	Coconut	Burger	Pizza	Cake	Banana	Poha

Final model



HeLLO	MOn	TUe	Wed	ThUXS	Fal	Sat
StUdent attendanCe	4I	13	19	I4	3	54
TOtaL NaIKS	I020	150	362	I74	I3	II9
BreaKfaSt	COCOnUt	BUVger	PIZZa	CaKe	Banana	POha

The End