B.Tech. Project Report

Handwritten Formulae Detection

under guidance of

Dr. Gaurav Harit
Computer Science and Engineering
Indian Institute of Technology, Jodhpur

Shashwat Kathuria
(B17CS050)
4th Year Undergraduate
Computer Science and Engineering
Indian Institute of Technology, Jodhpur

Satya Prakash Sharma
(B17CS048)
4th Year Undergraduate
Computer Science and Engineering
Indian Institute of Technology, Jodhpur

December 2020

Department of Computer Science and Engineering
Indian Institute of Technology, Jodhpur
December 2020

Certificate

It is certified that the work contained in the project report titled **“Handwritten Formulae Detection”** by **Shashwat Kathuria (B17CS050)** and **Satya Prakash Sharma (B17CS048)** has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Dr. Gaurav Harit
Dept. of Computer Science and Engineering
Indian Institute of Technology, Jodhpur
December 2020

Declaration

I declare that this project submission represents my ideas in my own words. Wherever others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented, fabricated or falsified any idea, data, fact, or source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Shashwat Kathuria
(B17CS050)
4th Year Undergraduate
Computer Science and Engineering
Indian Institute of Technology, Jodhpur
December 2020

Satya Prakash Sharma
(B17CS048)
4th Year Undergraduate
Computer Science and Engineering
Indian Institute of Technology, Jodhpur
December 2020

Acknowledgement

I would like to express my sincere gratitude to my project supervisor **Dr. Gaurav Harit** for helping me in taking such a crucial project, and giving valuable suggestions, comments and proper guidance throughout the course of the project and giving us the exposure to use the Nvidia GPU in our college.

I thank distributors from **Nvidia, Keras, TensorFlow, OpenCV and Tkinter** for making the libraries for this project publicly available. Also thanks to the distributors of **ICDAR 2017 dataset, IAM dataset and CROHME dataset** for providing the necessary real world data involving computerized page images from ICDAR with the ground truth involving page objects, and IAM and CROHME dataset for handwritten text and handwritten formulae.

Abstract

As the advent of Document Image Understanding (DUI) increases, there is attention from document analysis and recognition communities along with database and information extraction communities to detect and understand the structure of a page containing objects like formulae, tables, figures, etc.

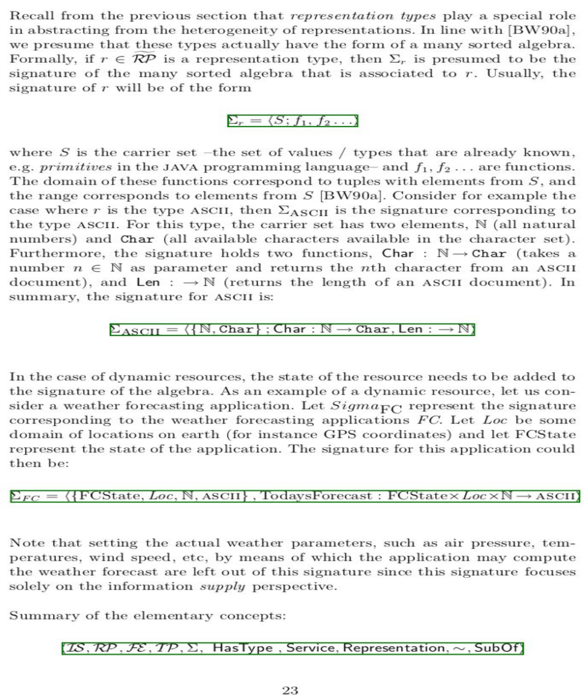
In this project, we implement a model to detect handwritten formulae extending from a computerized formulae detection model, which would be able to detect the handwritten formulae in real world text data and mathematical formulae.

The route we take for implementing this model is by implementing a RCNN model for detection of computerized text formulae detection, after which we generate a synthetic dataset for generating handwritten pages with different types of handwritings along with various handwritten formulae in different types of alignments as well as in between text.

Table of Contents

1. Introduction

There has been increasing attention to detect and understand the structure of a page containing objects like formulae, tables, figures, etc from document analysis and recognition communities along with database and information extraction communities. In this project, we will be focussing on the formulae detection in pages, both computerized and handwritten.



Recall from the previous section that *representation types* play a special role in abstracting from the heterogeneity of representations. In line with [BW90a], we presume that these types actually have the form of a many sorted algebra. Formally, if $r \in \widetilde{\mathcal{RP}}$ is a representation type, then Σ_r is presumed to be the signature of the many sorted algebra that is associated to r. Usually, the signature of r will be of the form

$$\Sigma_r = \langle S; f_1, f_2 \ldots \rangle$$

where S is the carrier set –the set of values / types that are already known, e.g. *primitives* in the JAVA programming language– and $f_1, f_2 \ldots$ are functions. The domain of these functions correspond to tuples with elements from S, and the range corresponds to elements from S [BW90a]. Consider for example the case where r is the type ASCII, then Σ_{ASCII} is the signature corresponding to the type ASCII. For this type, the carrier set has two elements, $\mathbb{N}$ (all natural numbers) and Char (all available characters available in the character set). Furthermore, the signature holds two functions, $\mathsf{Char} : \mathbb{N} \rightarrow \mathsf{Char}$ (takes a number $n \in \mathbb{N}$ as parameter and returns the nth character from an ASCII document), and $\mathsf{Len} : \rightarrow \mathbb{N}$ (returns the length of an ASCII document). In summary, the signature for ASCII is:

$$\Sigma_{\text{ASCII}} = \langle \{\mathbb{N}, \mathsf{Char}\}; \mathsf{Char} : \mathbb{N} \rightarrow \mathsf{Char}, \mathsf{Len} : \rightarrow \mathbb{N} \rangle$$

In the case of dynamic resources, the state of the resource needs to be added to the signature of the algebra. As an example of a dynamic resource, let us consider a weather forecasting application. Let $Sigma_{\text{FC}}$ represent the signature corresponding to the weather forecasting applications *FC*. Let *Loc* be some domain of locations on earth (for instance GPS coordinates) and let FCState represent the state of the application. The signature for this application could then be:

$$\Sigma_{FC} = \langle \{\text{FCState}, Loc, \mathbb{N}, \text{ASCII}\}, \text{TodaysForecast} : \text{FCState} \times Loc \times \mathbb{N} \rightarrow \text{ASCII} \rangle$$

Note that setting the actual weather parameters, such as air pressure, temperatures, wind speed, etc, by means of which the application may compute the weather forecast are left out of this signature since this signature focuses solely on the information *supply* perspective.

Summary of the elementary concepts:

$$\langle \mathcal{IS}, \mathcal{RP}, \mathcal{FE}, \mathcal{TP}, \Sigma, \mathsf{HasType}, \mathsf{Service}, \mathsf{Representation}, \sim, \mathsf{SubOf} \rangle$$

23

Our aim is to detect various types of formulas which can be present in images taken from PDF documents, like research papers. Some of the challenges that are posed while designing such a model are:

- ➔ formulas can be of various types
- ➔ can have different types of symbols inside them, with some having superscript and subscript
- ➔ some having very different forms like some present inside a table, some written inside matrices, etc.

We aim to build a model to predict such formulas in PDFs and along with analyzing the working and results obtained.

2. Implementation Steps

---

1. Study about RCNN Model and Page Object Detection

We study about the models implemented by various teams that participated in the ICDAR 2017 Page Object Detection Competition, and in that competition the most commonly used model was RCNN. We then further dive deep into the details of the RCNN model and study it.

2. Computerized Formulae Detection Model

We implement a RCNN model in our Nvidia GPU which can detect the computerized formulae given in the ICDAR 2017 dataset. The model is fine tuned to the details observed generally in formulae keeping all aspects in consideration.

3. Study about Generation of Synthetic Dataset

Our main aim is to detect handwritten formulae and for that we would need a database which can be sufficient to take into consideration the various types of common handwritings and mathematical formulae. We research about GANs but they are available only for computerized text to handwritten text data generation, not for generating mathematical formulae, so we design a heuristic which can take handwritten sentences, words, bits of text along with handwritten formulae and generate a handwritten document along with the ground truth.

4. Generate Handwritten Formulae Dataset

We design a heuristic which can randomly generate pages, by adding the handwritten words, sentences, bits of text into a plain canvas and also add formulae in between text, along with different types of alignment of formulae to give the effect of a real handwritten page, along with the ground truth. All of the pages involve common handwritings and formulae with the help of IAM and CROHME dataset.

5. Handwritten Formulae Detection Model

Finally, we fine tune the existing models to accommodate the new synthetic dataset generated, and then predict handwritten formulae with the help of the fine tuned model.

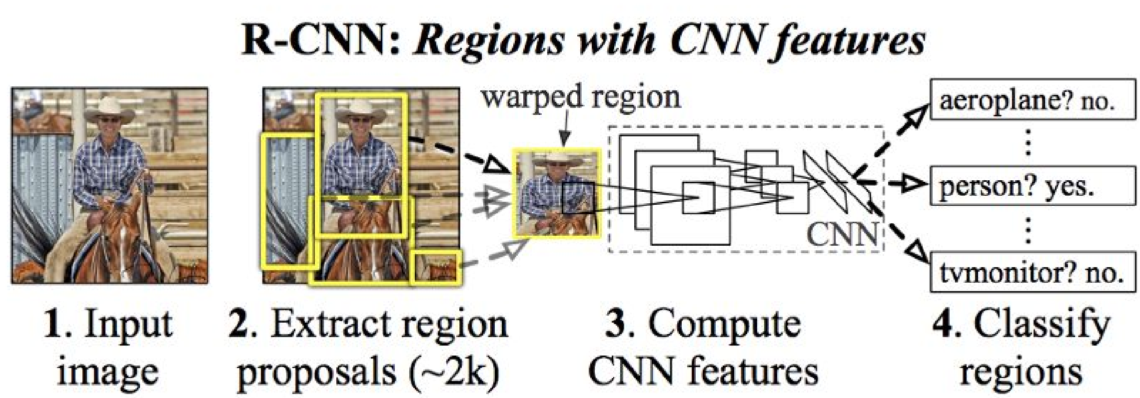
3. Background

3.1 Previous Work

In the ICDAR 2017 Page Object Detection Competition paper, we have seen that there are a lot of teams that have worked on building a Page Object Detection model, and most of them relied on the CNN model, with fine tuning required everywhere, also some have used COCO-TEXT dataset to enhance the performance of detection of formulas for fine tuning. We go forward with implementing the most promising of them, which is RCNN architecture with VGG-16.

3.2 RCNN Model

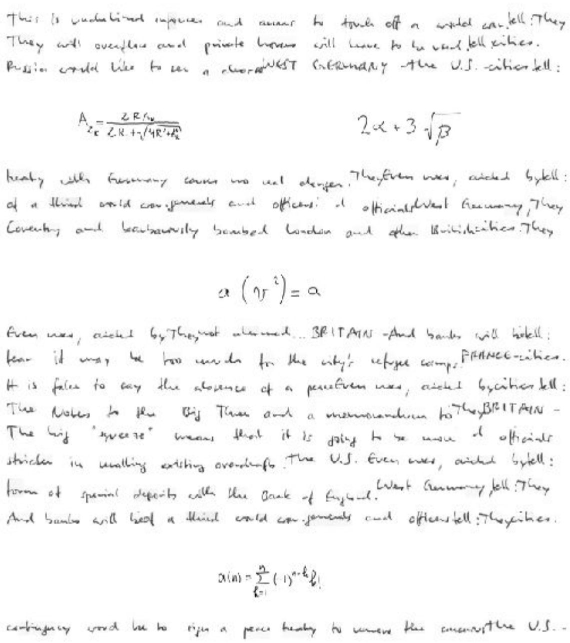
RCNN models work by finding the region of interest in the image, which we get using selective search and are called region proposals, after which the CNN features are extracted from the region proposals and then classification of the objects is performed on the extracted features. This model gives good results on specific tasks and we use it to implement our project.



3.3 Synthetic Dataset

In order to fine tune our computerized text formulae model for the handwritten formulae, we need to have a database which is somewhat in a common manner as real world handwritten pages. We have access to ICDAR 2017 POD dataset for the computerized text, but not a handwritten dataset, which raises the need for us to generate such a dataset. We create this dataset artificially, i.e., in a synthetic way by capturing bits of text, sentences, words with handwritten formulae in a page such that each of it looks like a real page. Our requirement here is syntactical, not semantical, as we need to generate a structure of a page similar to a real world page without the meaning in context.

We design a heuristic to randomly add handwritten bits of text, words and sentences along with handwritten formulae from the IAM and CROHME dataset in different types of alignments and in between text. We have a total of 25 different common types of handwritings and we generate a total of 1000 pages using one common writing in an individual page. The number of paragraphs, the order, the alignments, everything is randomized to give a real world sense of the data generated.



This is [illegible] and [illegible] to touch off a [illegible] tell: They
They will overflow and private [illegible] will have to be [illegible] tell cities.
Russia would like to see a [illegible] WEST GERMANY - the U.S. - cities tell:

$$A = \frac{2R}{2R + \sqrt{4R^2 + R^2}} \qquad 2\alpha + 3\sqrt{\beta}$$

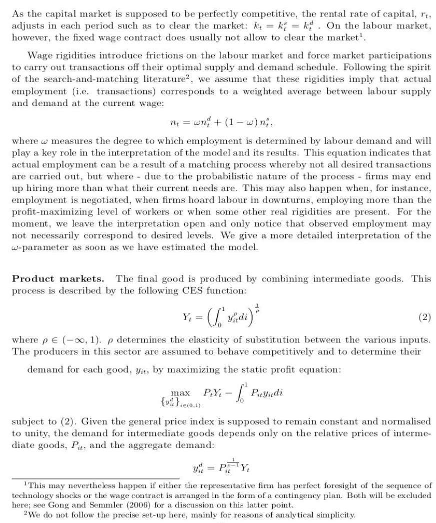
treaty with Germany [illegible] no real danger. They Even was, aided by tell:
of a third world [illegible] and officers: [illegible] officials West Germany, They
Coventry and barbarously bombed London and other British cities. They

$$a\,(\nu^2) = a$$

Even was, aided by They not [illegible] ... BRITAIN - And banks will tell:
fear it may be too [illegible] for the city's refugee camps. FRANCE - cities.
It is false to say the absence of a peace Even was, aided by cities tell:
The Notes to the Big Three and a memorandum to They BRITAIN -
The big "squeeze" means that it is going to be more [illegible] officials
stricter in [illegible] existing overdrafts. The U.S. Even was, aided by tell:
form of special deposits with the Bank of England. West Germany, tell: They
And banks will [illegible] a third world [illegible] and officers tell: They cities.

$$x(n) = \sum_{k=1}^{n} (-1)^{n-k} k!$$

contingency would be to sign a peace treaty to [illegible] the [illegible] the U.S. -



As the capital market is supposed to be perfectly competitive, the rental rate of capital, r_t, adjusts in each period such as to clear the market: $k_t = k_t^s = k_t^d$. On the labour market, however, the fixed wage contract does usually not allow to clear the market[1].

Wage rigidities introduce frictions on the labour market and force market participations to carry out transactions off their optimal supply and demand schedule. Following the spirit of the search-and-matching literature[2], we assume that these rigidities imply that actual employment (i.e. transactions) corresponds to a weighted average between labour supply and demand at the current wage:

$$n_t = \omega n_t^d + (1 - \omega)\, n_t^s,$$

where ω measures the degree to which employment is determined by labour demand and will play a key role in the interpretation of the model and its results. This equation indicates that actual employment can be a result of a matching process whereby not all desired transactions are carried out, but where - due to the probabilistic nature of the process - firms may end up hiring more than what their current needs are. This may also happen when, for instance, employment is negotiated, when firms hoard labour in downturns, employing more than the profit-maximizing level of workers or when some other real rigidities are present. For the moment, we leave the interpretation open and only notice that observed employment may not necessarily correspond to desired levels. We give a more detailed interpretation of the ω-parameter as soon as we have estimated the model.

Product markets. The final good is produced by combining intermediate goods. This process is described by the following CES function:

$$Y_t = \left(\int_0^1 y_{it}^{\rho} di \right)^{\frac{1}{\rho}} \tag{2}$$

where $\rho \in (-\infty, 1)$. ρ determines the elasticity of substitution between the various inputs. The producers in this sector are assumed to behave competitively and to determine their

demand for each good, y_{it}, by maximizing the static profit equation:

$$\max_{\{y_{it}^d\}_{i \in (0,1)}} P_t Y_t - \int_0^1 P_{it} y_{it} di$$

subject to (2). Given the general price index is supposed to remain constant and normalised to unity, the demand for intermediate goods depends only on the relative prices of intermediate goods, P_{it}, and the aggregate demand:

$$y_{it}^d = P_{it}^{\frac{1}{\rho - 1}} Y_t$$

[1] This may nevertheless happen if either the representative firm has perfect foresight of the sequence of technology shocks or the wage contract is arranged in the form of a contingency plan. Both will be excluded here; see Gong and Semmler (2006) for a discussion on this latter point.

[2] We do not follow the precise set-up here, mainly for reasons of analytical simplicity.

5

3.4 Requirements

This project has been implemented on IITJ Nvidia GPU and requires python3, tensorflow, keras, opencv, matplotlib, numpy, tkinter, etc to run the program and train the model. Additional requirements are the datasets that are linked in the references along with a basic context of the model, previous work and the references.

4. Implementation Details & Methodology

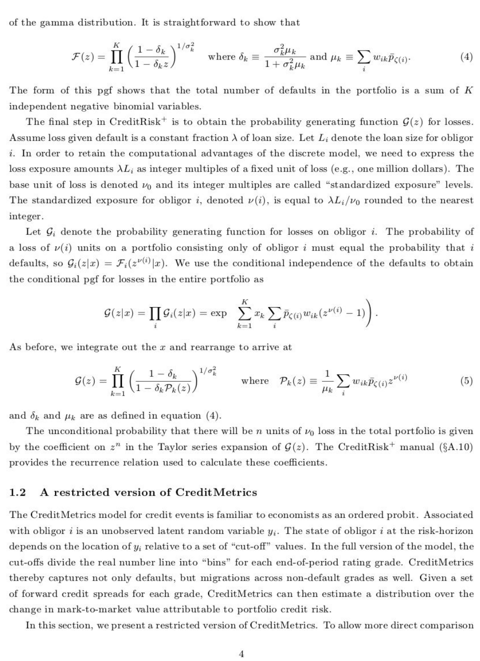
4.1 Dataset & Preprocessing

We make use of ICDAR, IAM and CROHME dataset in our project, the details of which are given below.

4.1.1 Normal

The ICDAR 2017 Page Object Detection dataset contains a total of 1600 images which have tables, formulas and figures annotations. There are additional 800 images but they are not annotated because the dataset is taken from a competition. Out of 1600 images, about 910 images contain formulas and the number for figures and tables is 904 and 549.

We convert the bitmap images to png images and then we parse the coordinates of the formula regions by parsing the annotation xml file.



of the gamma distribution. It is straightforward to show that

$$\mathcal{F}(z) = \prod_{k=1}^{K} \left(\frac{1-\delta_k}{1-\delta_k z} \right)^{1/\sigma_k^2} \quad \text{where } \delta_k \equiv \frac{\sigma_k^2 \mu_k}{1+\sigma_k^2 \mu_k} \text{ and } \mu_k \equiv \sum_i w_{ik} \bar{p}_{\zeta(i)}. \tag{4}$$

The form of this pgf shows that the total number of defaults in the portfolio is a sum of K independent negative binomial variables.

The final step in CreditRisk$^+$ is to obtain the probability generating function $\mathcal{G}(z)$ for losses. Assume loss given default is a constant fraction λ of loan size. Let L_i denote the loan size for obligor i. In order to retain the computational advantages of the discrete model, we need to express the loss exposure amounts λL_i as integer multiples of a fixed unit of loss (e.g., one million dollars). The base unit of loss is denoted ν_0 and its integer multiples are called "standardized exposure" levels. The standardized exposure for obligor i, denoted $\nu(i)$, is equal to $\lambda L_i/\nu_0$ rounded to the nearest integer.

Let $\mathcal{G}_i$ denote the probability generating function for losses on obligor i. The probability of a loss of $\nu(i)$ units on a portfolio consisting only of obligor i must equal the probability that i defaults, so $\mathcal{G}_i(z|x) = \mathcal{F}_i(z^{\nu(i)}|x)$. We use the conditional independence of the defaults to obtain the conditional pgf for losses in the entire portfolio as

$$\mathcal{G}(z|x) = \prod_i \mathcal{G}_i(z|x) = \exp \quad \sum_{k=1}^{K} x_k \sum_i \bar{p}_{\zeta(i)} w_{ik} (z^{\nu(i)} - 1) \Big).$$

As before, we integrate out the x and rearrange to arrive at

$$\mathcal{G}(z) = \prod_{k=1}^{K} \left(\frac{1-\delta_k}{1-\delta_k \mathcal{P}_k(z)} \right)^{1/\sigma_k^2} \quad \text{where} \quad \mathcal{P}_k(z) \equiv \frac{1}{\mu_k} \sum_i w_{ik} \bar{p}_{\zeta(i)} z^{\nu(i)} \tag{5}$$

and δ_k and μ_k are as defined in equation (4).

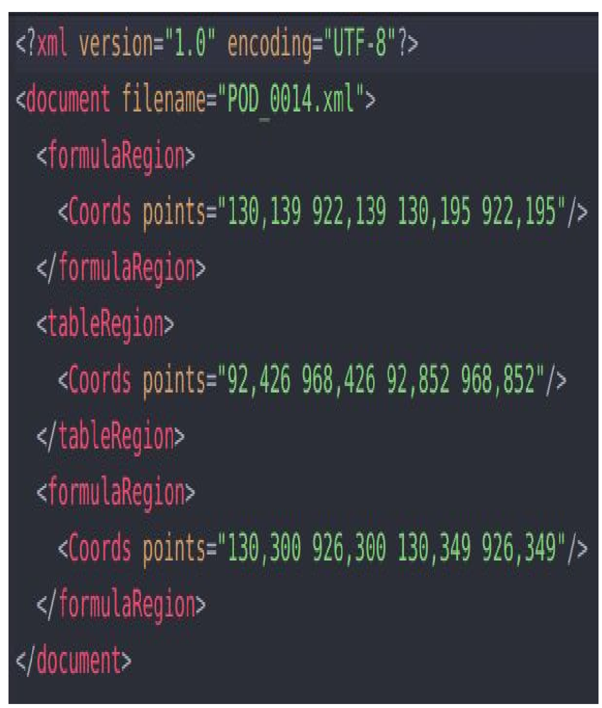
The unconditional probability that there will be n units of ν_0 loss in the total portfolio is given by the coefficient on z^n in the Taylor series expansion of $\mathcal{G}(z)$. The CreditRisk$^+$ manual (§A.10) provides the recurrence relation used to calculate these coefficients.

1.2 A restricted version of CreditMetrics

The CreditMetrics model for credit events is familiar to economists as an ordered probit. Associated with obligor i is an unobserved latent random variable y_i. The state of obligor i at the risk-horizon depends on the location of y_i relative to a set of "cut-off" values. In the full version of the model, the cut-offs divide the real number line into "bins" for each end-of-period rating grade. CreditMetrics thereby captures not only defaults, but migrations across non-default grades as well. Given a set of forward credit spreads for each grade, CreditMetrics can then estimate a distribution over the change in mark-to-market value attributable to portfolio credit risk.

In this section, we present a restricted version of CreditMetrics. To allow more direct comparison

4



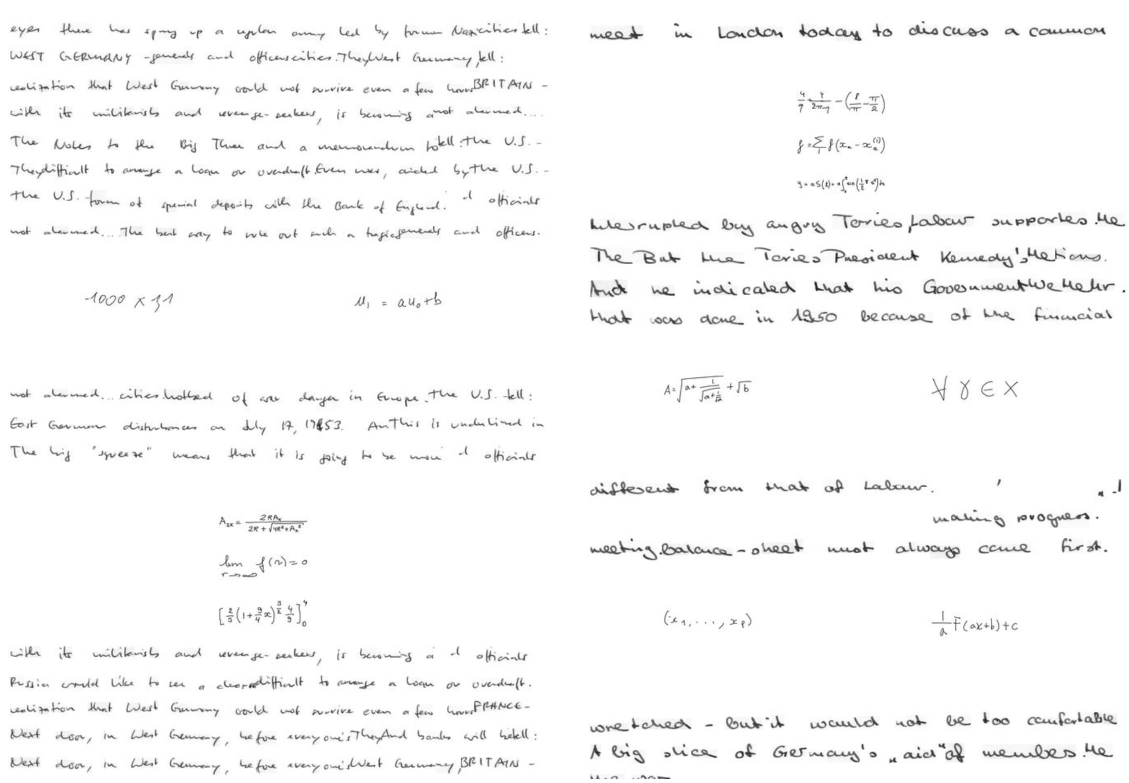
4.1.2 Synthetic

We have used IAM dataset having handwriting of various common types and CROHME dataset having mathematical handwritten formulae. The IAM dataset consists of 25 different types of common handwritings and the CROHME dataset consists of about 10500 formulas.

We have designed a heuristic to generate a synthetic dataset consisting of 1000 handwritten pages in a randomized manner having the following features:

- ➔ Having text and formulae regions with different types of handwritings
- ➔ Formulae in separate regions and also in between text in different types of alignments

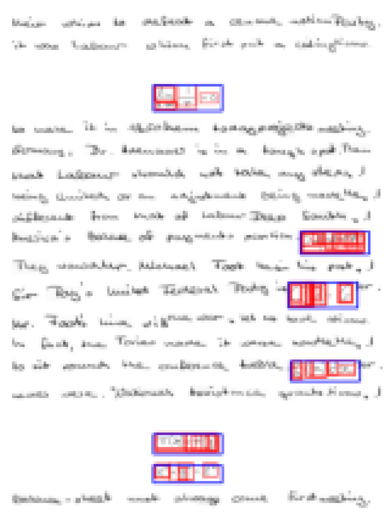
We parse the coordinates of the formula regions by parsing the ground truth of the generated images, which is written in a format of x1, y1, x2, y2 of the bounding box of the formulas in a page.



4.2 Training & Testing

For giving training input into our model, we use selective search to obtain region proposals of the images, after which we compute Intersection over Union (IoU) of the proposed region with any of the annotated formula region, and then add a corresponding label of formula or not a formula based on the value of the IoU. Because there are many white spaces inside some formulas, the IoU is set to >= 0.6 for formulae regions, and <= 0.05 for non formulae regions.

The splitting ratio we use for training and testing is 80:20. Also, to make our inputs unbiased we limit the total number of positive and negative samples entering the model for each page, to an equal of 20~30 per page, because otherwise a huge number of negative samples can enter the model, because there is more region which contains text than formula.



4.3 Model Details

The model implemented is Regions with Convolutional Neural Networks (RCNN) architecture with VGG-16. We use VGG-16 because it performs well with specific tasks, along with two unit softmax layers, because our prediction is binary in nature. We make some modifications on the images before passing to the model like horizontal flip, vertical flip and rotation to increase the dataset and observe that the prediction accuracy of the model increases.

Following is the layered model summary of the model implemented:

```
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 8194
=================================================================
Total params: 134,268,738
Trainable params: 126,633,474
Non-trainable params: 7,635,264
_________________________________________________________________
```

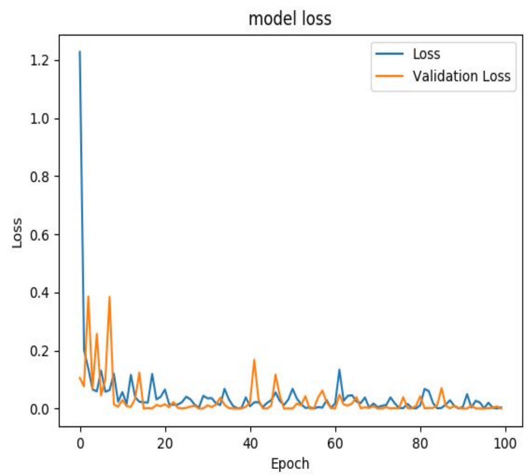
4.4 Training Strategy & Loss Function

We use Adam for optimizer because it can be used instead of the stochastic gradient descent procedure to update network weights iteratively based on training data. Because it is efficient in use and works well with large amounts of data, we make use of it. The learning rate is set to 0.001 .

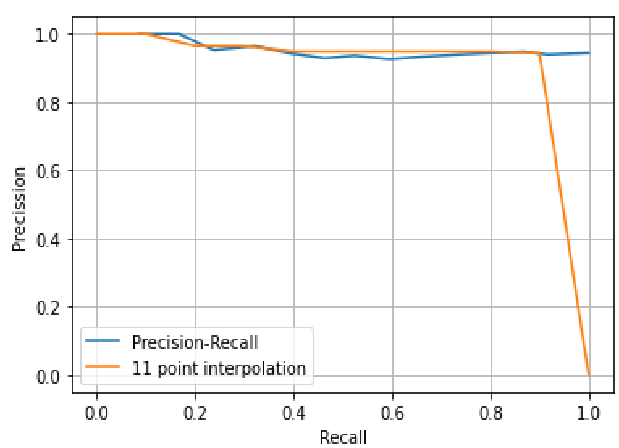
For loss function we use Categorical Cross Entropy because our prediction is categorical and binary, whether or not there is a formula (one hot encoding). We keep track of loss and accuracy after each epoch, for 1000 epochs with 10 steps per epoch and update the model parameters only if the loss value decreases.

For prediction, we get proposed regions using selective search and then obtain the model prediction, which if >= 0.75, we consider the proposed region as having formula, otherwise not.

4.5 Results



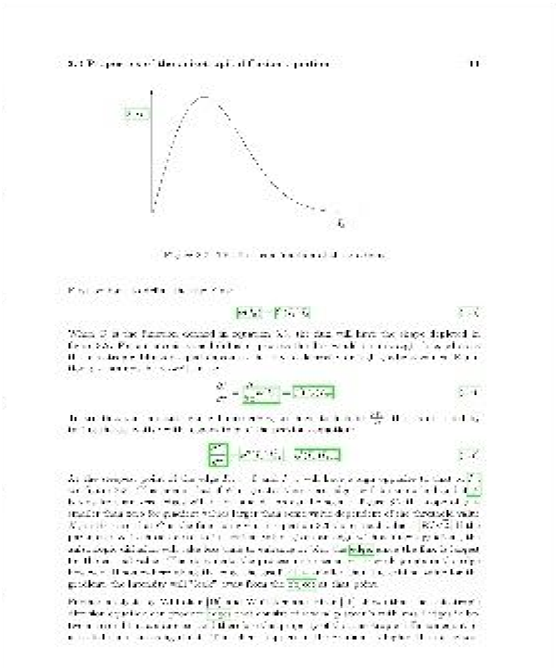
In the above diagram, the loss v/s epoch is plotted for each epoch during training of our model and presented. We can see that after a lot of epochs, loss is almost constant. We stop training after 100 epochs and during each epoch the model is updated only if the loss value decreases. We keep track of loss and accuracy after each epoch, for 1000 epochs with 10 steps per epoch and update the model parameters only if the loss value decreases.

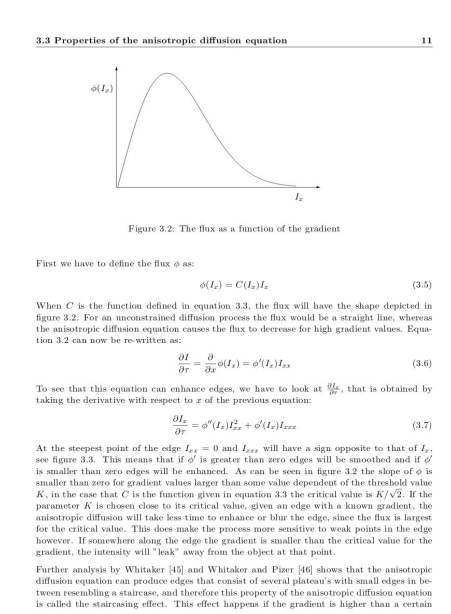


In the above diagram, the average precision of our model is calculated using 11 point interpolation and the result is **87.38%** . The precision and recall values are calculated using true positives, false positives and false negatives and the method for average precision is 11 point interpolation technique.

5. Sample Outputs

5.1 Computerized Formulae Detection





3.3 Properties of the anisotropic diffusion equation 11

Figure 3.2: The flux as a function of the gradient

First we have to define the flux ϕ as:

$$\phi(I_x) = C(I_x)I_x \tag{3.5}$$

When C is the function defined in equation 3.3, the flux will have the shape depicted in figure 3.2. For an unconstrained diffusion process the flux would be a straight line, whereas the anisotropic diffusion equation causes the flux to decrease for high gradient values. Equation 3.2 can now be re-written as:

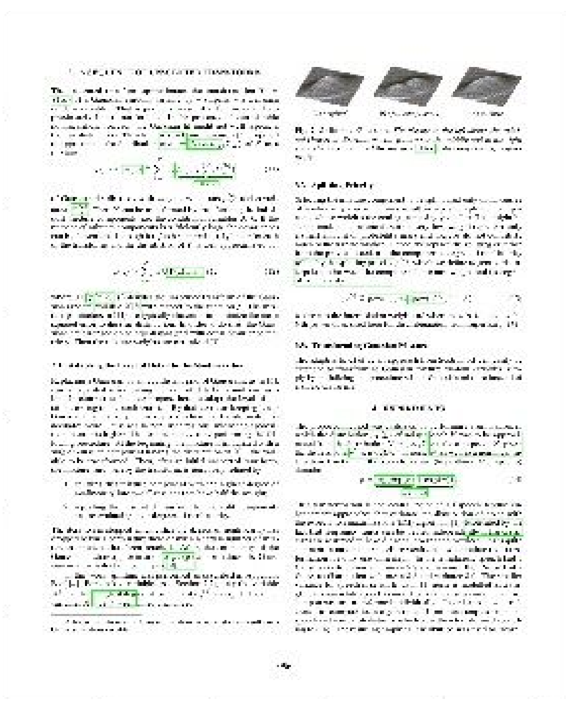
$$\frac{\partial I}{\partial \tau} = \frac{\partial}{\partial x}\phi(I_x) = \phi'(I_x)I_{xx} \tag{3.6}$$

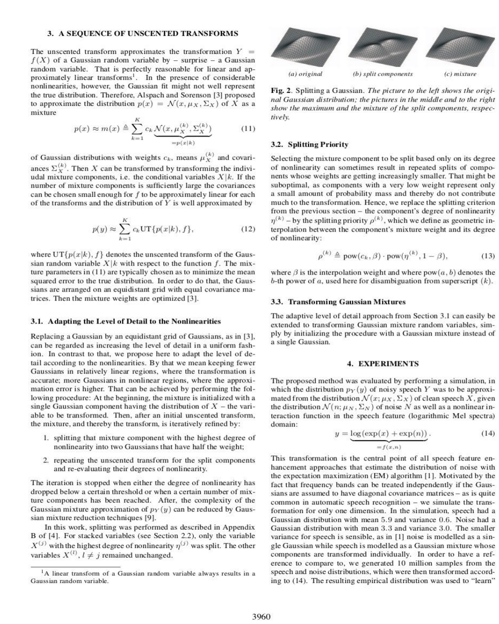
To see that this equation can enhance edges, we have to look at $\frac{\partial I_x}{\partial \tau}$, that is obtained by taking the derivative with respect to x of the previous equation:

$$\frac{\partial I_x}{\partial \tau} = \phi''(I_x)I_{xx}^2 + \phi'(I_x)I_{xxx} \tag{3.7}$$

At the steepest point of the edge $I_{xx} = 0$ and I_{xxx} will have a sign opposite to that of I_x, see figure 3.3. This means that if ϕ' is greater than zero edges will be smoothed and if ϕ' is smaller than zero edges will be enhanced. As can be seen in figure 3.2 the slope of ϕ is smaller than zero for gradient values larger than some value dependent of the threshold value K, in the case that C is the function given in equation 3.3 the critical value is $K/\sqrt{2}$. If the parameter K is chosen close to its critical value, given an edge with a known gradient, the anisotropic diffusion will take less time to enhance or blur the edge, since the flux is largest for the critical value. This does make the process more sensitive to weak points in the edge however. If somewhere along the edge the gradient is smaller than the critical value for the gradient, the intensity will "leak" away from the object at that point.

Further analysis by Whitaker [45] and Whitaker and Pizer [46] shows that the anisotropic diffusion equation can produce edges that consist of several plateau's with small edges in between resembling a staircase, and therefore this property of the anisotropic diffusion equation is called the staircasing effect. This effect happens if the gradient is higher than a certain





3. A SEQUENCE OF UNSCENTED TRANSFORMS

The unscented transform approximates the transformation $Y = f(X)$ of a Gaussian random variable by – surprise – a Gaussian random variable. That is perfectly reasonable for linear and approximately linear transforms[1]. In the presence of considerable nonlinearities, however, the Gaussian fit might not well represent the true distribution. Therefore, Alspach and Sorenson [3] proposed to approximate the distribution $p(x) = \mathcal{N}(x, \mu_X, \Sigma_X)$ of X as a mixture

$$p(x) \approx m(x) \triangleq \sum_{k=1}^{K} c_k \underbrace{\mathcal{N}(x, \mu_X^{(k)}, \Sigma_X^{(k)})}_{=p(x|k)} \tag{11}$$

of Gaussian distributions with weights c_k, means $\mu_X^{(k)}$ and covariances $\Sigma_X^{(k)}$. Then X can be transformed by transforming the individual mixture components, i.e. the conditional variables $X|k$. If the number of mixture components is sufficiently large the covariances can be chosen small enough for f to be approximately linear for each of the transforms and the distribution of Y is well approximated by

$$p(y) \approx \sum_{k=1}^{K} c_k \mathrm{UT}\{p(x|k), f\}, \tag{12}$$

where $\mathrm{UT}\{p(x|k), f\}$ denotes the unscented transform of the Gaussian random variable $X|k$ with respect to the function f. The mixture parameters in (11) are typically chosen as to minimize the mean squared error to the true distribution. In order to do that, the Gaussians are arranged on an equidistant grid with equal covariance matrices. Then the mixture weights are optimized [3].

3.1. Adapting the Level of Detail to the Nonlinearities

Replacing a Gaussian by an equidistant grid of Gaussians, as in [3], can be regarded as increasing the level of detail in a uniform fashion. In contrast to that, we propose here to adapt the level of detail according to the nonlinearities. By that we mean keeping fewer Gaussians in relatively linear regions, where the transformation is accurate; more Gaussians in nonlinear regions, where the approximation error is higher. That can be achieved by performing the following procedure: At the beginning, the mixture is initialized with a single Gaussian component having the distribution of X – the variable to be transformed. Then, after an initial unscented transform, the mixture, and thereby the transform, is iteratively refined by:

1. splitting that mixture component with the highest degree of nonlinearity into two Gaussians that have half the weight;
2. repeating the unscented transform for the split components and re-evaluating their degrees of nonlinearity.

The iteration is stopped when either the degree of nonlinearity has dropped below a certain threshold or when a certain number of mixture components has been reached. After, the complexity of the Gaussian mixture approximation of $p_Y(y)$ can be reduced by Gaussian mixture reduction techniques [9].

In this work, splitting was performed as described in Appendix B of [4]. For stacked variables (see Section 2.2), only the variable $X^{(j)}$ with the highest degree of nonlinearity $\eta^{(j)}$ was split. The other variables $X^{(l)}$, $l \neq j$ remained unchanged.

[1]A linear transform of a Gaussian random variable always results in a Gaussian random variable.

(a) original (b) split components (c) mixture

Fig. 2. Splitting a Gaussian. *The picture to the left shows the original Gaussian distribution; the pictures in the middle and to the right show the maximum and the mixture of the split components, respectively.*

3.2. Splitting Priority

Selecting the mixture component to be split based only on its degree of nonlinearity can sometimes result in repeated splits of components whose weights are getting increasingly smaller. That might be suboptimal, as components with a very low weight represent only a small amount of probability mass and thereby do not contribute much to the transformation. Hence, we replace the splitting criterion from the previous section – the component's degree of nonlinearity $\eta^{(k)}$ – by the splitting priority $\rho^{(k)}$, which we define as geometric interpolation between the component's mixture weight and its degree of nonlinearity:

$$\rho^{(k)} \triangleq \mathrm{pow}(c_k, \beta) \cdot \mathrm{pow}(\eta^{(k)}, 1-\beta), \tag{13}$$

where β is the interpolation weight and where $\mathrm{pow}(a, b)$ denotes the b-th power of a, used here for disambiguation from superscript (k).

3.3. Transforming Gaussian Mixtures

The adaptive level of detail approach from Section 3.1 can easily be extended to transforming Gaussian mixture random variables, simply by initializing the procedure with a Gaussian mixture instead of a single Gaussian.

4. EXPERIMENTS

The proposed method was evaluated by performing a simulation, in which the distribution $p_Y(y)$ of noisy speech Y was to be approximated from the distribution $\mathcal{N}(x; \mu_X, \Sigma_X)$ of clean speech X, given the distribution $\mathcal{N}(n; \mu_N, \Sigma_N)$ of noise N as well as a nonlinear interaction function in the speech feature (logarithmic Mel spectra) domain:

$$y = \underbrace{\log(\exp(x) + \exp(n))}_{=f(x,n)}. \tag{14}$$

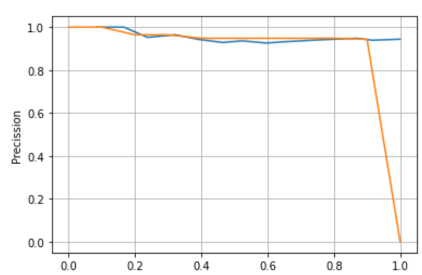
This transformation is the central point of all speech feature enhancement approaches that estimate the distribution of noise with the expectation maximization (EM) algorithm [1]. Motivated by the fact that frequency bands can be treated independently if the Gaussians are assumed to have diagonal covariance matrices – as is quite common in automatic speech recognition – we simulate the transformation for only one dimension. In the simulation, speech had a Gaussian distribution with mean 5.9 and variance 0.6. Noise had a Gaussian distribution with mean 3.3 and variance 3.0. The smaller variance for speech is sensible, as in [1] noise is modelled as a single Gaussian while speech is modelled as a Gaussian mixture whose components are transformed individually. In order to have a reference to compare to, we generated 10 million samples from the speech and noise distributions, which were then transformed according to (14). The resulting empirical distribution was used to "learn"

3960

5.2 Handwritten Formulae Detection

Conclusion

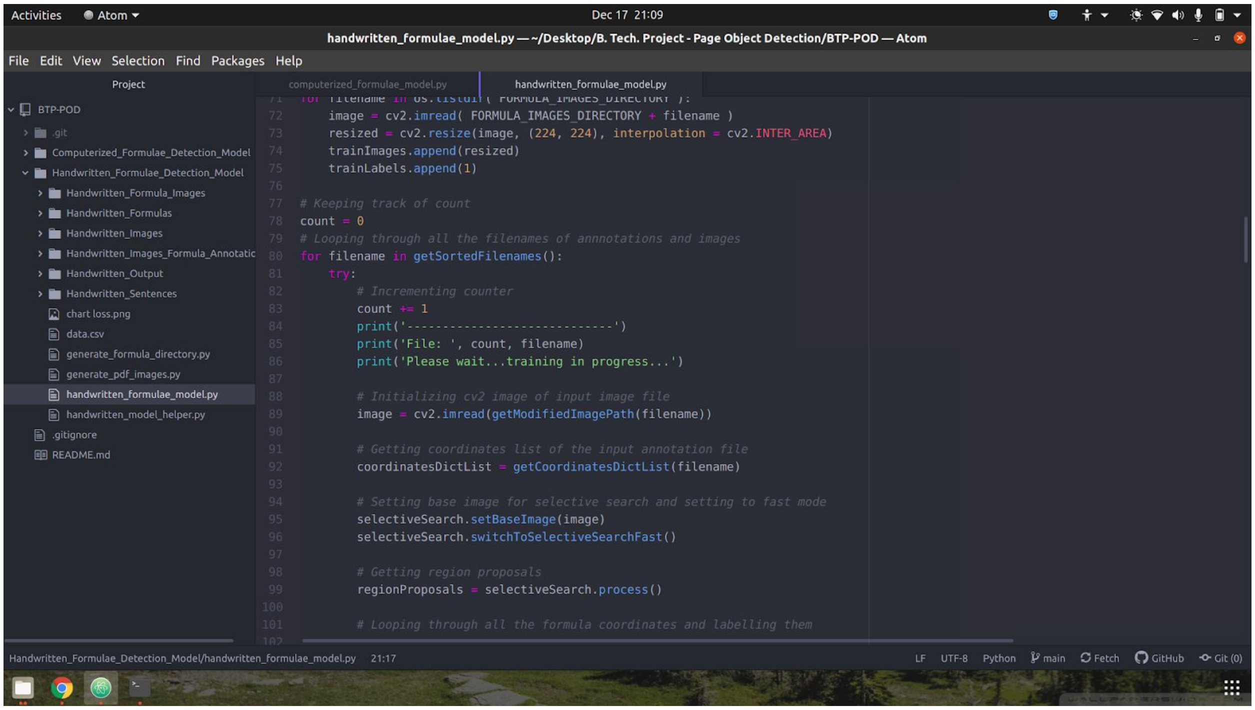
We have been able to implement the ‘Handwritten Formulae Detection Model’.
Average Precision (11 Point Interpolation): 87.38%

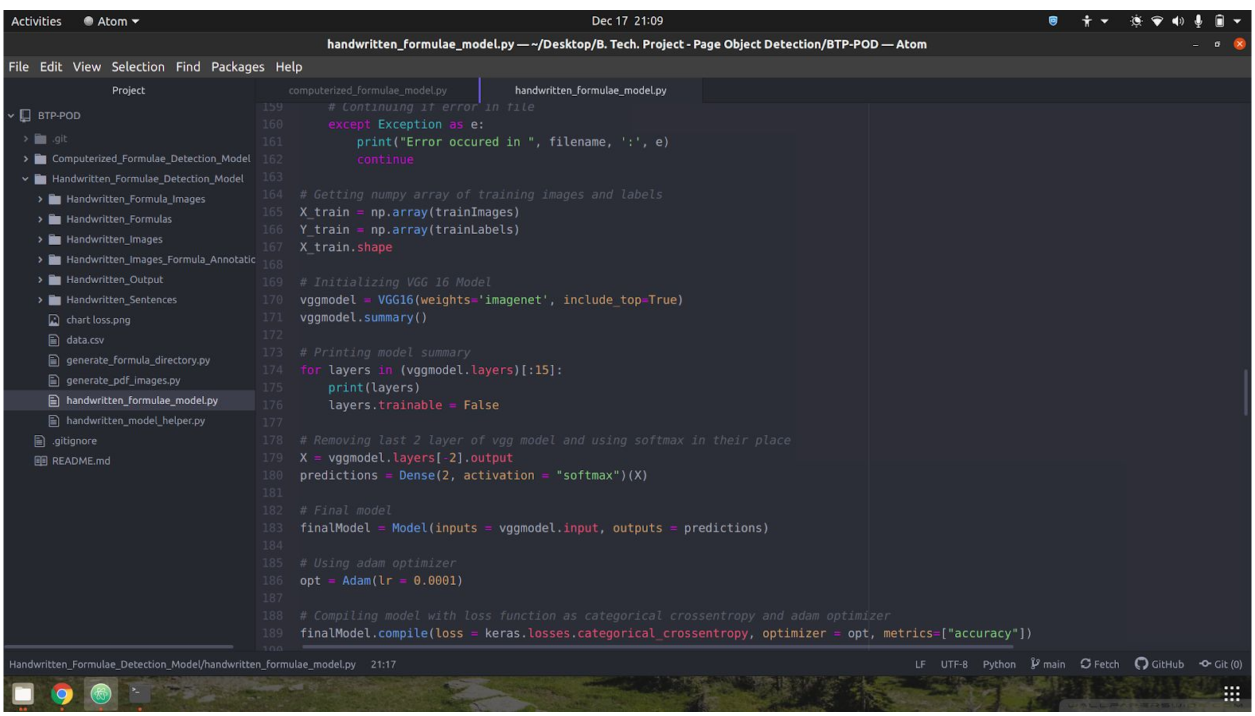


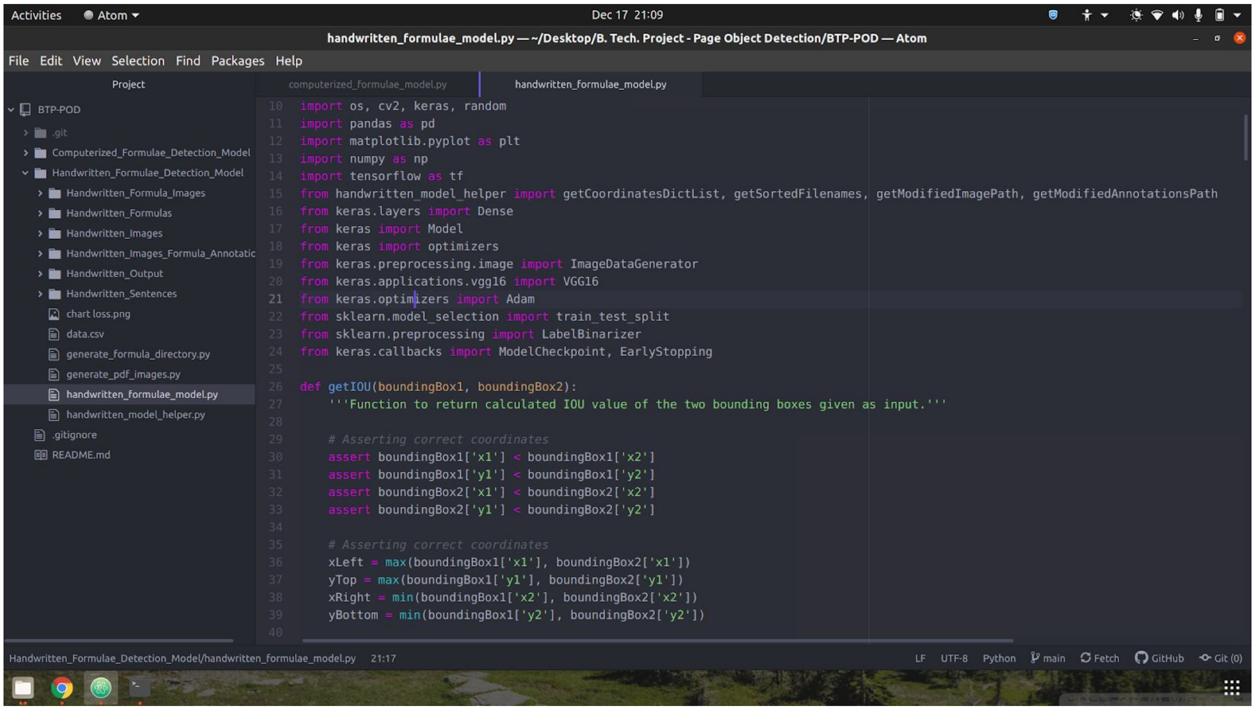
The code is available at:
https://drive.google.com/drive/folders/1melLTux5SQs13MkyC_tkehlHKo5U3947?usp=sharing

Code Snippets

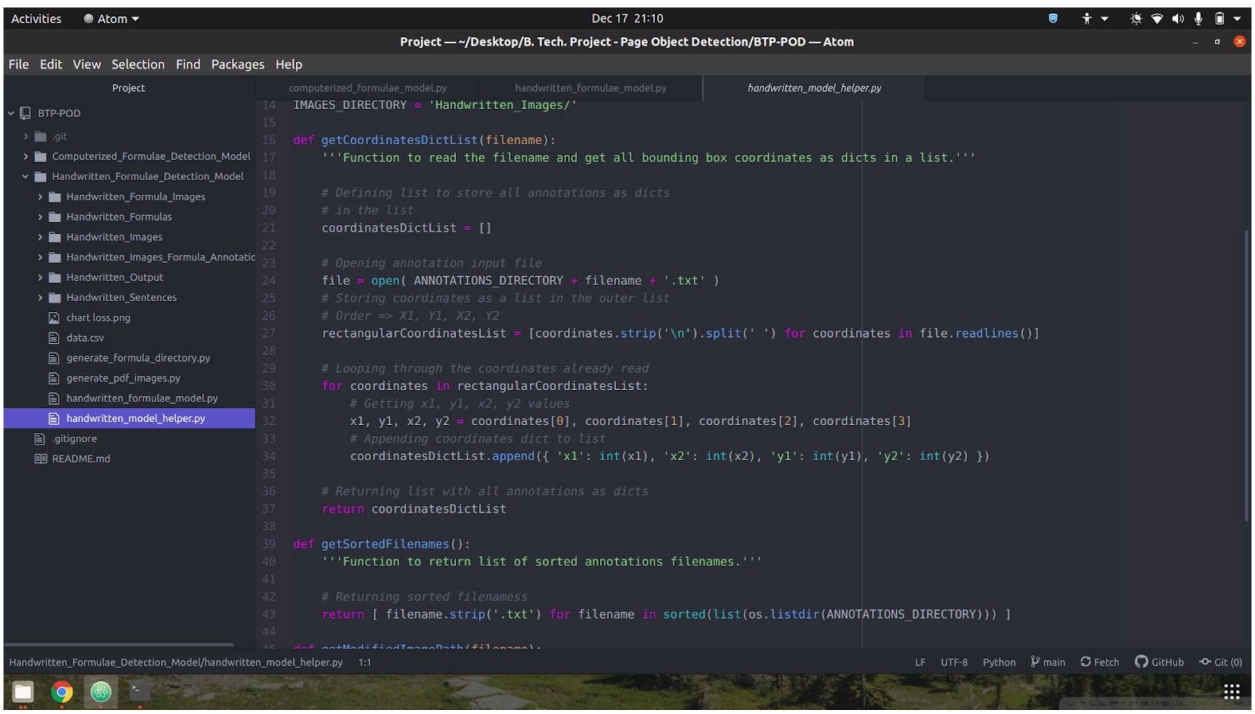
Here are some snippets of our code. The code is also available at https://drive.google.com/drive/folders/1melLTux5SQs13MkyC_tkehlHKo5U3947?usp=sharing



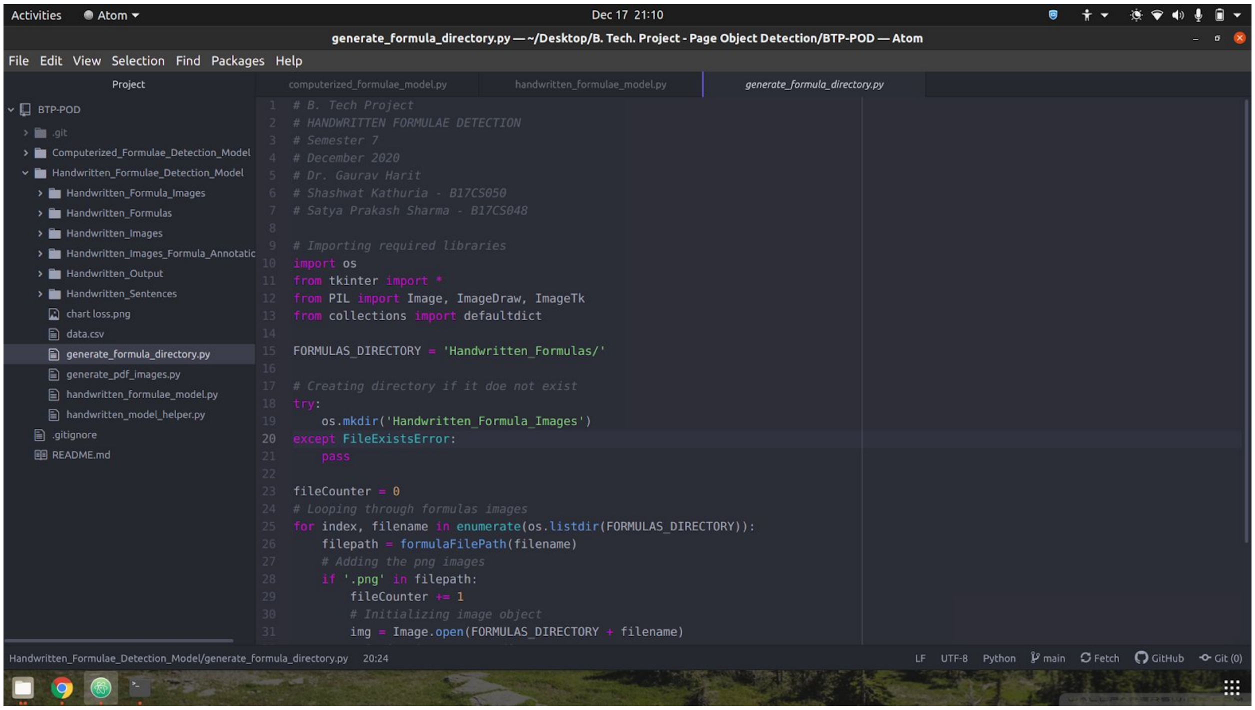


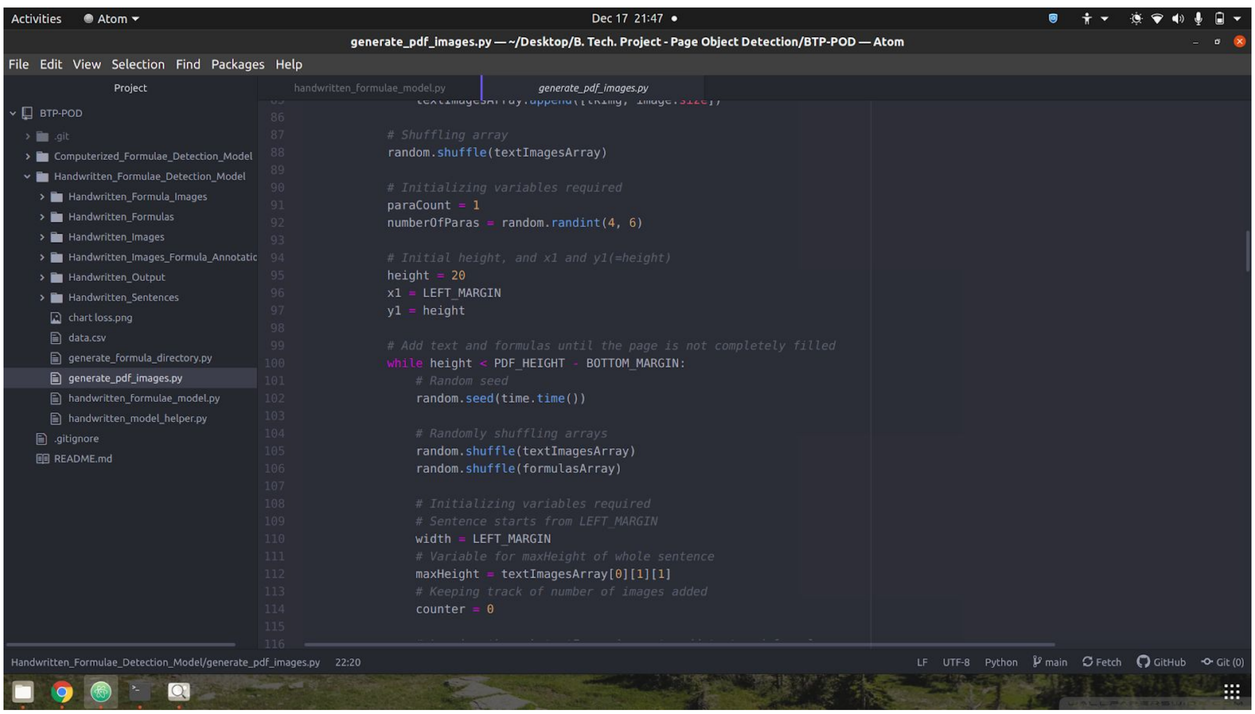


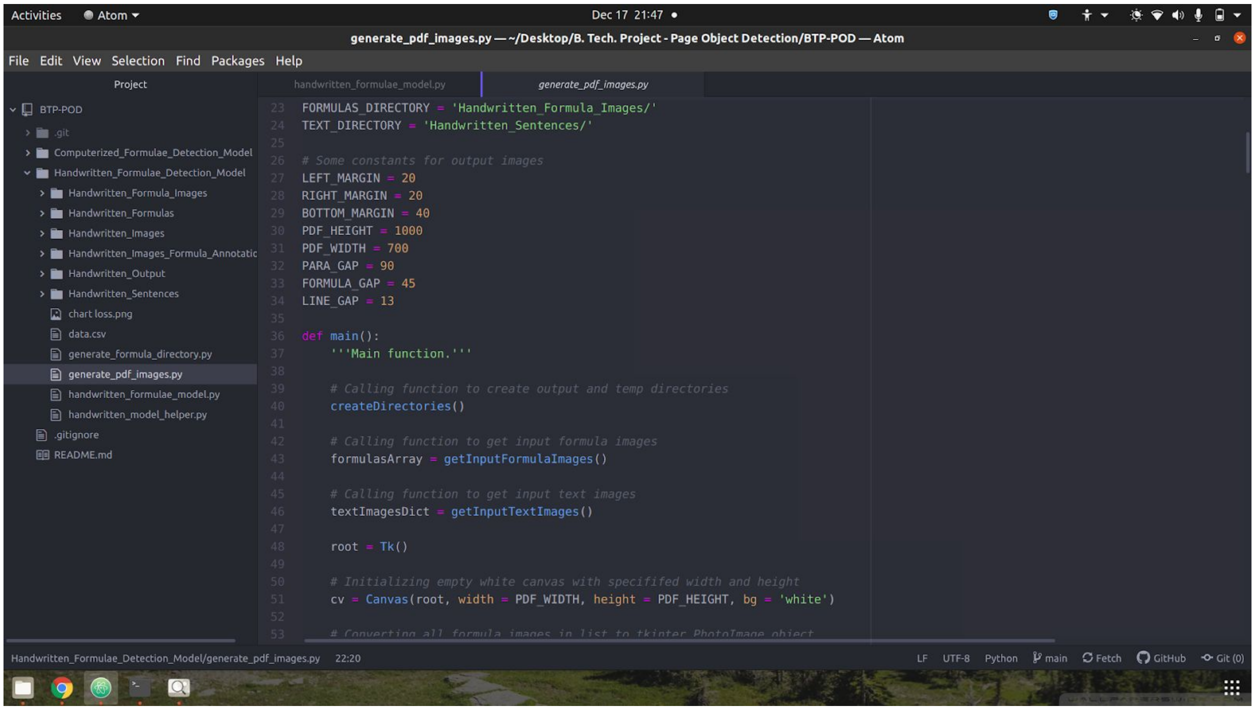
```python
import os, cv2, keras, random
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from handwritten_model_helper import getCoordinatesDictList, getSortedFilenames, getModifiedImagePath, getModifiedAnnotationsPath
from keras.layers import Dense
from keras import Model
from keras import optimizers
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.vgg16 import VGG16
from keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from keras.callbacks import ModelCheckpoint, EarlyStopping

def getIOU(boundingBox1, boundingBox2):
    '''Function to return calculated IOU value of the two bounding boxes given as input.'''

    # Asserting correct coordinates
    assert boundingBox1['x1'] < boundingBox1['x2']
    assert boundingBox1['y1'] < boundingBox1['y2']
    assert boundingBox2['x1'] < boundingBox2['x2']
    assert boundingBox2['y1'] < boundingBox2['y2']

    # Asserting correct coordinates
    xLeft = max(boundingBox1['x1'], boundingBox2['x1'])
    yTop = max(boundingBox1['y1'], boundingBox2['y1'])
    xRight = min(boundingBox1['x2'], boundingBox2['x2'])
    yBottom = min(boundingBox1['y2'], boundingBox2['y2'])
```

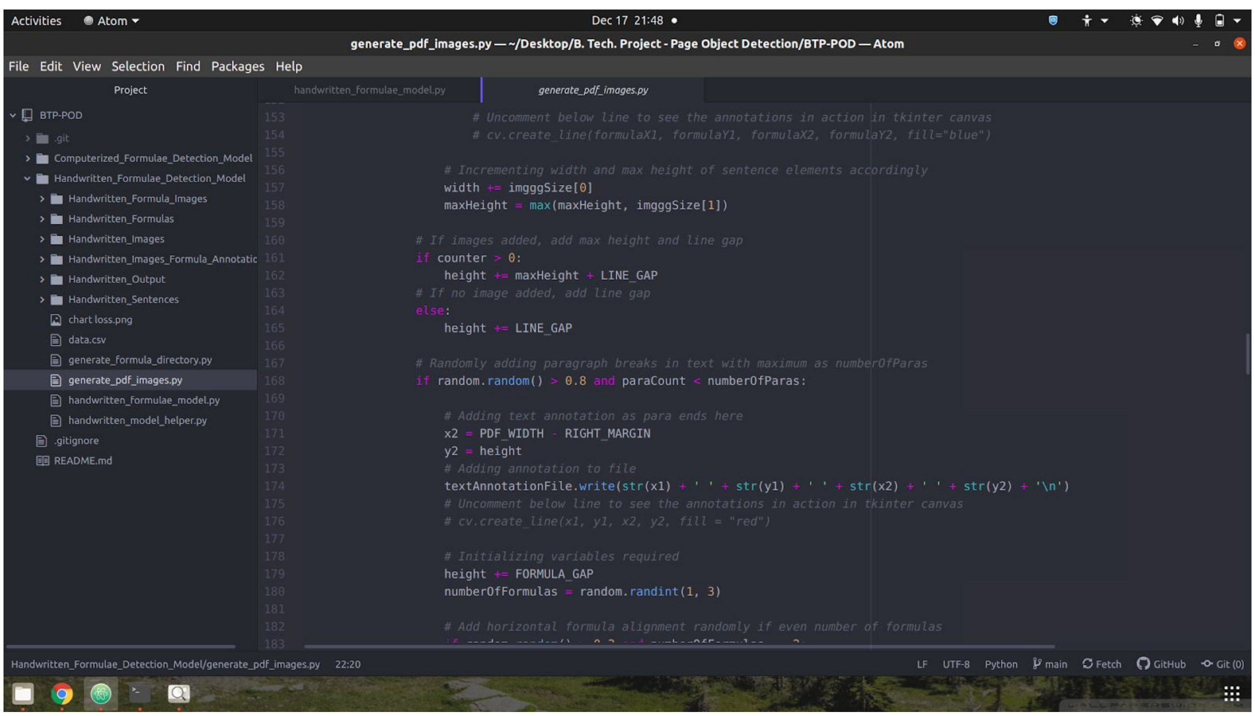


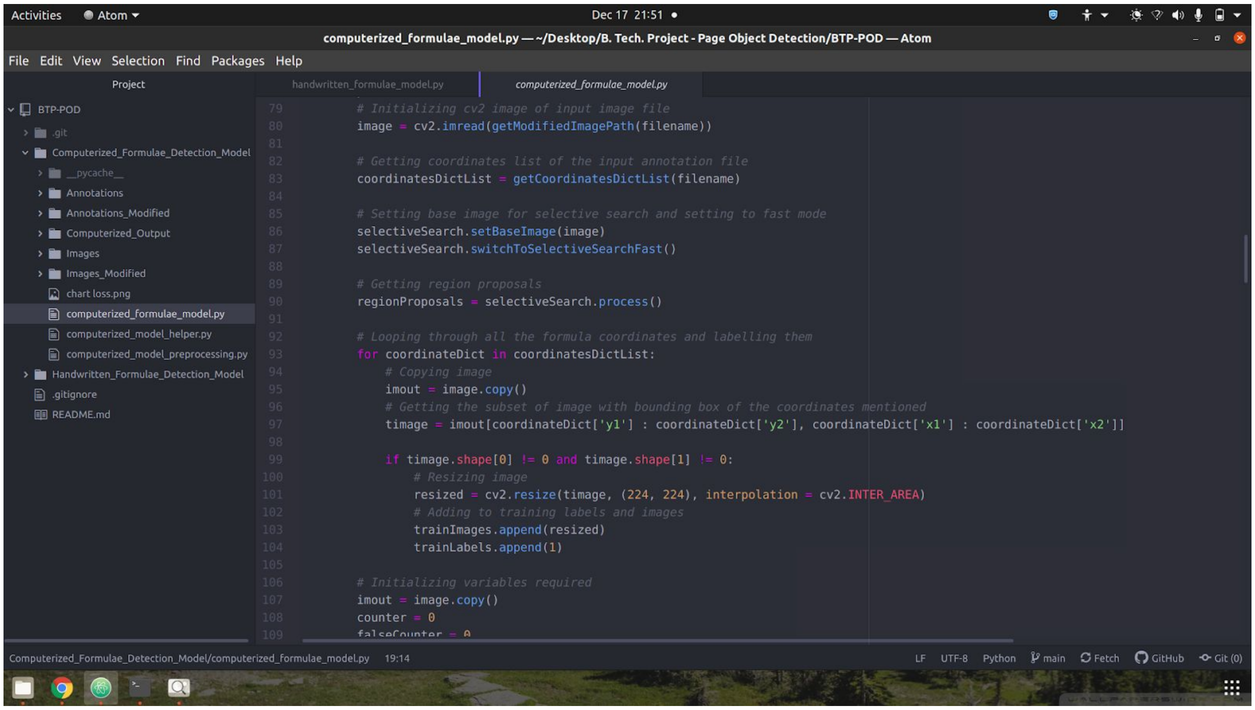
```python
IMAGES_DIRECTORY = 'Handwritten_Images/'

def getCoordinatesDictList(filename):
    '''Function to read the filename and get all bounding box coordinates as dicts in a list.'''

    # Defining list to store all annotations as dicts
    # in the list
    coordinatesDictList = []

    # Opening annotation input file
    file = open( ANNOTATIONS_DIRECTORY + filename + '.txt' )
    # Storing coordinates as a list in the outer list
    # Order => X1, Y1, X2, Y2
    rectangularCoordinatesList = [coordinates.strip('\n').split(' ') for coordinates in file.readlines()]

    # Looping through the coordinates already read
    for coordinates in rectangularCoordinatesList:
        # Getting x1, y1, x2, y2 values
        x1, y1, x2, y2 = coordinates[0], coordinates[1], coordinates[2], coordinates[3]
        # Appending coordinates dict to list
        coordinatesDictList.append({ 'x1': int(x1), 'x2': int(x2), 'y1': int(y1), 'y2': int(y2) })

    # Returning list with all annotations as dicts
    return coordinatesDictList

def getSortedFilenames():
    '''Function to return list of sorted annotations filenames.'''

    # Returning sorted filenames
    return [ filename.strip('.txt') for filename in sorted(list(os.listdir(ANNOTATIONS_DIRECTORY))) ]
```

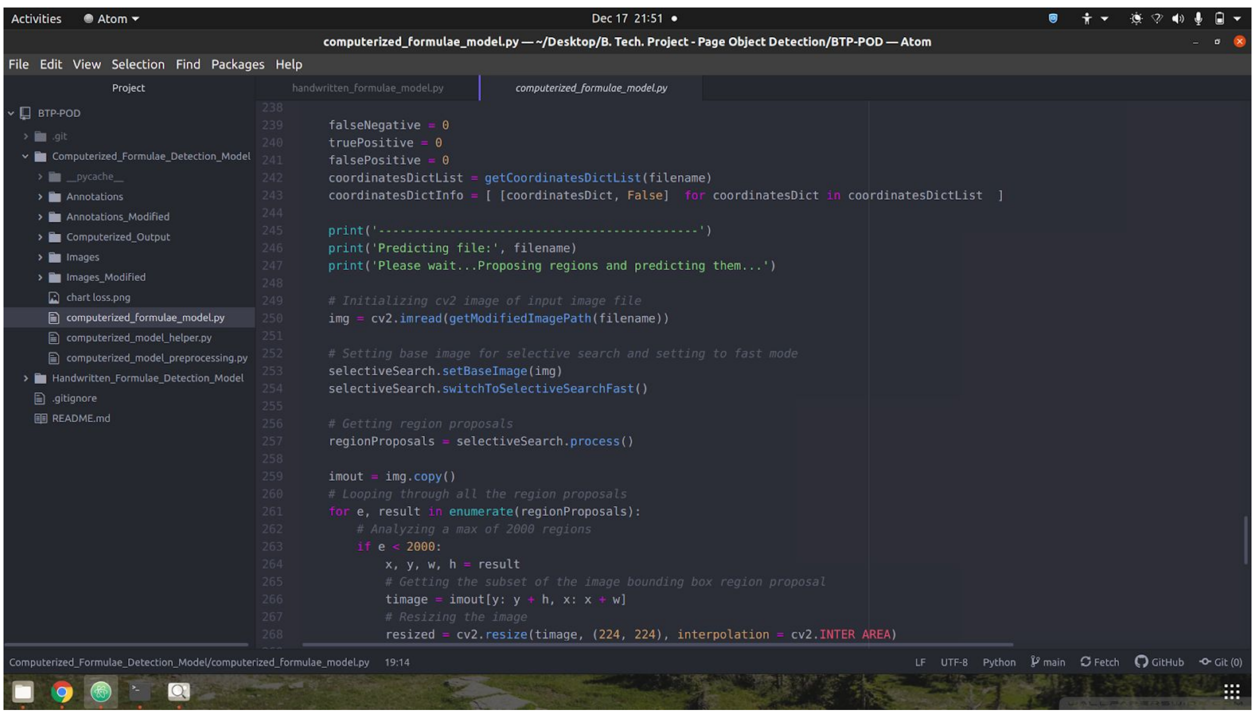

generate_formula_directory.py — ~/Desktop/B. Tech. Project - Page Object Detection/BTP-POD — Atom
# B. Tech Project
# HANDWRITTEN FORMULAE DETECTION
# Semester 7
# December 2020
# Dr. Gaurav Harit
# Shashwat Kathuria - B17CS050
# Satya Prakash Sharma - B17CS048
# Importing required libraries
import os
from tkinter import *
from PIL import Image, ImageDraw, ImageTk
from collections import defaultdict
FORMULAS_DIRECTORY = 'Handwritten_Formulas/'
# Creating directory if it doe not exist
try:
os.mkdir('Handwritten_Formula_Images')
except FileExistsError:
pass
fileCounter = 0
# Looping through formulas images
for index, filename in enumerate(os.listdir(FORMULAS_DIRECTORY)):
filepath = formulaFilePath(filename)
# Adding the png images
if '.png' in filepath:
fileCounter += 1
# Initializing image object
img = Image.open(FORMULAS_DIRECTORY + filename)

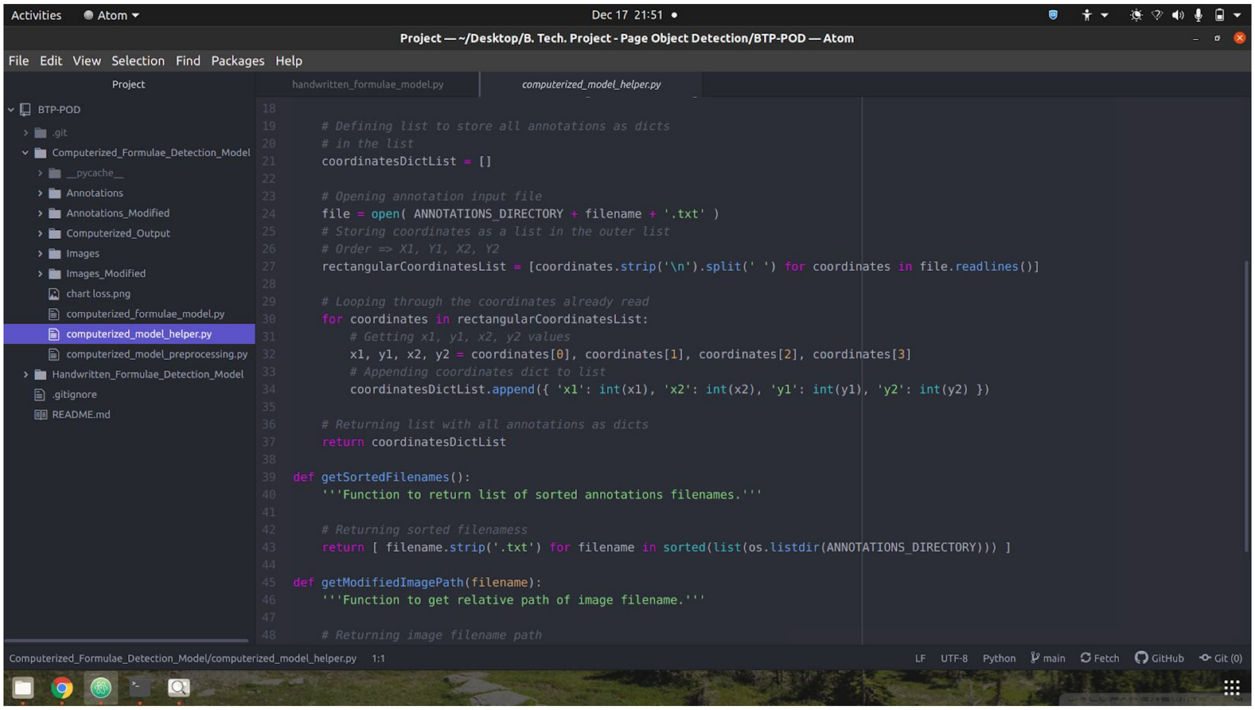

generate_pdf_images.py — ~/Desktop/B. Tech. Project - Page Object Detection/BTP-POD — Atom
# Shuffling array
random.shuffle(textImagesArray)
# Initializing variables required
paraCount = 1
numberOfParas = random.randint(4, 6)
# Initial height, and x1 and y1(=height)
height = 20
x1 = LEFT_MARGIN
y1 = height
# Add text and formulas until the page is not completely filled
while height < PDF_HEIGHT - BOTTOM_MARGIN:
# Random seed
random.seed(time.time())
# Randomly shuffling arrays
random.shuffle(textImagesArray)
random.shuffle(formulasArray)
# Initializing variables required
# Sentence starts from LEFT_MARGIN
width = LEFT_MARGIN
# Variable for maxHeight of whole sentence
maxHeight = textImagesArray[0][1][1]
# Keeping track of number of images added
counter = 0

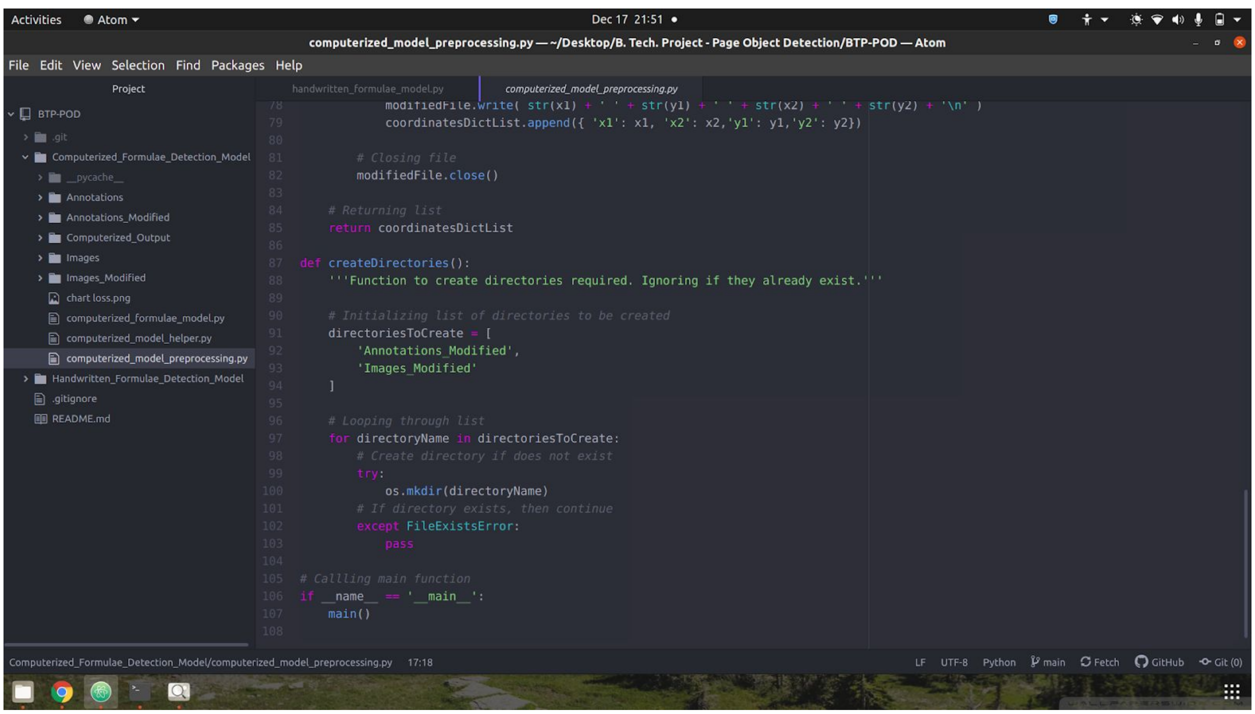

Activities
Atom
Dec 17 21:47
generate_pdf_images.py — ~/Desktop/B. Tech. Project - Page Object Detection/BTP-POD — Atom
File Edit View Selection Find Packages Help
Project
handwritten_formulae_model.py
generate_pdf_images.py
BTP-POD
.git
Computerized_Formulae_Detection_Model
Handwritten_Formulae_Detection_Model
Handwritten_Formula_Images
Handwritten_Formulas
Handwritten_Images
Handwritten_Images_Formula_Annotatic
Handwritten_Output
Handwritten_Sentences
chart loss.png
data.csv
generate_formula_directory.py
generate_pdf_images.py
handwritten_formulae_model.py
handwritten_model_helper.py
.gitignore
README.md
FORMULAS_DIRECTORY = 'Handwritten_Formula_Images/'
TEXT_DIRECTORY = 'Handwritten_Sentences/'
# Some constants for output images
LEFT_MARGIN = 20
RIGHT_MARGIN = 20
BOTTOM_MARGIN = 40
PDF_HEIGHT = 1000
PDF_WIDTH = 700
PARA_GAP = 90
FORMULA_GAP = 45
LINE_GAP = 13
def main():
'''Main function.'''
# Calling function to create output and temp directories
createDirectories()
# Calling function to get input formula images
formulasArray = getInputFormulaImages()
# Calling function to get input text images
textImagesDict = getInputTextImages()
root = Tk()
# Initializing empty white canvas with specifed width and height
cv = Canvas(root, width = PDF_WIDTH, height = PDF_HEIGHT, bg = 'white')
Handwritten_Formulae_Detection_Model/generate_pdf_images.py 22:20
LF UTF-8 Python main Fetch GitHub Git (0)


Activities
Atom
Dec 17 21:48
generate_pdf_images.py — ~/Desktop/B. Tech. Project - Page Object Detection/BTP-POD — Atom
File Edit View Selection Find Packages Help
Project
handwritten_formulae_model.py
generate_pdf_images.py
BTP-POD
.git
Computerized_Formulae_Detection_Model
Handwritten_Formulae_Detection_Model
Handwritten_Formula_Images
Handwritten_Formulas
Handwritten_Images
Handwritten_Images_Formula_Annotatic
Handwritten_Output
Handwritten_Sentences
chart loss.png
data.csv
generate_formula_directory.py
generate_pdf_images.py
handwritten_formulae_model.py
handwritten_model_helper.py
.gitignore
README.md
Uncomment below line to see the annotations in action in tkinter canvas
cv.create_line(formulaX1, formulaY1, formulaX2, formulaY2, fill="blue")
Incrementing width and max height of sentence elements accordingly
width += imgggSize[0]
maxHeight = max(maxHeight, imgggSize[1])
# If images added, add max height and line gap
if counter > 0:
height += maxHeight + LINE_GAP
# If no image added, add line gap
else:
height += LINE_GAP
# Randomly adding paragraph breaks in text with maximum as numberOfParas
if random.random() > 0.8 and paraCount < numberOfParas:
# Adding text annotation as para ends here
x2 = PDF_WIDTH - RIGHT_MARGIN
y2 = height
# Adding annotation to file
textAnnotationFile.write(str(x1) + ' ' + str(y1) + ' ' + str(x2) + ' ' + str(y2) + '\n')
# Uncomment below line to see the annotations in action in tkinter canvas
# cv.create_line(x1, y1, x2, y2, fill = "red")
# Initializing variables required
height += FORMULA_GAP
numberOfFormulas = random.randint(1, 3)
# Add horizontal formula alignment randomly if even number of formulas
Handwritten_Formulae_Detection_Model/generate_pdf_images.py 22:20
LF UTF-8 Python main Fetch GitHub Git (0)


computerized_formulae_model.py — ~/Desktop/B. Tech. Project - Page Object Detection/BTP-POD — Atom
File Edit View Selection Find Packages Help
handwritten_formulae_model.py
computerized_formulae_model.py
# Initializing cv2 image of input image file
image = cv2.imread(getModifiedImagePath(filename))
# Getting coordinates list of the input annotation file
coordinatesDictList = getCoordinatesDictList(filename)
# Setting base image for selective search and setting to fast mode
selectiveSearch.setBaseImage(image)
selectiveSearch.switchToSelectiveSearchFast()
# Getting region proposals
regionProposals = selectiveSearch.process()
# Looping through all the formula coordinates and labelling them
for coordinateDict in coordinatesDictList:
# Copying image
imout = image.copy()
# Getting the subset of image with bounding box of the coordinates mentioned
timage = imout[coordinateDict['y1'] : coordinateDict['y2'], coordinateDict['x1'] : coordinateDict['x2']]
if timage.shape[0] != 0 and timage.shape[1] != 0:
# Resizing image
resized = cv2.resize(timage, (224, 224), interpolation = cv2.INTER_AREA)
# Adding to training labels and images
trainImages.append(resized)
trainLabels.append(1)
# Initializing variables required
imout = image.copy()
counter = 0
falseCounter = 0
Computerized_Formulae_Detection_Model/computerized_formulae_model.py 19:14


computerized_formulae_model.py — ~/Desktop/B. Tech. Project - Page Object Detection/BTP-POD — Atom
File Edit View Selection Find Packages Help
handwritten_formulae_model.py
computerized_formulae_model.py
falseNegative = 0
truePositive = 0
falsePositive = 0
coordinatesDictList = getCoordinatesDictList(filename)
coordinatesDictInfo = [[coordinatesDict, False] for coordinatesDict in coordinatesDictList]
print('---')
print('Predicting file:', filename)
print('Please wait...Proposing regions and predicting them...')
# Initializing cv2 image of input image file
img = cv2.imread(getModifiedImagePath(filename))
# Setting base image for selective search and setting to fast mode
selectiveSearch.setBaseImage(img)
selectiveSearch.switchToSelectiveSearchFast()
# Getting region proposals
regionProposals = selectiveSearch.process()
imout = img.copy()
# Looping through all the region proposals
for e, result in enumerate(regionProposals):
# Analyzing a max of 2000 regions
if e < 2000:
x, y, w, h = result
# Getting the subset of the image bounding box region proposal
timage = imout[y: y + h, x: x + w]
# Resizing the image
resized = cv2.resize(timage, (224, 224), interpolation = cv2.INTER_AREA)
Computerized_Formulae_Detection_Model/computerized_formulae_model.py 19:14


Activities
Atom
Dec 17 21:51
Project — ~/Desktop/B. Tech. Project - Page Object Detection/BTP-POD — Atom
File Edit View Selection Find Packages Help
Project
handwritten_formulae_model.py
computerized_model_helper.py
BTP-POD
.git
Computerized_Formulae_Detection_Model
__pycache__
Annotations
Annotations_Modified
Computerized_Output
Images
Images_Modified
chart loss.png
computerized_formulae_model.py
computerized_model_helper.py
computerized_model_preprocessing.py
Handwritten_Formulae_Detection_Model
.gitignore
README.md
    # Defining list to store all annotations as dicts
    # in the list
    coordinatesDictList = []
    # Opening annotation input file
    file = open(ANNOTATIONS_DIRECTORY + filename + '.txt')
    # Storing coordinates as a list in the outer list
    # Order => X1, Y1, X2, Y2
    rectangularCoordinatesList = [coordinates.strip('\n').split(' ') for coordinates in file.readlines()]
    # Looping through the coordinates already read
    for coordinates in rectangularCoordinatesList:
        # Getting x1, y1, x2, y2 values
        x1, y1, x2, y2 = coordinates[0], coordinates[1], coordinates[2], coordinates[3]
        # Appending coordinates dict to list
        coordinatesDictList.append({ 'x1': int(x1), 'x2': int(x2), 'y1': int(y1), 'y2': int(y2) })
    # Returning list with all annotations as dicts
    return coordinatesDictList
def getSortedFilenames():
    '''Function to return list of sorted annotations filenames.'''
    # Returning sorted filenamess
    return [filename.strip('.txt') for filename in sorted(list(os.listdir(ANNOTATIONS_DIRECTORY)))]
def getModifiedImagePath(filename):
    '''Function to get relative path of image filename.'''
    # Returning image filename path
Computerized_Formulae_Detection_Model/computerized_model_helper.py 1:1
LF UTF-8 Python main Fetch GitHub Git (0)


Activities
Atom
Dec 17 21:51
computerized_model_preprocessing.py — ~/Desktop/B. Tech. Project - Page Object Detection/BTP-POD — Atom
File Edit View Selection Find Packages Help
Project
handwritten_formulae_model.py
computerized_model_preprocessing.py
BTP-POD
.git
Computerized_Formulae_Detection_Model
__pycache__
Annotations
Annotations_Modified
Computerized_Output
Images
Images_Modified
chart loss.png
computerized_formulae_model.py
computerized_model_helper.py
computerized_model_preprocessing.py
Handwritten_Formulae_Detection_Model
.gitignore
README.md
            modifiedFile.write(str(x1) + ' ' + str(y1) + ' ' + str(x2) + ' ' + str(y2) + '\n')
            coordinatesDictList.append({ 'x1': x1, 'x2': x2,'y1': y1,'y2': y2})
        # Closing file
        modifiedFile.close()
    # Returning list
    return coordinatesDictList
def createDirectories():
    '''Function to create directories required. Ignoring if they already exist.'''
    # Initializing list of directories to be created
    directoriesToCreate = [
        'Annotations_Modified',
        'Images_Modified'
    ]
    # Looping through list
    for directoryName in directoriesToCreate:
        # Create directory if does not exist
        try:
            os.mkdir(directoryName)
        # If directory exists, then continue
        except FileExistsError:
            pass
# Callling main function
if __name__ == '__main__':
    main()
Computerized_Formulae_Detection_Model/computerized_model_preprocessing.py 17:18
LF UTF-8 Python main Fetch GitHub Git (0)

References

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks: Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun: https://arxiv.org/abs/1506.01497

ICDAR 2017 Competition on Page Object Detection: Liangcai Gao, Xiaohan Yi, Zhuoren Jiang, Leipeng Hao, Zhi Tang: https://ieeexplore.ieee.org/document/8270162

ICDAR 2017 Dataset:
https://mega.nz/file/6QlwGaAb#BKf962iBlfeL7oEqaVnDC4K3F47zrqtaU12OCJlcbTw

IAM Dataset: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database

CROHME Dataset: https://www.isical.ac.in/~crohme/CROHME_data.html

Tensorflow: https://www.tensorflow.org/guide
OpenCV: https://opencv.org/
Keras: https://keras.io/

THANK YOU