B. Tech. Project

Handwritten Formulae Detection

Mentor - Prof. Dr. Gaurav Harit

Shashwat Kathuria - B17CS050

Satya Prakash Sharma - B17CS048

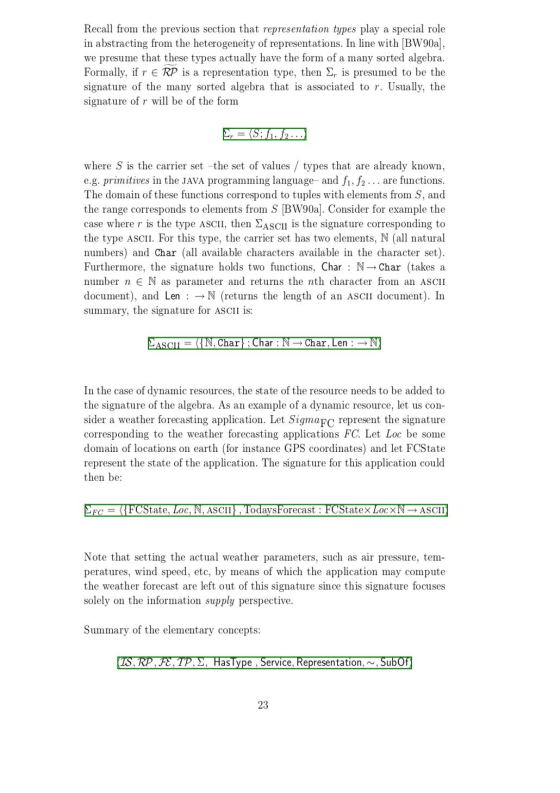
19th December 2020

Outline

I. Aim of Project
II. Implementation Steps
III. Explanation of Model Implemented
    A. Dataset (Normal & Synthetic) & Preprocessing
    B. Training & Testing
    C. Model
    D. Training Strategy & Loss Function
    E. Results
IV. Conclusion
V. Sample Outputs
VI. References

Aim of Project

- **Detect various types of formulas** which can be present in images taken from PDF documents, like research papers.
- Various challenges:
  - **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.



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

$$\boxed{\Sigma_r = \langle S; f_1, f_2 \dots \rangle}$$

where S is the carrier set –the set of values / types that are already known, e.g. *primitives* in the JAVA programming language– and $f_1, f_2 \dots$ 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 $\mathtt{Char}$ (all available characters available in the character set). Furthermore, the signature holds two functions, $\mathsf{Char} : \mathbb{N} \rightarrow \mathtt{Char}$ (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:

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

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:

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

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:

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

23

Implementation Steps

Study about RCNN Model and Page Object Detection

Study about RCNN Model and ICDAR 2017 Page Object Detection Competition.

Computerized Formulae Detection Model

Implement an RCNN model to detect computerized formulae in the ICDAR 2017 dataset.

Study about Generation of Synthetic Datasets

Research about GANs and other possible techniques to generate a handwritten dataset with annotations.

Generate Handwritten Formulae Dataset

Generate dataset with different types of handwritings and different types of alignments of formulae.

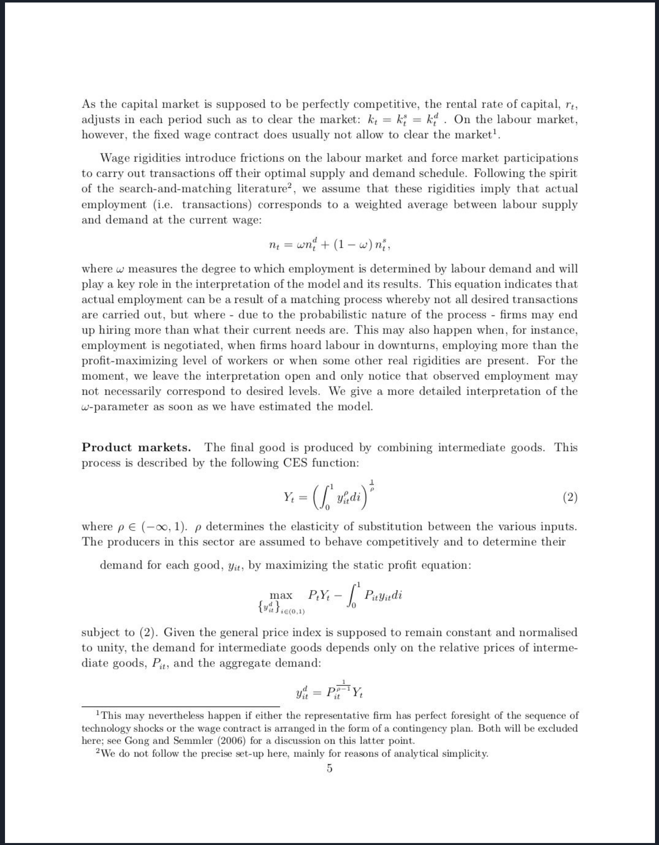
Handwritten Formulae Detection Model

Fine tune existing RCNN model to detect handwritten formulae using the generated dataset.

Dataset & Preprocessing

I. Computerized Text Formulae

- The ICDAR 2017 dataset contains of **1600 images** having **tables, formulas** and **figures annotations.**
- Out of 1600 images, about **910 images contain formulas**
- Parse the coordinates of the formula regions by parsing the ground truth xml file.



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.

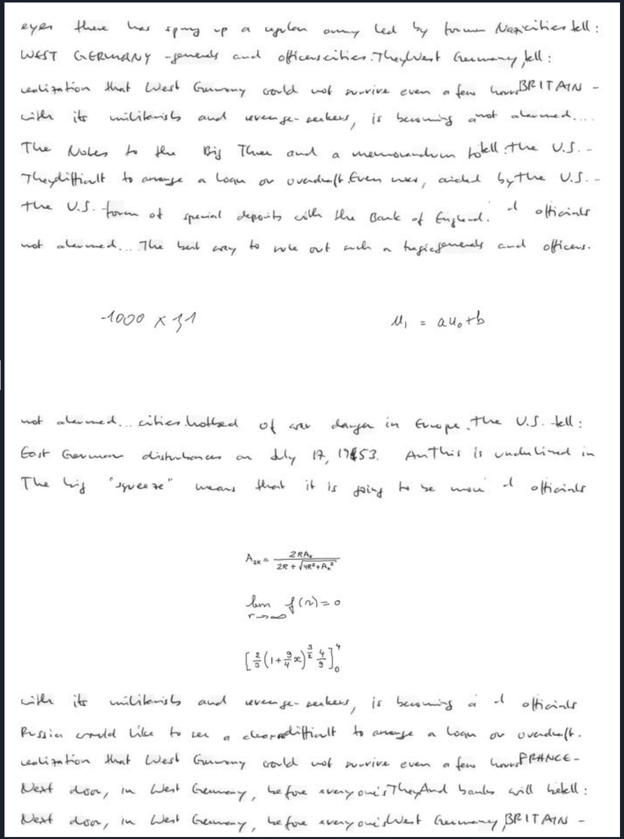
[2]We do not follow the precise set-up here, mainly for reasons of analytical simplicity.

5

Synthetic Dataset & Preprocessing

II. Handwritten Text Formulae

- ➔ Used IAM dataset having **handwriting of various common types** and CROHME dataset having **mathematical handwritten formulae**.
- ➔ 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
- ➔ Parse the coordinates of the formula regions by parsing the ground truth of the generated images.



Training & Testing

- ➔ Selective search to obtain **region proposals of the images.**
- ➔ **Intersection over Union (IoU)** of the **proposed region with the annotated formula region used to add images to the training set as labelled or not.**
- ➔ **Limit the number of positive and negative samples** entering the model, to an equal of 30 per page.
- ➔ Because there are **many white spaces inside some formulas**, the **IoU is set to >= 0.6 for formulae region**, and <= 0.05 for non formulae region.
- ➔ Splitting Ratio: **80 : 20**

Model

- ➔ The model implemented is **Regions with Convolutional Neural Networks (RCNN)**.

- ➔ We use **VGG 16 Model** because it performs well with specific tasks, along with **two unit softmax layer**, 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.

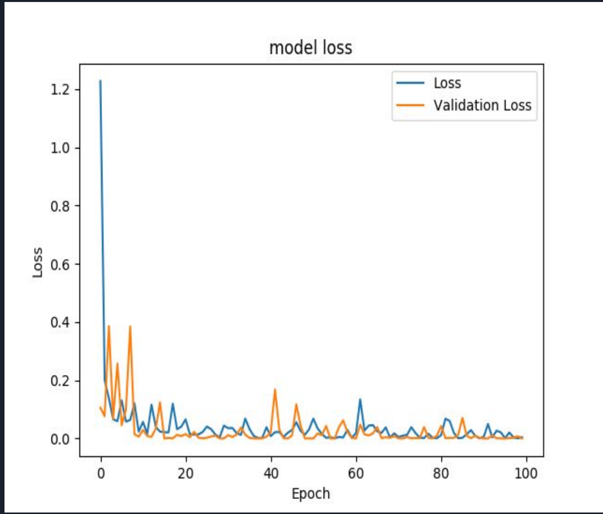
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 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

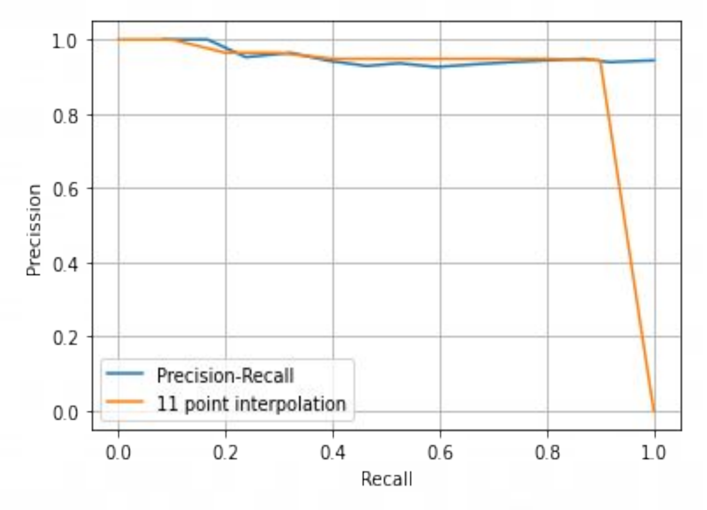
Total params: 134,268,738
Trainable params: 126,633,474
Non-trainable params: 7,635,264

Training Strategy & Loss Function

- **Adam for optimizer** because it is efficient in use and as it does stochastic optimization. The learning rate is set to 0.001 .
- Loss and accuracy are kept track of **in each epoch**, for 100 epochs with 10 steps per epoch and **update the model parameters only if the loss value decreases.**
- C**ategorical cross entropy is used as loss** because our prediction is categorical and binary, whether or not there is a formula (one hot encoding).
- For **prediction**, we **get proposed regions using selective search** and then obtain the **model prediction**, which if it is >= 0.75, we consider the proposed region as having formula, otherwise not.

Results





- Loss and accuracy are kept track of **in each epoch**, for 100 epochs with 10 steps per epoch and **update the model parameters only if the loss value decreases.**

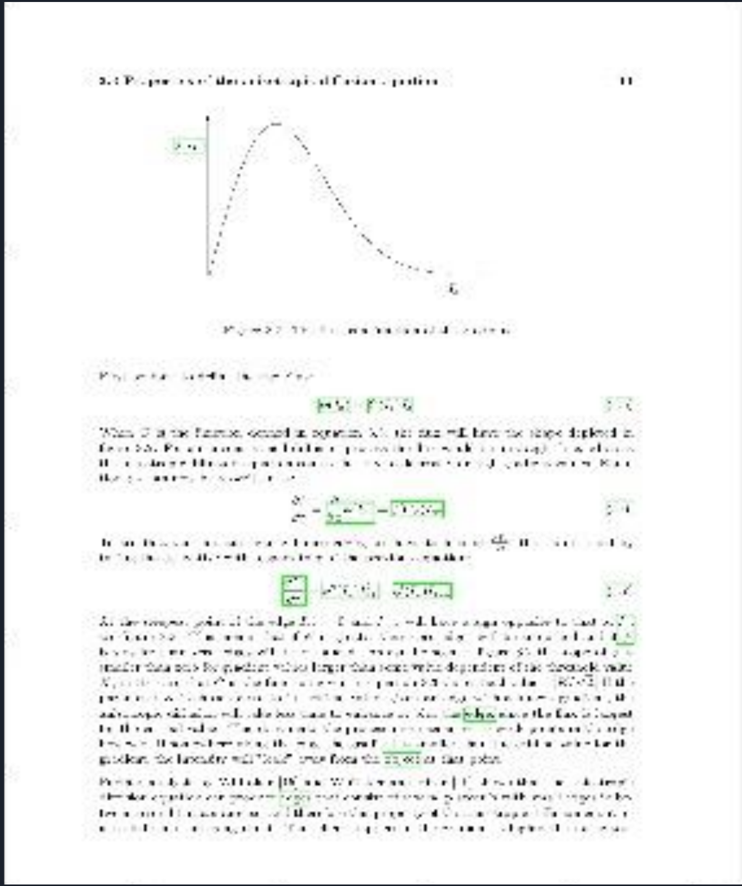
- Average Precision: **87.38%** , calculated using 11 Point Interpolation Technique

Conclusion

- ➔ The model we have implemented is able to identify formulae in computerized as well as handwritten forms in the images.
- ➔ Some regions were formulas get selected inside text are when there are **equations/formulas inside the text itself, or inside tables or figures**.
- ➔ Average Precision of our Handwritten Formulae Detection is: **87.38%**
- ➔ Code is available at:

  https://drive.google.com/drive/folders/1melLTux5SQs13MkyC_tkehlHKo5U3947?usp=sharing

Computerized Formulae Detection Sample Results



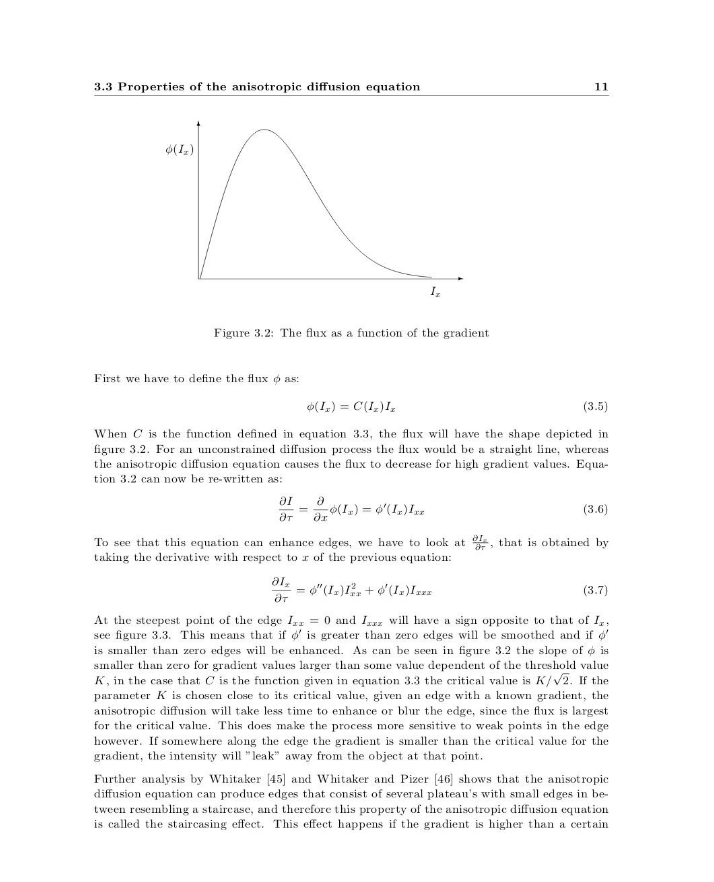


Figure 3.2: The flux as a function of the gradient

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:

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

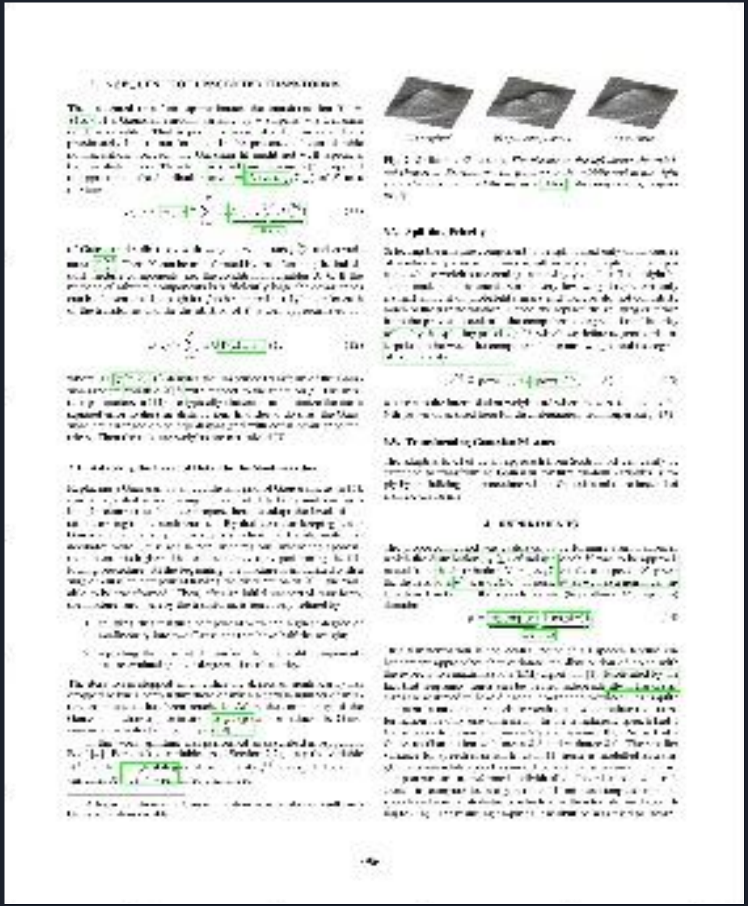
To see that this equation can enhance edges, we have to look at $\frac{\partial I_x}{\partial \tau}$, that is obtained by taking the derivative with respect to x of the previous equation:

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

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

Computerized Formulae Detection Sample Results



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.

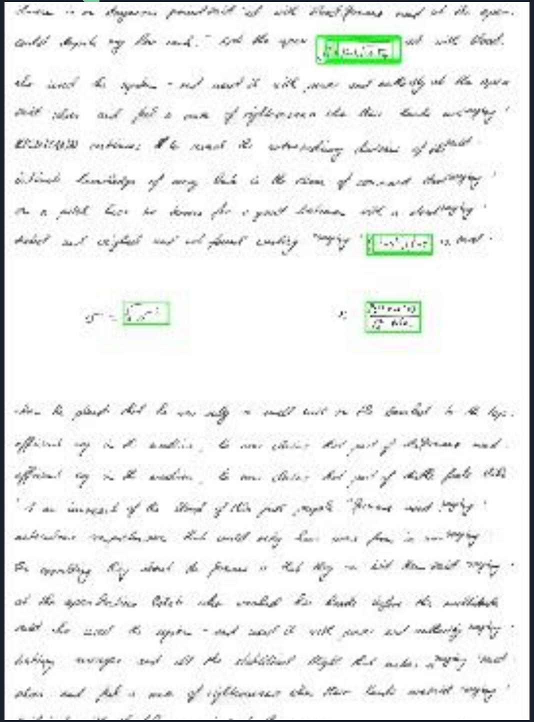
4. EXPERIMENTS

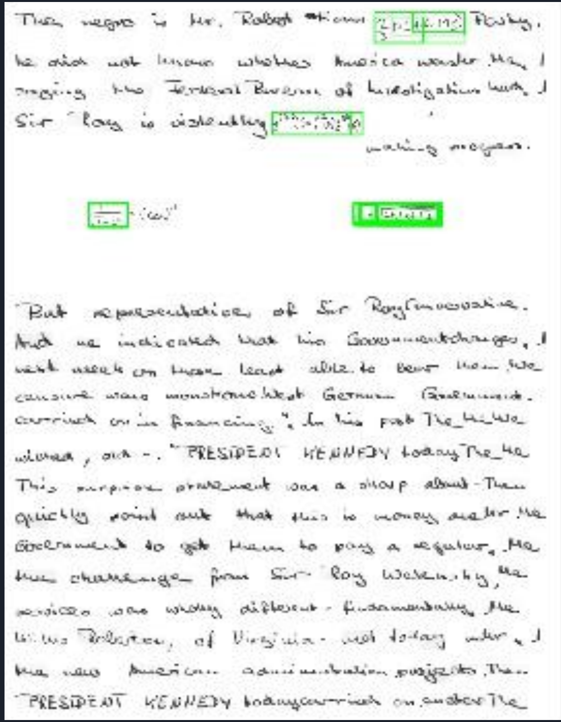
The proposed method was evaluated by performing a simulation, in which the distribution $p_Y(y)$ of noisy speech Y was to be approximated from the distribution $\mathcal{N}(x; \mu_X, \Sigma_X)$ of clean speech X, given the distribution $\mathcal{N}(n; \mu_N, \Sigma_N)$ of noise N as well as a nonlinear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

Handwritten Formulae Detection Sample Results





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
- Datasets
  - 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

Thank You