

Technische Hochschule Deggendorf
Faculty of Angewandte Informatik
The University of South Bohemia in České Budějovice
Faculty of Science

Degree Master Künstliche Intelligenz und Data Science

**VERBESSERUNG DER VALIDIERUNG VON
TABLENET: EVALUIERUNG DER GENAUIGKEIT DER
BESTEHENDEN ARCHITEKTUR MIT NEUARTIGEN
TRAININGSDATEN**

**ENHANCING TABLENET'S VALIDATION:
EVALUATING THE ACCURACY OF THE EXISTING
ARCHITECTURE WITH NOVEL TRAINING DATA**

Master's thesis to obtain the academic degree:

Master of Science (M.Sc.)

at the Technical University of Deggendorf
and the University of South Bohemia

Presented by:
Krishna Sai Sahukara
Matriculation Number:
807635

Supervisor:
Prof. Dr. Andreas Fischer

Co-Supervisor:
Prof. Dr. Andreas Berl

On: 10. August 2023

Erklärung

Name des Studierenden: Krishna Sai Sahukara

Name des Betreuenden: Prof. Dr. Andreas Fischer

Thema der Abschlussarbeit:

Verbesserung der Validierung von TableNet: Evaluierung der Genauigkeit der bestehenden Architektur mit neuartigen Trainingsdaten

.....
.....

1. Ich erkläre hiermit, dass ich die Abschlussarbeit gemäß § 35 Abs. 7 RaPO (Rahmenprüfungsordnung für die Fachhochschulen in Bayern, BayRS 2210-4-1-4-1-WFK) selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Deggendorf, ...

Datum

.....

Unterschrift des Studierenden

2. Ich bin damit einverstanden, dass die von mir angefertigte Abschlussarbeit über die Bibliothek der Hochschule einer breiteren Öffentlichkeit zugänglich gemacht wird:

- ☐ Nein
☐ Ja, nach Abschluss des Prüfungsverfahrens
☐ Ja, nach Ablauf einer Sperrfrist von ...Jahren.

Deggendorf, ...

Datum

.....

Unterschrift des Studierenden

Bei Einverständnis des Verfassenden vom Betreuenden auszufüllen:

Eine Aufnahme eines Exemplars der Abschlussarbeit in den Bestand der Bibliothek und die Ausleihe des Exemplars wird:

- ☐ Befürwortet
☐ Nicht befürwortet

Deggendorf, ...

Datum

.....

Unterschrift des Betreuenden

Annotation

This thesis introduces a innovative approach to automatically create annotated datasets in which different table layouts are systematically generated along with corresponding ground truth coordinates to enhance table detection in images. This thesis aims to implement the existing deep learning architecture (TableNet), renowned for its table detection capabilities, and assess its performance on the established marmot dataset and our newly generated dataset through Bitwise XOR and Intersection over Union (IOU) metrics. This novel dataset facilitates enhanced evaluation of current state-of-the-art architectures and empowers the development of more effective models through comprehensive training. This evaluation seeks to assess the effectiveness of the proposed dataset generation method in improving the accuracy of table detection using TableNet.

I declare that I am the author of this qualification thesis and that in writing it I have used the sources and literature displayed in the list of used sources only.

Deggendorf, ..10.08.2023
Datum

S. Krishna Sai
Signature of student

Abstract

In the modern age, smartphones and scanners capture and share images of crucial documents, often containing tables with vital data. However, manually extracting data from these tables proves to be a cumbersome and inefficient process. To address this challenge, this thesis introduces an innovative approach that focuses on the realm of data augmentation to enhance table detection in document images. This approach focuses on creating a novel training data set in which different table layouts are systematically generated along with corresponding ground truth coordinates. Leveraging this augmented dataset paves the way for more robust and accurate model development. This methodology not only expedites the process but also reduces the overall effort required for dataset annotation. In conjunction with the data augmentation strategy, our framework harnesses the capabilities of an existing end-to-end Deep learning architecture that recognizes tabular sections in document images and performs end-to-end table structure identification, significantly simplifying and accelerating the entire process, as it efficiently handles the interdependent tasks of table detection and structure identification. This novel dataset allows for easy evaluation of existing state-of-the-art architectures and enables the development of better-performing models through training on this extensive dataset. The fusion of data augmentation techniques and deep learning paradigms paves the way for transformative advancements in this field. As the demand for extracting accurate information from document images continues to grow, and our methodology has the potential to revolutionize and improve data analysis and document processing.

Contents

Abstract	vii
1 Introduction	1
1.1 Motivation	1
1.2 Outline	2
2 Related Work	3
3 Background Knowledge	5
3.1 Convolutional Neural Networks	5
3.2 Object Detection	6
3.3 VGG-19	7
3.4 Semantic Segmentation	9
3.5 Encoder-Decoder	10
3.6 TableNet	12
4 Methodology	15
4.1 Dataset	15
4.2 Novel Training Data	18
4.3 Generating Co-ordinates of the Table	20
5 Implementation	23
5.1 Architecture	23
5.2 Tables_Generation	24
5.3 Data_Preprocessing For MARMOT Dataset	25
5.4 Generating_Ground_Truth_Coordinates	27
5.5 TableNet	28
5.6 Predict	30
5.7 Noise_Removal	31
5.8 Evaluation_Pipeline	31
5.8.1 Performance_Metric_IOU	31
5.8.2 Bitwise_XOR	32
6 Evaluation	35
6.1 Intersection Over Union (IOU)	35
6.2 Bitwise XOR	36
6.3 Results	37

7 Conclusion and Future Works	51
--------------------------------------	-----------

1 Introduction

Smartphones and scanners are widely utilized in the current era to take photos of important documents and transmit them on a frequent basis. An increasing number of people are exchanging their documents as smartphone images. Many of these documents have one or more tables with a lot of information in them. Extracting this information from the image is a task of the utmost importance because these tables frequently contain highly vital information [1]. The necessity to extract the information held in unstructured document images, such as insurance claim forms, financial invoices, and retail receipts is growing increasingly urgent due to the widespread usage of mobile phones and scanners to photograph and upload documents. In the contemporary era, information extraction from these tables is done manually, which takes a lot of time and effort and could be more efficient. Using two different models, previous approaches have attempted to handle the challenges of table detection and structure recognition independently.

1.1 Motivation

In recent times, the application of Deep Learning models has achieved notable success in various computer vision tasks. Nonetheless, the effectiveness of these models heavily relies on the availability of extensive and carefully annotated datasets for training. Regrettably, the domain of table detection has encountered a significant obstacle due to the scarcity of manually annotated datasets, hindering the advancement of superior model performance.

To address this critical challenge, this study endeavors to propose an innovative solution by creating a new training dataset tailored specifically for table detection. The primary objective is to facilitate the evaluation of existing state-of-the-art architectures associated with table detection and empower the development of highly proficient models through training on this extensive and meticulously annotated data.

The traditional manual annotation approach is laborious and time-consuming, requiring substantial efforts from experts. Our novel approach offers a more efficient alternative, alleviating the burden of manual annotation by automating the process of generating annotated data. This not only expedites the training process but also ensures a diverse and comprehensive dataset, leading to a more robust and accurate model.

Table detection includes locating the picture pixel coordinates of the tabular sub-image, whereas tabular structure identification entails identifying the individual rows and columns of the identified Table. Our time and effort will be much reduced if we have a deep-learning

model to complete this task. Therefore, having a system that is capable of recognizing tabular sections in a document image, using only the image as input, which also has end-to-end functionality, detect the table structure (columns), and then automatically extract the textual content from the tabular region will be of significant assistance. This will simplify the task and speed it up significantly. It not only locates the Table in an image but also creates the Table's structure by segmenting the Table's columns. In order to read the contents of the Table after determining its structure from the image, one can utilize the Pytesseract OCR library.

Furthermore, to enhance the performance of table detection models, this thesis explores an existing innovative end-to-end deep learning model that leverages the intrinsic relationship between two crucial objectives: table detection and table structure identification. The interconnected nature of these tasks enables us to treat them as linked sub-problems, effectively solved using a single neural network. Specifically, the model automatically segments the table and column masks when tables are present in the input image, streamlining the process of extracting table contents using Optical Character Recognition (OCR) [2].

By harnessing the natural dependency between these tasks, our proposed approach not only enhances accuracy but also demonstrates significant efficiency gains. The resulting model becomes relatively lightweight and less computationally intensive, making it an appealing option for real-world applications where computational resources may be limited.

In summary, this thesis aims to revolutionize table detection by introducing a novel process for generating training data and capitalizing on the interdependence of table detection and structure identification through an innovative end-to-end deep learning model. The combined approach has the potential to elevate the performance of table detection models, paving the way for various applications, including document processing, data extraction, and information retrieval.

1.2 Outline

This section outlines the organization of the chapters in this thesis. Chapter 2 discusses previous research that relates to our work in this thesis. Chapter 3 covers the background information necessary for understanding the contents of this thesis, including Convolutional Networks [3], VGG19 [4], Object Detection [5, 6], Semantic Segmentation [7], Encoder and Decoder Architecture [8], and TableNet Architecture [9]. Chapter 4 provides details on the dataset used in this thesis and the methodology for generating the novel training data. Chapter 5 describes the implementation of generating annotated training data, the TableNet algorithm, and the evaluation pipeline in this thesis. Chapter 6 presents a brief description of the evaluation Pipeline, i.e., Intersection Over Union (IOU) and Bitwise XOR, and compares the results of our experiments on this benchmark. Chapter 7 concludes the work in this thesis and suggests potential directions for future research.

2 Related Work

Numerous publications have been dedicated to studying table comprehension, accompanied by comprehensive surveys that provide descriptions and summaries of the current state-of-the-art in this domain [10, 11, 12, 13]. Most of them reported table identification and data extraction results separately [14].

Before the era of deep learning, table identification was primarily based on heuristics or metadata. TINTIN[15] used structural information to identify tables and their component fields. Cesarini et al.[16], 2002, was among the pioneers in applying machine learning techniques to the table identification task. They presented hierarchical representations based on the MXY tree, which was the first attempt to use machine learning techniques to detect tables. Their method, known as Tabfinder, involved converting the document into an MXY tree view and identifying blocks enclosed by horizontal or vertical lines. A deep search on these nodes revealed potential table candidates.

T. Kasar et al.[17] focused on detecting intersecting horizontal and vertical lines as well as low-level features using an SVM classifier to determine whether an image region belongs to a table region. They developed a set of hand-crafted features that they used to train an SVM-based classifier. Although this method eliminated the need for heuristics or user-defined parameters, its application was limited because it was highly dependent on visible guidelines. Using unsupervised learning of weak labels for each document line and linguistic information extracted from the region.

Probabilistic graphical models were also used to examine the tables; Silva et al.[18] developed a joint probability distribution based on sequential observations of page views and hidden row states (HMMs) that allowed possible table rows to be combined into full tables. Silva went on to emphasize the importance of integrating probabilistic models and multiple approaches, stressing their superiority over strict heuristics in his Ph.D. thesis [19]. Jing Fang et al.[20] began the discovery process by using the table header and analyzing its elements. Raskovich et al.[21] attempted to detect borderless tables using whitespaces as a heuristic instead of content. Fan and Kim[22] successfully trained a set of generative and discriminative classifiers to detect tables.

Compared to table detection, the field of table structure identification has received relatively less attention in research. Among the early successful systems, Kieninger and Dengel[23] introduced the T-RECS approach. This method is one of the first works which uses clustering of some word segments and text overlap analysis within tables to extract table data. This method initially groups words into columns by evaluating their horizontal overlaps and subsequently divides them into cells based on the margin structure of the columns. Wang et al.[24] devised a

2 Related Work

seven-step process for table structure understanding, resembling the X-Y cut algorithm. Their system utilizes probability optimization, with probabilities derived from measurements obtained from a training corpus. Thus, their approach is data-driven. Considering the need for adaptability to diverse input sources, Shigarov et al. [25] proposed a system that offers extensive configuration options for algorithms, thresholds, and rule sets used in table decomposition. Their approach heavily relies on PDF metadata, such as font and character bounding boxes, as well as ad-hoc heuristics.

Recently, Hao et al.[26] presented one of the first methods that applied deep learning techniques to table detection in PDF documents. Their approach included learned features, loose heuristics, and metadata from underlying PDF documents. In contrast to machine learning-based methods, Tran et al.[27] proposed a method that achieved competitive results in a widely recognized 2013 in the ICDAR dataset [28]. They presented in [10] focuses on localization of text components and extraction of text blocks. The height of each block of text is then compared to the average height, and if it matches a set of rules, a region of interest (ROI) is identified as a table. Their approach was based on regions of interest and the spatial arrangement of extracted blocks of text. Notably, their method worked directly with document images. However, since the authors did not disclose which 2013 used part of the ICDAR tabular data set for algorithm development and analysis, a direct comparison with our results is not possible. This restriction also applies to subsequent articles published by the same research group.

Recently, a method called DeepDeSRT[29] has been proposed, which uses deep learning for both table detection and table structure recognition. This method focuses on determining the locations of rows, columns, and cells in discovered tables. in 2013 DeepDeSRT achieved state-of-the-art performance on the ICDAR tabletop competition dataset. In a subsequent study [30], he combined deep convolutional neural networks, graph models, and key concepts to find tables and graphs in documents. Their technique was adapted to the extended 2013 version of the ICDAR tabletop competition dataset and showed better performance compared to existing models. Aswin et al. [31] use formatting hints from semi-structured HTML tables to extract data from web pages. In HTML tables, cells are already separated by tags. Singh et al.[32] use object detection techniques to understand the layout of documents.

3 Background Knowledge

3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) [3] is a collection of two types of layers, i.e., Hidden layers and the Classifier part. The Hidden layers consist of Convolutions, Pooling, Strides, and Dropout layers. The main building block of CNN is the convolutional layer. The convolution operation is applied to our image to encode our pixel data onto a smaller feature map. We have a matrix called a convolution matrix which is also called a kernel. We place this kernel on our actual data and perform an element-wise multiplication to get the convoluted value. We get the entire feature map by shifting our kernel on our data. In Fig. 3.1 the filter is applied at the top left corner of the input. After performing the element-wise multiplication, the output of the convolution operation “4” is stored in the feature map. Then we slide the filter based on the stride value and perform the same operation and store these results in the feature map.

After we apply convolution to our data, we pass it through a pooling layer Fig. 3.2. The Pooling operation decreases the dimensionality of our data. It partitions our original grid into sub-grids of the same size and takes one of the values in that sub-grid as the representative of that sub-grid. That one value we choose from the sub-grid is the maximum value in that sub-grid. We reduce the dimensionality of data without changing the meaning of the data very much. We generally break our data into smaller sub-grids so that the maximum value is a good representative of the whole sub-grid. This tells us the amount by which we shift our kernel on our data grid during convolution. By default, this value is 1. If we want to reduce the overlap between the fields, we can use bigger strides. By using larger strides, we can even reduce the size of the feature map. During training, we randomly drop out some of the learning neurons to prevent overfitting our training data.

The classifier part contains the flow of layers like flatten and fully connected, and these lay-

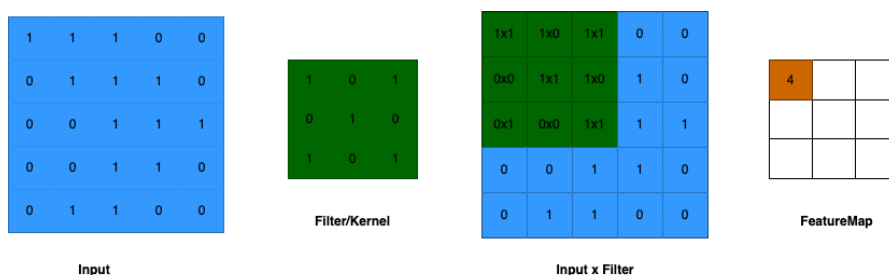


Figure 3.1: Convolutional Operation

3 Background Knowledge

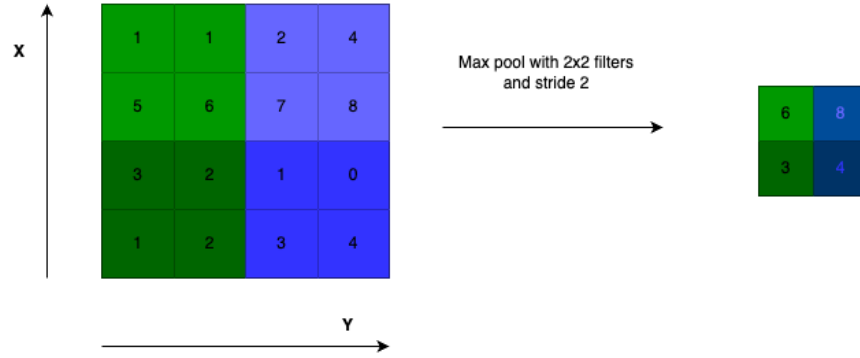


Figure 3.2: Pooling Operation

ers use the features from the output of the convolution and pooling layers to classify the input image based on the training data. After the convolution and pooling layers, the output is forwarded to the flatten layer. The output from the pooling layer is of shape a 4D array. But a fully connected layer takes the input of shape a 2D array. So to convert the 4D array to a 2D array we use the flatten layer. It will reduce the three dimensions of an image into a single dimension. Fully connected layers are also the main part of CNN. It will perform recognizing and classifying tasks. The features of the images from the convolution and pooling layers are passed as input to the fully connected layer, and it derives the final classification decision.

3.2 Object Detection

Object detection [5, 6] is a challenging problem in computer vision and has been the subject of extensive research and development for many years. This task includes several steps. First, the algorithm must identify potential regions of interest in the image and then need to classify those regions into specific object classes and draws a bounding box across their locations Fig. 3.3. One of the key challenges in object detection is the presence of multiple objects of different sizes and orientations in the image. Object appearance, background clutter, and lighting conditions vary widely, making it difficult for object detection algorithms to generalize well across diverse scenarios.

The advent of deep learning, especially convolutional neural networks (CNNs), has greatly improved the accuracy and robustness of object detection. Object detection algorithms based on deep learning can learn hierarchical representations of objects directly from raw image data, allowing them to recognize and use complex patterns and features. This makes them very effective when working with different object variants and background clutter. Common deep learning-based object detection architectures include R-CNN [33], Fast R-CNN [34], Faster R-CNN [35], YOLO (You Only Look Once) [36], and SSD (Single Shot MultiBox Detector) [37]. Each of these models has its own characteristics, but they all strive to find a balance between accuracy and speed.

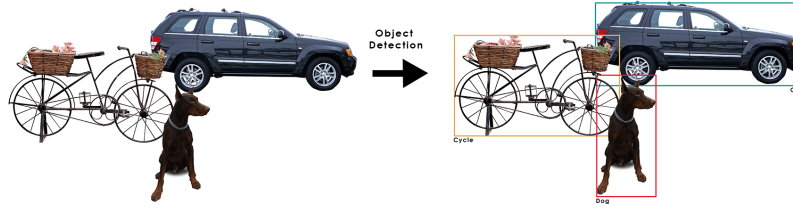


Figure 3.3: Object Detection

Object recognition has many practical applications. In the context of autonomous vehicles [38], object detection is very important to identify pedestrians, other vehicles and obstacles on the road and ensure safe navigation. In Surveillance systems to detect intruders and suspicious activities [37]. Retailers can use object recognition to track inventory on shelves or identify objects in warehouses to optimize logistics [39]. In Medical imaging to diagnose diseases [40] by identifying and analyzing specific structures and abnormalities.

3.3 VGG-19

The VGG19 [4] architecture was proposed by the Visual Geometry Group (VGG) at the University of Oxford in 2014. It is an extension of the VGG16 model, designed to investigate whether increasing the depth of a convolutional neural network would improve its performance on challenging visual recognition tasks. The main objective behind VGG19's development was to explore the relationship between model depth and accuracy, which was a crucial area of research at that time.

VGG19, as the name suggests, comprises 19 layers, making it a relatively deep network compared to its predecessors. The network's architecture in Fig. 3.4 is characterized by its simplicity and uniformity, with a consistent pattern of alternating convolutional layers followed by max-pooling layers. The convolutional layers comprise small 3x3 filters, whereas the max-pooling

3 Background Knowledge

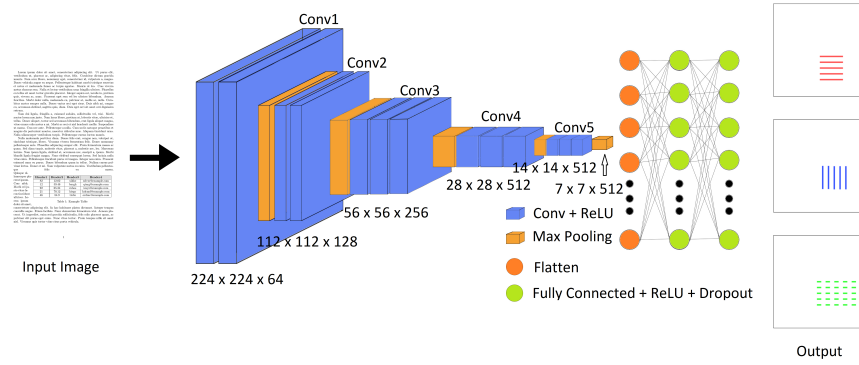


Figure 3.4: VGG19 Architecture

layers use 2x2 pooling windows with a stride of 2. The application of these 3x3 filters periodically throughout the network enables VGG19 to learn rich hierarchical data derived from the input images, contributing to its powerful representation abilities.

One of the notable aspects of VGG19 is that it avoids more complex techniques, such as using different filter sizes or introducing additional non-linearities between convolutional layers. This simplicity and the exclusive use of 3x3 filters make the architecture more interpretable and easier to analyze, providing valuable insights into the inner workings of deep neural networks. The design choice of using 3x3 filters is motivated by several factors. Firstly, stacking multiple 3x3 filters allows the network to learn complex patterns by gradually combining low-level features into higher-level representations. This hierarchical learning approach helps VGG19 effectively capture and represent the spatial hierarchies present in the input images, which is crucial for understanding complex visual scenes.

Secondly, using smaller filters reduces the number of parameters in the network compared to larger filters. While it may initially seem counterintuitive, the consecutive application of 3x3 filters can be seen as an effective way to create a larger receptive field without significantly increasing the model's computational complexity. This design choice was especially advantageous given the limited computational resources available when VGG19 was introduced. The straightforward architecture of VGG19 has made it a highly accessible model for researchers and practitioners to experiment with. The uniformity of its design simplifies the process of modifying and adapting the architecture for various tasks and datasets. Additionally, VGG19 serves as a strong baseline for comparison when developing more complex architectures, allowing researchers to gauge the incremental improvements offered by advanced techniques.

However, VGG19's depth also poses challenges, particularly in terms of computational resources. The extensive number of layers results in a higher demand for memory and processing power during training and inference. As a result, deploying VGG19 on resource-constrained devices or real-time applications may be impractical. Despite this limitation, the impact of VGG19 on the field of computer vision has been profound. It played a significant role in inspiring subsequent research in deeper architectures and transfer learning. The model's ability to extract rich features from images has made it a popular choice for transfer learning applications, where pre-trained VGG19 models are fine-tuned [41] on specific tasks. This approach allows researchers and developers to leverage the network's learned features, saving time and computational resources in new computer vision projects.

In conclusion, VGG19 is a remarkable convolutional neural network architecture that contributed to the advancement of computer vision. Its simplicity, uniformity, and use of 3x3 convolutional filters have provided valuable insights into deep learning design principles. While its computational demands may limit its practicality in some scenarios, VGG19's impact on the field continues to resonate through its influence on subsequent research and its wide application in transfer learning.

3.4 Semantic Segmentation

Semantic segmentation [7] is a sophisticated task in the realm of computer vision, allowing machines to comprehend images at a much more granular level compared to conventional object detection methods. While object detection typically involves drawing bounding boxes around objects, semantic segmentation takes it further by classifying each pixel in an image into specific categories or classes. This provides detailed pixel-wise labeling, essentially assigning meaningful labels to every pixel. This fine-grained analysis finds applications in numerous domains. For instance, in autonomous driving, self-driving cars utilize semantic segmentation to interpret their surroundings accurately. They can segment the road, pedestrians, other vehicles, and various objects, enabling them to make informed decisions and navigate complex road scenarios safely. From the Fig. 3.5, we can observe that the model can segment the table, depicting rows in red, columns in blue, and cell contents in green.

Semantic segmentation relies heavily on deep learning, with convolutional neural networks (CNNs) being the favored choice. These models are trained on extensive datasets containing images with meticulously hand-labeled pixel-level annotations. During training, the model learns to recognize patterns and features associated with different objects and regions, enabling it to generalize and perform segmentation on unseen images.

Despite its advancements, semantic segmentation faces challenges. Achieving pixel-level accuracy is not easy, especially in regions with ambiguous boundaries or overlapping objects. Image noise, occlusions, and variations in lighting can also affect segmentation performance. Additionally, smaller or heavily occluded objects pose difficulties in accurate segmentation.

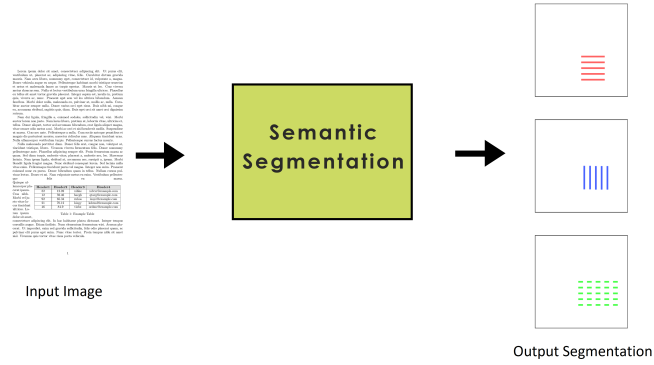


Figure 3.5: Semantic Segmentation

Researchers are actively working to address these challenges and improve segmentation models through various means. Advanced network architectures, novel loss functions, and effective data augmentation techniques have been proposed to enhance accuracy and robustness. Transfer learning, where a pre-trained model is fine-tuned for specific tasks, has also proven effective, particularly in scenarios with limited training data.

In conclusion, semantic segmentation is a critical and rapidly evolving aspect of computer vision. Its ability to provide a detailed, pixel-wise understanding of images has numerous applications across diverse industries, transforming how machines interpret and interact with visual data in the real world. As research progresses, the capabilities of semantic segmentation are expected to continue expanding, making it an essential tool in the AI toolkit.

3.5 Encoder-Decoder

The encoder-decoder architecture [8] stands as a robust and widely embraced solution for handling the intricate task of semantic segmentation in object detection. In this context, semantic segmentation refers to the process of classifying each pixel in an image, assigning it to a specific category like an object or the background. This architecture comprises two fundamental components: the encoder and the decoder. The encoder typically relies on a convolutional neural network (CNN) pretrained on a vast dataset like VGG, ResNet [42], or MobileNet [43]. Its primary function is to extract valuable and distinctive features from the input image. As the image progresses through the convolutional layers, it captures diverse levels of visual details.

The deeper the encoder goes, the more abstract the features become, encompassing both

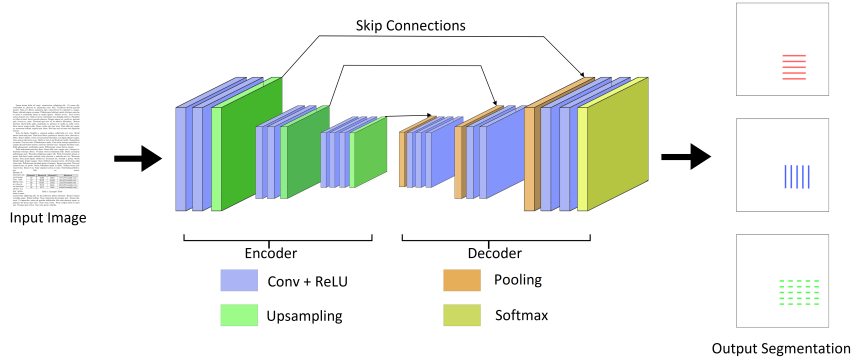


Figure 3.6: Encoder Decoder Architecture

low-level characteristics such as edges and textures and high-level traits representing complex shapes and patterns. This hierarchical representation allows the encoder to encode a comprehensive understanding of the image's content. Following this, the encoder compresses the high-dimensional feature maps into a bottleneck or latent space representation. This bottleneck condenses the feature maps, reducing the spatial dimensions while summarizing essential information from the original image. The latent space acts as a compact yet informative representation essential for efficient computation in the subsequent decoder phase.

The decoder, in turn, takes this compressed representation and reconstructs a segmentation map with the same spatial dimensions as the original input image. It achieves this through techniques like transposed convolutions or interpolation, effectively restoring the fine-grained spatial details lost during the encoder's downsampling. To further enhance the segmentation results and preserve intricate information, the decoder employs skip connections Fig. 3.6. These connections link corresponding feature maps from the encoder to the decoder at varying spatial scales. By doing so, the network benefits from both high-resolution, fine-grained details from the encoder and upsampled feature maps from the decoder. This integration of multi-scale features improves object boundary accuracy and segmentation quality.

Ultimately, the decoder generates a pixel-wise prediction map. Each pixel is classified into distinct semantic categories, successfully identifying objects and their boundaries within the image. This high-resolution prediction map provides a detailed understanding of the image's content, empowering diverse object detection tasks. The encoder-decoder architecture's ability to capture intricate spatial relationships between objects, its hierarchical feature extraction, and fine-grained reconstruction makes it an invaluable asset in the realm of computer vision.

Its adaptability and accuracy contribute to continuous advancements in object detection capabilities.

3.6 TableNet

In earlier approaches to deep learning, detecting tables and columns in documents was treated as two separate problems. When all columns were known beforehand, identifying the table region was straightforward. However, independently searching for columns in a document resulted in many false positives due to their vertical alignment of words or numerical blocks. To enhance column detection, leveraging the knowledge of the tabular region proved beneficial, as tables and columns share common regions. Convolution filters were utilized to detect tables and further reinforced by column-detecting filters, leading to improved model performance.

The TableNet model [9], employing an encoder-decoder model for semantic segmentation, capitalized on this insight. This model employs the same encoder for both table and column detection tasks, while separate decoders are used for each task. The encoding layers used ground truth values for both tables and columns during training, resulting in two distinct computational graphs for training. Input images were converted to RGB format, resized to 1024x1024 resolution, and fed into the model. The model generated output masks with binary target pixel values, indicating whether each pixel belonged to the table, column, or background region.

Detecting tables in documents is akin to recognizing objects in real-world images. A pixel-wise approach was adopted for table and column region anticipation, similar to semantic segmentation advancements. The FCN architecture, initially proposed by Long et al. [44], exhibited remarkable accuracy in semantic segmentation through pixel-wise prediction using encoder-decoder network structures. The skip-pooling technique combined low-resolution feature maps from the decoder network with high-resolution features from encoder networks.

The TableNet architecture Fig. 3.7 comprises a base network, two decoder branches, and a rule-based row extraction module. Pre-trained VGG-19 features were used to initialize the base network for extracting high-level features from input images. Common encoder layers were derived from VGG19's conv1 to pool5, shared by both table and column graphs. After pool5, two (1x1) convolutional layers were introduced, each followed by ReLU activation and a dropout layer (dropout probability of 0.8). Subsequently, two decoder branches were appended, one for table region segmentation and the other for column region segmentation within a table region.

In the table branch, an additional (1x1) convolution layer (conv7 table) was employed, followed by fractionally-strided convolution layers for image upscaling. The output of the conv7 table was up-scaled and integrated with the pool4 pooling layer. Likewise, in the column detection branch, an extra convolution layer (conv7 column) with ReLU activation and a dropout layer (same dropout probability) were added. The feature maps were up-sampled using fractionally-

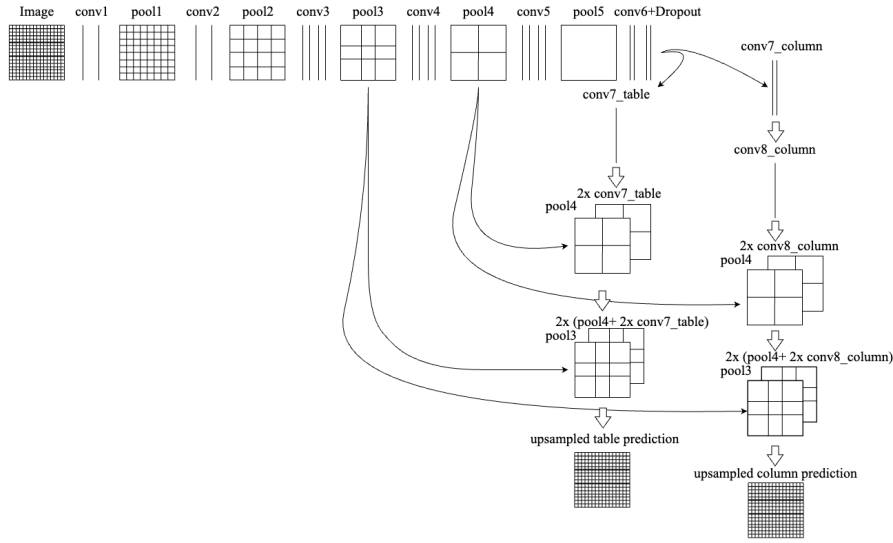


Figure 3.7: TableNet Architecture[9]

strided convolutions, followed by a (1x1) convolution (conv8 column) layer. These up-sampled feature maps were combined with the pool4 pooling layer and further up-sampled while maintaining the same dimension with the pool3 pooling layer. Finally, the feature map was up-scaled to match the original image.

Both branches used multiple (1x1) convolution layers before employing transposed layers to reduce feature map dimension, which improved pixel class prediction. The output layers (from the encoder network) had channels corresponding to the number of classes, with the highest probability channel assigned to corresponding pixels. These output layers were subsequently up-sampled. As a result, the two branches of computational graphs produced masks for table and column regions, enabling effective detection in the semantic segmentation process. This approach shows promising results for table and column detection tasks in documents.

4 Methodology

In this chapter, we discuss the existing Marmot dataset and the methods to augment the data for generating novel training data, along with its respective annotations.

4.1 Dataset

The Marmot dataset has gained wide acceptance in the field of tabular survey evaluation due to its large and diverse content. It was introduced as a large-scale dataset with multiple document images, making it suitable for evaluating the performance of table detection algorithms in real-world situations. This dataset [45] was proposed by the Institute of Computer Science and Technology (Peking University) and afterward developed by Fang et al. [46]. A ratio of almost 1:1 exists between the positive and negative samples in the collection. The Citeseer website was crawled for the English pages. The pages display a wide range of page layouts and table styles. Over 1500 conference and journal papers from various sectors that date from the year 1970 to the year 2011 publications were crawled among them. The English pages are a combination of one-column and two-column layouts, while the e-Book pages are predominantly one-column layouts.

The main strength of the Marmot dataset is its large diversity. Covers many document types within the English language, such as research papers, financial statements, forms, and invoices. This diversity empowers algorithms trained on the dataset to effectively process tables across a wide spectrum of document formats encountered in real-world scenarios. Also worth noting is the realistic nature of the dataset, as it contains images of authentic documents that represent the complexities and challenges of real documents. This matter-of-factness is very important since table detection models must cope with complicated textures, varying font sizes, occlusions, and background noise, all of which commonly occur in real documents.

The Marmot dataset comprises images in .bmp format, and the corresponding annotations for these images are provided in .xml format, as illustrated in the Fig. 4.1 and Fig. 4.2. The document pictures are saved in the .bmp format, and the associated XML files provide the coordinates of the tables in the accompanying images. These XML files are processed to retrieve ground-truth annotations, providing accurate bounding box information for each table area. Researchers can use this baseline data to quantify the performance of their table detection models.

The Marmot dataset offers a variety of table styles, including simple and complex tables with nested cells, nested tables, and irregular layouts. This variety accurately reflects the variety of

4 Methodology

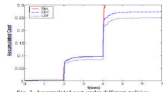


Fig. 2. Accumulated cost over time for three pendulums. The performance of each pendulum is obtained by the following cost function, which measures the error e_i of the pendulum weight w_i from its target value w_i^* .

Note that w_i is the duration of the evolution period, which typically should be less than the period of the pendulum. After executing along, at time t , the value of the cost function is calculated as follows:

B. Results

The simulation we run keeps the following time sequence: at time $t=0$, only the control task controlling the first pendulum is released. After executing along, at time $t=1$, another control task is released to control the second pendulum. Finally the third control task is released at time $t=2$. The three control tasks are parallel until $t=7$. Before the release time of each control task, the corresponding pendulum is in equilibrium, its release time is only controlled by the pendulum length, a parameter that is adjustable for each pendulum. This simulation pattern is repeated for different scheduling algorithms: RM, EDF and LRF. During each experiment, the cost function presented earlier and the reaching schedule are recorded. The cost function evaluation period goes from the beginning of each simulation, $t=0$, to the simulation completion, at $t=7$. For each scheduling policy, the results are obtained on an averaged of 1000 runs.

RM. RM is a static scheduling algorithm in open loop, which assigns priorities to tasks according to their release times. The cost function values for the three pendulums are shown in the Figure 2 and Table 1. Figure 2 shows the accumulated cost function, J , during the evolution period for each scheduling policy. Under RM the schedulability condition for our experiment is given by $U < 0.79$. Taking into account that task execution time is denoted by e_i , the processor utilization is 0.71 and the control performance is good. From Fig. 2, the cost function curve is profiled and

their common 0.99 CPU. That is, both axes deadlines and the inverted pendulum 1 falls down. This explains the fact that the cost function in Figure 2 goes to infinity. Note that under RM, the task is not schedulable.

EDF. EDF is a dynamic algorithm in open loop, which assigns priorities to tasks according to their schedulability condition is given by $U < 1$. For our simulation, since $U < 1$, the task set is schedulable, and the three pendulums can be controlled as it can be seen in Figure 2: the accumulated cost reaches a finite value, which means that the system is controlled by each pendulum that affected each of the three pendulums could be adequately controlled. The performance achieved by EDF is also given in Figure 2 in terms of the cost function, reaching a value of 0.2752 at the completion of the evolution, ending at $t=7$.

LRF. LRF is a dynamic scheduling algorithm in open loop, which adjusts the schedule based on consistent feedback of each control loop. Under this scheduler, in the simulation we observe that the three inverted pendulums can be perfectly controlled. As it can be seen in Figure 2, the cost function of LRF goes to 0, which means that the system is controlled by each pendulum that affected each of the three pendulums could be adequately controlled. LRF performs better than EDF. Note that the final cost of LRF is 0.2490.

C. Discussion

The exact cost for each one of the three pendulums for each scheduling algorithm is summarized in the Table 1. RM fails in controlling the three pendulums. EDF and LRF are able to control the three pendulums. However, EDF gives the best control in terms of reaching schedule.

This result shows that scheduling policies that take advantage of the application dynamics and are able to adjust the schedule accordingly can provide, in some specific scenarios, better performance in terms of the application control performance in our case.

For each scheduling algorithm we identified one function 1) derive main negative aspects: low and CPU utilization, the lack of feedback, disturbance, and poor response CPU utilization in the sense that control tasks are executed only when they are released, they are not executed when the control phase, after disturbance. Otherwise, they execute with the lowest priority rate in order to give more to other tasks with higher priority. In addition, our approach is based on the idea of feedback, which offers the possibility of making at run time the



Figure 4.1: Sample images from MARMOT dataset and its respective ground truth coordinates

tables in various documents. By revealing patterns for various table structures, the dataset encourages the development of algorithms that can efficiently handle tables of different sizes. In addition, the dataset contains ground-truth annotations, where each image in the document is manually annotated with precise bounding boxes around the table regions. These precise annotations are the gold standard for evaluating the performance of table tracking models. Researchers can comprehensively and objectively evaluate their algorithms by comparing model predictions to the ground truth.

However, it is essential to acknowledge the limitations of the data set. One drawback is the limited display of certain domains or rare table styles. Therefore, the performance of the model in these less common structures cannot be properly evaluated. Also, manual annotation can add a degree of subjectivity. While an effort is made to maintain consistency throughout the annotation, individual commentators may interpret the boundaries of the table slightly differently, resulting in minor differences. These variations in the ground truth can affect the evaluation results and introduce potential biases in the evaluation algorithms.

		Source		
		LESS 1988	LESS 1995	LESS 1997
Male	Urban	75.0	69.9	72.8
	Rural	79.2	76.3	75.4
Female	Urban	29.3	19.5	26.2
	Rural	53.9	23.0	17.3

Table 3—Unemployment rate, by sex, education, and region, economically active population aged 15–64

Table 3—Unemployment rate, by sex, education, and region, economically active population aged 15–64

	LFSS 1995				EHBS 1997			
	With search		Without search		With search		Without search	
	Urban	Rural	Urban	Rural	Urban	Rural	Urban	Rural
Males								
Below secondary	1.3	0.4	0.7	5.4	2.0	3.3	7.3	3.3
Secondary and above	14.1	28.6	18.9	11.3	12.7	11.9	13.6	16.8
Total	7.6	7.5	7.5	8.4	5.4	6.8	10.1	7.7
Females								
Below secondary	4.1	0.3	0.7	9.1	6.5	7.5	17.3	21.8
Secondary and above	30.9	57.7	40.0	23.2	40.5	28.5	31.1	53.2
Total	27.6	21.4	24.0	20.1	23.0	21.3	28.1	37.5

[illegible]

3.1 The Naive Approach

Initially (for the first few points) this approach may be successful in detecting outliers/change points. As more points come in, RLS estimates the model parameters considering all the points seen so far, and thus it gives equal weight to all the points in the unrestricted window. As a result, the changes in the dynamics of the system may be hidden by the dominance of the older points. Figure 3.1 illustrates this phenomenon. As a result RLS by itself is not adequate. The problem is due to the fact that RLS tries to

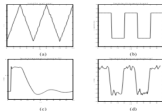


Figure 1. Effect of older points in RLS-based outlier/change point detection. (a) saw-tooth data stream. (b) slopes of the line segments in the saw-tooth. (c) slope estimate using RLS. (d) slope estimate using a moving win-

3.2 Moving Window Approach

One approach to forget older points is to have a moving window over the timeseries' data stream and estimate the model over this window. At the same time, we would like to maintain the efficiency of the RLS algorithm.

3.2.1 Using Forward-Backward RL

One method to achieve this would be to derive a backward recursion and alternately use the forward and backward recursions as follows. Suppose the current window spans $[t_1, \dots, t_n]$. When a new data point comes in at t_{n+1} , we use the forward recursion (equations 7–9) to derive a model for the points in $[t_1, \dots, t_n, t_{n+1}]$. Then, we use the backward recursion to remove the effects of the point at t_n , and thus get a model for the points in $[t_2, \dots, t_n, t_{n+1}]$.

In what follows, we derive the backward recursion for RLS. Suppose we have $\hat{\theta}_{RLS}$ and P_{RLS} over certain data S , where

$$F_{\text{tot}} = \left\{ \sum_{k \in S} \varphi(k) \varphi^*(k) \right\}^{-1} \quad (10)$$

[illegible]

```

1 <annotation>
2   <folder>MARROT_ANNOTATION</folder>
3
4   <filename>10.1.1.1.2014.4.bmp</filename>
5
6   <path>h:\msk\ Desktop\MARROT_ANNOTATION\10.1.1.1.2014.4.bmp</path>
7
8   <source>
9     <database>Unknown</database>
10
11   </source>
12
13   <size>
14     <width>816</width>
15     <height>1056</height>
16     <depth>1</depth>
17
18   </size>
19
20   <segmented>0</segmented>
21
22   <object>
23     <name>column</name>
24     <pose>Unspecified</pose>
25     <truncated>0</truncated>
26     <difficult>0</difficult>
27     <bndbox>
28
29       <xmin>72</xmin>
30       <ymin>264</ymin>
31       <xmax>162</xmax>
32       <ymax>685</ymax>
33
34     </bndbox>
35
36   </object>
37
38   <object>
39     <name>column</name>
40     <pose>Unspecified</pose>
41     <truncated>0</truncated>
42     <difficult>0</difficult>
43     <bndbox>
44
45       <xmin>181</xmin>
46       <ymin>263</ymin>
47       <xmax>375</xmax>
48       <ymax>689</ymax>
49
50     </bndbox>
51
52   </object>
53
54 </annotation>

```

17

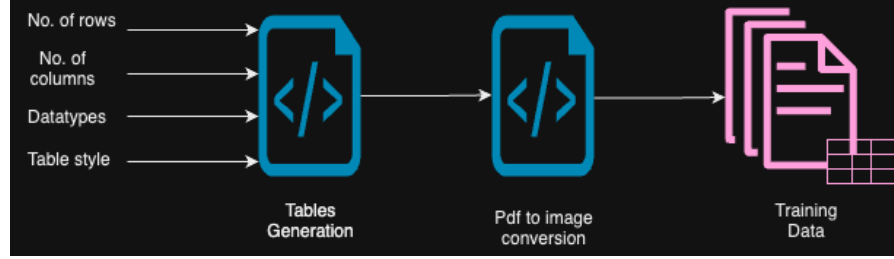


Figure 4.3: Workflow for generation of training data

4.2 Novel Training Data

The Marmot dataset is widely recognized as an important benchmark for evaluating table recognition algorithms in real-world situations. However, the manual annotation process for this dataset has the significant drawback of being time and resource intensive. A new novel training approach was introduced to overcome this limitation and facilitate the creation of more effective table detection models. The purpose of this innovative approach is to generate a wide range of training data automatically.

The crux of this novel approach lies in the autogeneration of training data, where tables are surrounded by text, and arbitrary data fills the table cells. Additionally, precise coordinates for the table regions are provided, facilitating accurate ground truth annotations. The workflow for generating novel training data is illustrated in Fig. 4.3. Generating this novel data harnesses the capabilities of LaTeX, a typesetting system often employed in document preparation. By utilizing LaTeX, images with tables surrounded by text are produced, mirroring the layout of real documents. To enhance the realism of the generated data, the well-known Lorem Ipsum package from LaTeX is employed to generate random text around the tables. This approach simulates the natural context in which tables typically appear within documents or journals, adding valuable diversity to the dataset. The LaTeX output, conveniently provided in PDF format, is then processed using `pdf_to_images` module to yield the desired images for training the table detection models.

The novelty of this data generation process lies in the focus on six distinct styles of tables, as shown in Fig. 4.4, each with its unique characteristics. These styles encompass tables with dashed lines, tables without vertical borders, tables featuring colored headers, tables devoid of top, mid, and bottom rules, tables following the booktabs style, and finally, data with no tables at all. Each autogenerated table is filled with arbitrary data, offering an array of table styles to enrich the dataset.

This method provides a high degree of freedom in generating various table data and allows users extensive control over multiple parameters to create tables tailored to their specific needs. The key parameters that the user can manipulate include the number of rows, the number of columns, the datatypes, and the style of the tables. The number of rows and columns can be

Header1	Header2	Header3	Header4
82	12.02	xihhe	xdvvr@example.com
12	59.46	bacgh	qtarg@example.com
92	65.34	rnhos	izayz@example.com
21	70.12	kiegy	kdcmd@example.com
46	84.9	tizbx	ordmc@example.com

Table 1: Example Table

Header1	Header2	Header3	Header4
82	12.02	xihhe	xdvvr@example.com
12	59.46	bacgh	qtarg@example.com
92	65.34	rnhos	izayz@example.com
21	70.12	kiegy	kdcmd@example.com
46	84.9	tizbx	ordmc@example.com

Table 1: Example Table

Header1	Header2	Header3	Header4
82	12.02	xihhe	xdvvr@example.com
12	59.46	bacgh	qtarg@example.com
92	65.34	rnhos	izayz@example.com
21	70.12	kiegy	kdcmd@example.com
46	84.9	tizbx	ordmc@example.com

Table 1: Example Table

Header1	Header2	Header3	Header4
82	12.02	xihhe	xdvvr@example.com
12	59.46	bacgh	qtarg@example.com
92	65.34	rnhos	izayz@example.com
21	70.12	kiegy	kdcmd@example.com
46	84.9	tizbx	ordmc@example.com

Table 1: Example Table

Header1	Header2	Header3	Header4
82	12.02	xihhe	xdvvr@example.com
12	59.46	bacgh	qtarg@example.com
92	65.34	rnhos	izayz@example.com
21	70.12	kiegy	kdcmd@example.com
46	84.9	tizbx	ordmc@example.com

Table 1: Example Table

Figure 4.4: Tables with different styles: dashed lines, without vertical borders, colored headers, devoid of top, mid, and bottom rules, and booktabs

specified as integer values, giving users the flexibility to define the size of the table according to their needs. This method limits these parameters so that only integer values are accepted to preserve data integrity. Users also have the freedom to determine the data types that populate the table. The Available options, such as integer, float, text, and email, give users the flexibility to simulate different data scenarios. Specific scopes and properties for each data type, such as a random 5-character lowercase text sequence for text and a 17-character lowercase email string for email, contribute to the reliability and diversity of the data generated. We generate random text wrapped by the table using the Lorem Ipsum package. The table is positioned at a random location based on the alignment of the random text generated by the Lorem Ipsum package. This random text is wrapped around the table using the wrapfig package. Additionally, this method provides a selection of formatting styles that the user can select to customize the appearance of the generated table. The freedom to choose from table styles such as `no_vertical_lines`, `dashed_lines`, `colored_headers`, `no_top_mid_bottom_rules`, and `booktabs` allows users to customize the appearance of the table to the intended use case.

The automated generation of data overcomes the laborious process of manual annotation, saving considerable time and effort for researchers. By efficiently generating diverse and realistic table layouts, this novel approach facilitates a more comprehensive evaluation of table detection algorithms. Moreover, as the Marmot dataset itself comprises a limited number of images, this approach significantly augments the dataset, substantially increasing the pool of

training examples available. The expanded dataset enhances the training process, enabling models to be trained on a more extensive and diverse set of data. As a result, the trained models exhibit improved detection performance and greater generalization abilities. By leveraging this novel data for training, researchers can further develop robust table detection algorithms that excel in detecting various table structures encountered in real-world documents.

The diversity introduced by the autogenerated data promotes versatility in table detection models. The exposure to multiple table styles enables models to adapt to a wide range of table representations, a crucial factor in ensuring real-world applicability. Through this diverse dataset, researchers can gain deeper insights into the strengths and weaknesses of their table detection algorithms, facilitating iterative improvements. The novel approach of autogenerating data for table detection using the Marmot dataset fills a significant gap in the realm of document analysis. By automating data generation and expanding the dataset, this approach accelerates the progress of table detection research. The wide variety of table styles incorporated in the generated data enhances the adaptability of models, leading to improved detection results and fostering advancements in the domain of table detection and document analysis. With this novel approach, researchers are empowered to develop more robust and versatile table detection algorithms that hold immense potential for practical applications.

4.3 Generating Co-ordinates of the Table

Generating precise coordinates for the tables present in the images is an indispensable step in the evaluation process of table detection algorithms, particularly in the context of utilizing novel training data. Accurate ground truth values of the table's locations are essential for thorough and reliable performance assessment. To achieve this, generating two distinct types of images for each table instance is initiated. The first type consists of the table surrounded by relevant text, simulating the context in which tables often appear within real documents. The second type comprises images with only the table itself, omitting any surrounding text or extraneous page elements, which is achieved by replacing the text with a white color.

Subsequently, the focus narrows down to the images featuring only the table. The goal is to precisely determine the table's exact location within the image and outline its boundaries with utmost accuracy. This localization process involves analyzing the distribution of black pixels within the table. These black pixels essentially form the table's structural outline, enabling a delineation of its shape. By meticulously examining the spatial distribution of these pixels, the exact coordinates of the table's starting position (min_x, min_y) where the black pixels begin, i.e., upper leftmost black pixel and ending position (max_x, max_y) where the black pixels end, i.e., lower right-most black pixel as shown in Fig. 4.5 are derived which describe the bounding box of the table. These coordinates serve as the ground truth values, providing an indisputable reference for the subsequent evaluation of the table detection algorithms.

Through this rigorous approach, the generated training data becomes enriched with highly accurate ground truth values for each table. This accuracy ensures that the algorithms are thor-

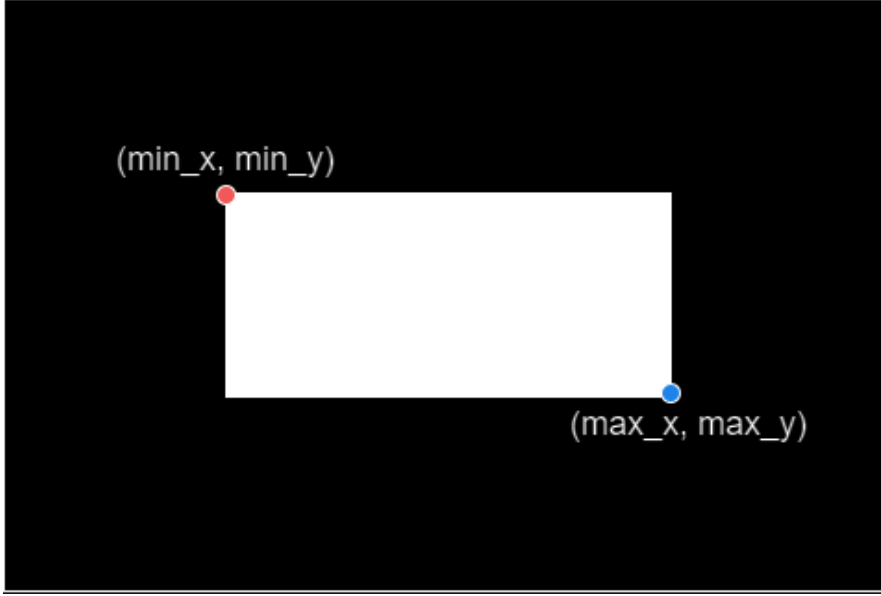


Figure 4.5: Determining Coordinates of the autogenerated Novel Data

oughly assessed in detecting tables with precision, even in the presence of complex document layouts and diverse table structures. Consequently, researchers gain valuable insights into the algorithms' efficacy and adaptability across various real-world scenarios. Moreover, the precise ground truth values act as a solid foundation for training the table detection models. The models perform with exceptional accuracy and robustness by making use of this refined and comprehensive dataset. Hence the development of more advanced and versatile table detection algorithms is encouraged, which directly enables practical applications in diverse fields such as retrieval of information, data extraction, and document analysis.

The value and reliability of the novel training data are enhanced through the meticulous generation of accurate coordinates for the tables present in the images. Combining accurate ground truth values and diverse table styles facilitates a more thorough evaluation of table detection algorithms. Ultimately, this approach empowers researchers to push the boundaries of table detection capabilities, unlocking the potential for advancements in document analysis and contributing to the broader landscape of artificial intelligence research.

In this chapter, we have examined the advantages and disadvantages of the Marmot dataset. Manual data annotation requires significant time and effort. We opted to generate a novel training dataset. This dataset generates tables with various styles based on input parameters such as the number of rows, columns, data types, and table styles. The tables are populated with arbitrary data and wrapped by text to replicate the structure of the Marmot training data. We utilized LaTeX and the Lorem Ipsum package for data augmentation purposes. For the generated images, the ground truth coordinates were calculated by analyzing similar images without text. The upper-leftmost black pixel in that image determined (min_x, min_y) , while

4 Methodology

the lower-rightmost black pixel determined (max_x, max_y) . Utilizing these coordinates, table masks were generated for the novel training data. These masks enable the training of models for table detection tasks and facilitate the evaluation of state-of-the-art architectures using the autogenerated novel training data.

5 Implementation

In the implementation section, we devised a unique approach to generate diverse training data by incorporating various table styles based on user input. We then determined the ground truth coordinates for this novel training data. Subsequently, the model underwent training using the Marmot dataset, comprising annotated images with bounding boxes marking the tables. Post-processing techniques, including noise removal, were then applied to enhance the quality of predicted images. For evaluating the model's performance, we employed the Intersection over Union (IoU) [47, 48, 35, 36] and Bitwise XOR metrics to gauge the accuracy of predicted table regions against ground truth annotations. To achieve precise and robust table detection results, we conducted extensive fine-tuning and hyperparameter optimization.

5.1 Architecture

The architecture of the proposed system encompasses several crucial steps, each contributing to the generation of tables, preprocessing the Marmot dataset, and evaluating TableNet Architecture on the autogenerated novel training dataset. The initial phase involves preprocessing the Marmot dataset [45] as it comes with XML annotations. We utilize `data_preprocessing` module to generate accurate ground truth annotations for column and table masks in the Marmot dataset. In this architecture Fig. 5.1, the number of rows, columns, data types, and table styles are passed to the `tables_generation` module. This module uses LaTeX to generate various PDF images simulating tables that appear in scientific documents and journals. The tables are populated with arbitrary data based on the input parameters provided. These tables are surrounded by text to mimic the appearance of tables in Scientific papers and journals. This is achieved by utilizing the 'Lorem Ipsum' package. These PDF files are processed into images using the `pdf_to_images` module.

To avoid the time-consuming task of manually annotating the autogenerated images, the `generating_ground_truth_coordinates` module is employed to generate ground truth coordinates. These autogenerated novel images can be used to train or test a TableNet model via the `tablenet` or `predict` module. Denoising techniques such as Gaussian blur and thresholding are applied to improve the accuracy of the prediction mask. This step ensures that the resulting table masks are evaluated accurately. After noise removal, the `evaluation_pipeline` module comes into play. This module uses Intersection over Union (IOU) and Bitwise XOR as performance metrics. IOU [47, 48, 35, 36] measures the area of overlap between the ground truth and the predicted mask, and Bitwise XOR compares pixel-level differences between the two. Bitwise XOR is advantageous because it compares each pixel and computes the error rate, giving the percentage of pixels that were mispredicted.

5 Implementation

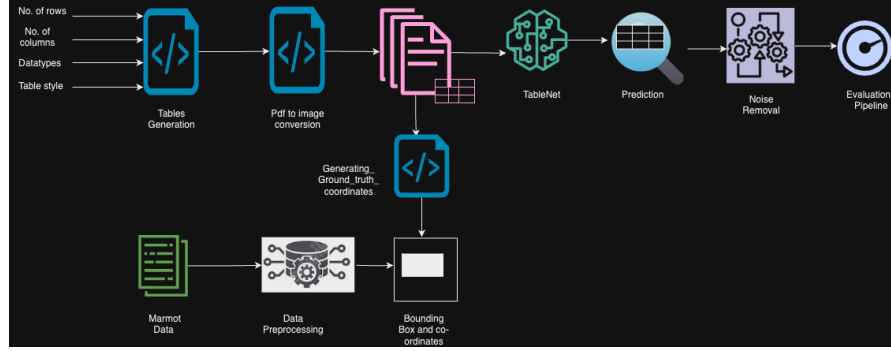


Figure 5.1: Architecture

5.2 Tables_Generation

The `tables_generation` module plays a pivotal role in facilitating the generation of various types of images that closely resemble tables typically found in journals or scientific documents. By harnessing the power of LaTeX, this versatile module opens up a world of possibilities for users in the field of table detection. With the `tables_generation` module, users gain complete control over the characteristics of the generated table images. Inputting desired row and column numbers enables them to tailor the table's dimensions to align precisely with their specific requirements. Moreover, the module grants the ability to specify data types for each column, encompassing integers, floats, text, and email formats. Leveraging this information, the module infuses the tables with arbitrary data, resulting in a wide range of table representations that mimic real-world scenarios.

This module takes the number of rows and columns, data types, and table styles as input. The number of rows and columns is given as integers. Data types can be provided as a list, including options such as integer, float, text, and email, based on the user's choice. Arbitrary data is filled into the generated tables depending on these data types. Random integer and float values are generated within the range of 1 to 1000. Random text is generated as the content of table cells, with a length of 5 characters, and similarly, random email addresses are created with a length of 17 characters. Based on this random arbitrary data and the desired style of the table, tables are generated using LaTeX. We generate random text wrapped by the table using the Lorem Ipsum package. The table is positioned at a random location based on the alignment of the random text generated by the Lorem Ipsum package. This random text is wrapped around the table using the `wrapfig` package [49]. We have utilized the `wraptable` environment from the `wrapfig` package, which takes two parameters. The first parameter specifies the alignment, which can be `l`, `r`, `c`, `i`, or `o` for left, right, center, inner, and outer, respectively. The second parameter is the width of the table container.

In addition to customizable table content, one can select from various table styles when using the `tables_generation` module. Whether they require tables with "dashed_lines", "no_vertical_lines", "colored_headers", "no_horizontal_lines", or "booktabs" style, this module

can effortlessly produce tables that closely resemble those found in Journals or scientific publications. However, what truly sets the `tables_generation` module apart is its ability to create realistic table images that appear embedded in context, just like they are often presented in journals or scientific articles. This is achieved by leveraging the renowned "Lorem Ipsum" package from LaTeX, which automatically generates random and coherent text surrounding the tables as illustrated in Fig. 5.2. As a result, the tables are seamlessly integrated into the generated context, adding an element of authenticity to the images.

A key aspect that enhances the utility of the `tables_generation` module is its capability to generate two different images for each input. One image showcases the table with the context wrapped around it as depicted in Fig. 5.3. In contrast, the other image solely displays the table itself without a title of the table and page number, as depicted in Fig. 5.4. This duality is crucial in accurately determining the ground truth coordinates of the table within the image. By providing this valuable information, one can fine-tune their table detection models and assess their performance more precisely. The profound impact of the `tables_generation` module extends far beyond its capacity to produce realistic and diverse training data. Through harnessing this potent tool, users surmount the constraints imposed by existing small-scale datasets like Marmot [45] and ICDAR [50], which often stifle the progress of table detection research. With an augmented pool of training data at their disposal, users can effectively evaluate a plethora of state-of-the-art table detection architectures, instigating advancements in the field and empowering the development of more precise and resilient table detection models. In conclusion, the `tables_generation` module emerges as a crucial asset in the quest to advance table detection. Its versatility, coupled with the customization options it offers, empowers users to explore new possibilities and enhance their understanding of table detection algorithms. By generating realistic and context-rich table images, this module paves the way for more efficient and accurate table detection methods, contributing to the broader landscape of document analysis and information extraction.

5.3 Data_Preprocessing For MARMOT Dataset

The Marmot dataset [45] contains images in .bmp format along with annotations of the tables present in each image in .xml format. To train the TableNet [9] model, we need to preprocess these XML files and generate the corresponding table and column masks for each image. The XML files contain vital information about the bounding box coordinates of individual columns within an image dataset. By parsing these XML files, one needs to extract the width and height of the images, providing essential dimensions for further processing. Subsequently, we initialize two grayscale image arrays, one for column masks and the other for table masks. These arrays are set to zeros, preparing them to act as filters to highlight specific regions of interest within the images.

The core part of this module lies in the iteration through each column entry in the XML file. For each column, the bounding box coordinates (x_{min} , y_{min} , x_{max} , y_{max}) are extracted,

5 Implementation

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Header1	Header2	Header3	Header4
82	12.02	xihhe	xdvyr@example.com
12	59.46	bacgh	qtarg@example.com
92	65.34	rnhos	izayz@example.com
21	70.12	kiegy	kdcmd@example.com
46	84.9	tizbx	ordmc@example.com

Table 1: Example Table

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultrices et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Figure 5.2: Table in Booktabs style surrounded by text

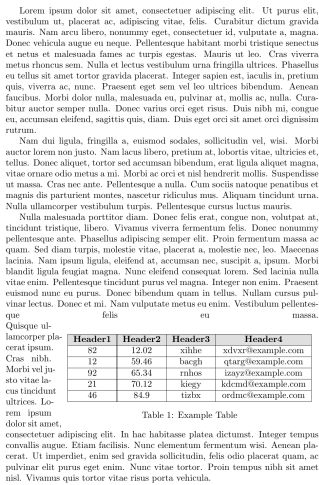


Figure 5.4: Table not surrounded by text

Figure 5.4 shows a table not surrounded by text. The table has four columns: Header1, Header2, Header3, and Header4. The data rows are as follows:

Header1	Header2	Header3	Header4
82	12.02	xbhle	xvcr@example.com
12	59.46	baegh	qtarg@example.com
92	65.34	mbow	iazr@example.com
21	70.12	kleeg	kalcml@example.com
46	84.9	tiazbx	ordmc@example.com

and the corresponding pixels in the column mask are set to 255, turning them white to mark the region encompassing each column. In addition to column segmentation, this module intelligently determines if the columns belong to the same table. To achieve this, it analyzes the vertical distance between the bounding boxes of adjacent columns. If two columns are deemed to belong to the same table, the table mask is updated to highlight the region encompassing these columns. This process continues for all columns in the XML file until all relevant regions are effectively marked and extracted in both the column and table masks. Once the iteration is complete, the final column and table masks are saved as JPEG images in separate directories.

5.4 Generating_Ground_Truth_Coordinates

In the context of table detection, the `tables_generation` module produces images that exclusively feature the tables without any accompanying surrounding text. We employ a series of image processing techniques to conduct a thorough analysis of these images and precisely identify the table's location. Initially, we transform the output image from the `tables_generation` module into a binary representation. This conversion simplifies the image to a single channel, where the table and its contents are rendered in black while the background appears white. This binary representation provides a clear distinction between the table region and the rest of the image, laying the foundation for subsequent steps.

5 Implementation

Next, the process of pinpointing the precise coordinates of the table within the binary image unfolds. By identifying the pixels that are not equal to 255, we effectively isolate the black pixels, which represent the table and its contents. Leveraging the numpy function `"np.min()"`, we effortlessly locate the starting coordinates where the black pixels initiate, unveiling the exact position where the table commences within the image. Similarly, through the use of `"np.max()"`, we determine the ending coordinates where the black pixels terminate, marking the point where the table concludes. With the minimum and maximum coordinates now available, we proceed to define the bounding box that encapsulates the entire table region. This bounding box effectively outlines the spatial extent of the table within the image, acting as a fundamental reference for subsequent processing steps.

Subsequently, the `"cv2.rectangle()"` function comes into play, enabling the creation of a binary mask based on the bounding box coordinates. Within this binary mask, pixels that fall inside the bounding box are assigned a value of 255, representing the color white. In contrast, pixels located outside the bounding box are assigned a value of 0, representing the color black. This binary mask serves as a powerful tool, as it accurately isolates and highlights the region of interest while effectively suppressing all other parts of the image. In the final stage of the process, the binary mask is converted into a PIL image using the `"Image.fromarray()"` function. This conversion facilitates easy visualization and further manipulation of the binary mask using the capabilities provided by the Python Imaging Library (PIL). By meticulously following this process, we successfully generate a binary mask that precisely delineates the table's location within the original image. This vital step in isolating the table region holds significant importance, as it sets the stage for subsequent tasks such as table detection, segmentation, or any further in-depth analysis. Armed with this enhanced understanding of the table's structure and content, we pave the way for developing of advanced and highly effective table detection methodologies, thereby advancing the state-of-the-art in document analysis and information extraction.

5.5 TableNet

This module implements a TableNet [9] model using TensorFlow, a popular deep-learning framework for building and training neural networks. TableNet is a state-of-the-art model designed for table and column segmentation in grayscale images. The model is trained on a Marmot dataset consisting of XML files with bounding box annotations for tables and columns. Before training the model, the dataset is preprocessed to prepare the images and corresponding masks. The images are normalized to a range between 0 and 1 by dividing the pixel values by 255, making them suitable for training. For decoding and resizing the images and masks to the required dimensions of 256 x 256 pixels and 1024 x 1024 pixels, this module makes use of TensorFlow functions such as `"tf.image.decode_bmp()"`, `"tf.image.decode_jpeg()"`, `"tf.image.decode_png()"` and `"tf.image.resize()"`. This ensures that the network's input size is uniform.

We utilized the `"tf.image.decode_bmp()"` function from TensorFlow's `"tf.image"` module that

decodes a BMP image from a string tensor. It takes a binary version of a BMP image and returns a decoded image in tensor form. This image is stored as a 3D array with size dimensions like height, width, and color depth. For instance, RGB images have a color depth of 3, while grayscale images have a depth of 1. The result's data type is typically "tf.uint8". similar to "tf.image.decode_bmp()" function, "tf.image.decode_jpeg()" is another function in TensorFlow that decodes JPEG images from a tensor containing JPEG-formatted data. The output is a tensor holding the decoded image. The tensor is a multi-dimensional array with a shape of [height, width, depth], where depth represents the number of color channels in the image. The tf.image.decode_png() function plays an important role in our processing pipeline because it allows us to convert PNG encoded images into a format suitable for TensorFlow operations. This function takes a PNG image file in compressed form and decodes it, and transforms it into a tensor with pixel values and color channels, which can be further manipulated and used as input for various deep-learning models.

The architecture of the TableNet [9] model is based on a pre-trained VGG-19 [4] base, a well-known convolutional neural network (CNN) model. The VGG-19 base is employed as a feature extractor, which helps capture high-level features from the input images. An RGB image with an input resolution of 1024 x 1024 or 256 x 256 is passed as input to TableNet. The Marmot dataset is divided into 90% for training and 10% for the test dataset. The model then branches into two decoder paths for table and column segmentation. These decoder paths consist of convolutional layers and upsampling layers to upscale the feature maps, combined with intermediate feature maps from the VGG-19 base using concatenation.

The model is trained to minimize the segmentation error using a custom loss function. The loss function is defined as Sparse Categorical Crossentropy, which is commonly used in multi-class segmentation tasks like this one. The code employs this loss function separately for both table and column outputs, allowing the model to learn to accurately segment these structures. During training, the model uses the Adam optimizer [51], a popular choice for deep learning tasks, with a learning rate of 0.0001 and epsilon value of 1e-08. The Adam optimizer efficiently adjusts the model's weights to minimize the loss function and improve segmentation accuracy.

To monitor the training progress and prevent overfitting, the code includes several callbacks. The DisplayCallback class is used to visualize model predictions on sample images after each epoch, allowing easy assessment of segmentation performance. Additionally, the ModelCheckpoint and EarlyStopping callbacks are employed to save the best model checkpoints based on validation loss and implement early stopping if validation loss stops improving.

In summary, this module effectively implements the TableNet model using TensorFlow, training it on a Marmot dataset for table and column segmentation in grayscale images. TensorFlow functions such as "tf.image.decode_bmp" and "tf.image.resize" are utilized for image preprocessing. The model is constructed with a VGG-19 base and custom decoder paths, and it is trained with the Adam optimizer using Sparse Categorical Crossentropy as the loss function. The training process is monitored with callbacks, providing insights into the model's performance and ensuring optimal segmentation results.

5.6 Predict

This module implements a prediction pipeline using TensorFlow to predict table and column masks on a collection of PNG images stored in a specified folder. Even though the Marmot dataset [45] consists of input images in .bmp format, we choose to store the input images in .png format instead of .jpeg and .bmp formats. This decision is based on the characteristics of these formats. JPEG is a lossy compression format that sacrifices some image details to achieve higher compression rates, making it suitable for photographs and images with gradients. On the other hand, BMP is an uncompressed format that retains all image data without any compression, resulting in larger file sizes compared to compressed formats. Meanwhile, PNG is a lossless compression format that preserves all image data without any loss of quality. It employs techniques like "deflate" compression to reduce file size while maintaining quality. Consequently, we opt to save the input images in .png format to avoid introducing artifacts and ensure image quality preservation. This choice allows us to strike a balance between file size and image fidelity.

The pipeline makes use of a custom-trained TableNet model to segment tables and columns in each input image. Firstly, this module defines the `create_mask()` function, which processes the output predictions of the TableNet model to generate separate table and column masks. Additionally, the `show_predictions()` function is provided to visualize the input image along with its corresponding table and column masks. Both these functions play a crucial role in the later stages of the pipeline.

The `predict_mask()` function is at the core of this pipeline. It processes all PNG images found in the specified folder, resizes them to the desired dimensions (256 x 256) or (1024 x 1024), such as the input size of the TableNet model, and then feeds each image into the custom-trained TableNet model for prediction. The generated table and column masks for each image are stored in separate lists, which are later returned by the function. To demonstrate the pipeline's functionality, a sample folder path to predict `"/dataset/test.data"` and image dimensions (256x256) are to be given as input to this function. This module takes these inputs, calls the `predict_mask()` function, and retrieves the predictions. Finally, the `display_predictions()` function is utilized to display all the input images along with their respective table and column masks. All in all, an end-to-end pipeline for the utilization of TableNet, a custom-trained deep learning model that predicts the table and column masks for group of PNG images is provided by this module. By providing images through this pipeline, the analysis and visualization of the table and columns can be efficiently done by the users within their dataset. This implementation helps users to manage the automatic segmentation of tables and columns in their image data, which is very helpful for a wide range of image analysis tasks that may include document processing, data extraction, and much more.

5.7 Noise_Removal

Through the use of this pipeline, we observe that the table mask prediction generated by the `predict` module, unfortunately contains a substantial number of artifacts. To overcome this issue, we perform a series of steps to improve the correctness of the predicted table mask. Initially, we transform the table mask into a binary image, thus making the subsequent operations easier. From the OpenCv library, we utilize the `"cv2.GaussianBlur()"` function to perform Gaussian blur [52] on the table mask to remove the artifacts present in the predicted mask. Onwards, to distinguish between the table and the background artifacts correctly, we manually apply a threshold operation to the binary image. Thus, this series of steps results in the smoothing of the image and thereby reduces the artifacts present in the image significantly and filters the predicted table mask.

5.8 Evaluation_Pipeline

For evaluating the performance of the TableNet model, we utilize two necessary performance metrics: Intersection over Union (IoU) and Bitwise XOR. These criteria provide a quantitative assessment of the model's capabilities. Intersection over Union (IoU) is extensively used for measuring image segmentation tasks. It gauges the degree of overlap between predicted and actual segmentation masks. Bitwise XOR presents another evaluation metric, focusing on the pixel-wise comparison of predicted and actual masks. Through these performance metrics, we can quantitatively evaluate the TableNet model's proficiency in generating table and column masks. This evaluation process facilitates data-driven decisions on model enhancement and fine-tuning for improved segmentation outcomes. Moreover, these metrics allow for fair comparisons with other models and serve as standard benchmarks for evaluating similar image segmentation tasks.

5.8.1 Performance_Metric_IOU

Now equipped with a denoised version of the predicted table mask from `Noise_Removal` module, we can proceed to evaluate its performance using the Intersection over Union (IoU) [47, 48, 35, 36] performance metric. To achieve this, we rely on the `Generating_Ground_Truth_Coordinates` module, which aids in extracting the coordinates of the predicted table mask after denoising.

With the ground truth coordinates and the predicted coordinates obtained, we feed them into the `calculate_iou()` function, which calculates the IoU score. The IoU score is determined by calculating the intersection area (the overlapping region between the ground truth and predicted masks) and the union area (the total region covered by both masks). The IoU score is then derived by dividing the intersection area by the union area Fig. 5.5. A higher IoU value close to 1 indicates a substantial overlap between the predicted table region and the ground truth region, implying that the TableNet model has accurately predicted the table's location.

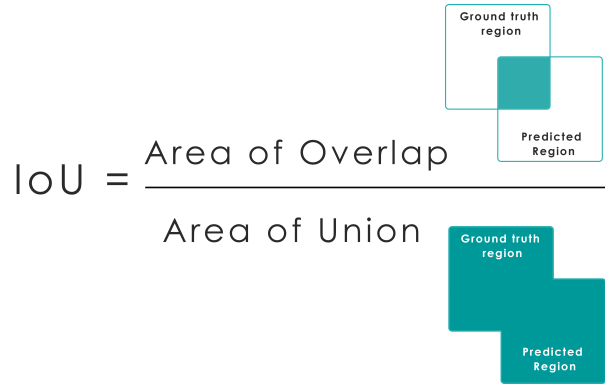


Figure 5.5: Intersection Over Union

Conversely, a lower IoU value close to 0 suggests a limited overlap, indicating that the TableNet model has not precisely predicted the table's location. Thus, by making use of this pipeline, we can refine the predicted table mask, assess its accuracy, and obtain valuable insights into the performance of the TableNet model.

5.8.2 Bitwise XOR

Throughout this module, we explore another performance criterion, Bitwise XOR, to extensively assess the TableNet model's effectiveness in table detection. In the initial phase, we obtain the prediction mask image output from the `Noise Removal` module. This represents the prediction table mask for the TableNet model. Similarly, get a ground truth mask image that accurately outlines the actual table mask. Then, perform a bitwise XOR operation between these two images to get an output image that shows the similarities and differences between the prediction mask and the ground truth mask. Here the critical step is performed using "`cv2.bitwise_xor()`" to apply a bitwise XOR operation between the binary image of the prediction mask and the ground truth mask. The resulting XOR image Fig. 5.6 contains pixels with a value of 1 where there are discrepancies between the two masks and pixels with a value of 0 where the masks are similar. By examining the resulting image, we can determine the accuracy of the model in predicting table locations.

For evaluating the performance of the TableNet model, the number of white pixels (value of 1) in the output resulting image from Bitwise XOR is counted, which represents the misclassified pixels. A high degree of overlap between the predicted and ground truth masks is

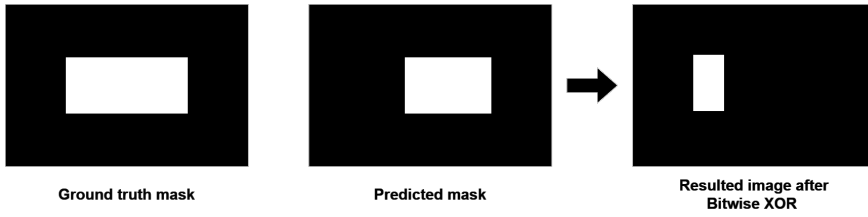


Figure 5.6: Bitwise XOR

indicated by the lower count of white pixels. As a result, the TableNet model has correctly predicted the table locations. On the contrary, a small degree of overlap between the predicted and ground truth masks is indicated by the higher count of white pixels. Hence this indicates that the prediction of TableNet models is less accurate, and there can be some differences between the predicted and grounded truth table locations. By utilizing this pipeline and making use of Bitwise XOR performance criteria, we obtain valuable insights into the actual capabilities of the model. This thorough evaluation approach enables users to identify the potential areas for improvements and thereby could lead to the development of a more accurate and reliable table detection model in the field of document analysis and information extraction.

$$\text{Bitwise XOR Error Rate} = \frac{\text{Number of White Pixels in Bitwise XOR Result}}{\text{Total Number of Pixels in the Image}} \times 100 \quad (5.1)$$

With the help of Bitwise XOR, we calculated the error rate, representing the percentage of misclassified pixels. The error rate is obtained by dividing the number of white pixels in the Bitwise XOR resulted image between predicted and ground truth masks by the total number of pixels in the image, indicating the degree of discrepancies between the predicted and ground truth table masks. To assess the TableNet model's performance, we calculate the error rate using Bitwise XOR on the test dataset. One key benefit of using Bitwise XOR as a metric is that it allows us to compare pixel-level differences between the ground truth and predicted masks. This provides a way to evaluate the TableNet model based on this score.

In this chapter, we implemented the architecture, where we preprocessed the Marmot dataset for table and column masks. We also performed data augmentation to generate additional training data for the table detection task using the `tables_generation` module. This module produces PDFs based on input parameters such as the number of rows, columns, data types, and table style. The PDFs are then processed using the `pdf_to_images` module to generate corresponding images. Manual data annotation requires significant time and effort, and we utilized the `generating_ground_truth_coordinates` module to create ground truth masks. Subsequently, the TableNet model was trained using the Marmot training dataset, and its performance was evaluated using two key performance metrics: Intersection over Union and Bitwise XOR error rate. The presence of substantial noise in the predicted masks prevented the accurate evaluation of the model using Intersection over Union. This underscored a

5 Implementation

significant advantage for Bitwise XOR, which can effectively evaluate the model by comparing pixel-level differences and similarities between the ground truth and predicted masks.

6 Evaluation

In this part of our thesis, we will compare two different table detection models; our specially trained TableNet model with an input resolution of 256 x 256 and another one with an input resolution of 1024 x 1024. The aim is to understand how these models perform by examining both qualitative and quantitative aspects.

6.1 Intersection Over Union (IOU)

To effectively assess how well the models are performing, we have chosen to use the Intersection over Union (IOU)[47, 48, 35, 36] as our performance metric. The IOU metric is commonly used in computer vision tasks, such as object detection and segmentation, to measure the overlap between predicted and actual regions. To evaluate our results, we are fortunate enough to have table coordinates obtained from our unique training data. These coordinates serve as a reference point and show us the accurate positions of the tables in the images. Additionally, we also have table coordinates generated by our TableNet model. With this information, we can now calculate the IOU between the predicted and actual table regions using a well-defined formula.

$$IOU = \frac{\text{Area of Intersection}}{\text{Area of Union}} \quad (6.1)$$

Where Area of Intersection represents the portion where the predicted table region overlaps with the ground truth table region. On the other hand, Area of Union represents the overall area covered by both the predicted and ground truth table regions.

The IOU value allows us to measure how well the predicted table regions match the ground truth regions. A high IOU score means that the predictions closely align with the table's locations, indicating accurate and reliable table detection. Additionally, we analyze by visually inspecting and comparing the outputs of both models. We look at sample images from both models to understand how effectively they can identify and outline tables in document layouts. This qualitative assessment helps us pinpoint any strengths, limitations, or shortcomings of each model in real-world situations. By conducting this evaluation, we gain valuable insights into the capabilities and weaknesses of our trained TableNet model with a 1024 x 1024 input resolution compared to the 256 x 256 input resolution. With this knowledge, we can make model selection and fine-tuning decisions. Furthermore, it provides guidance for optimizing the architecture and parameters of our model to achieve even better results in table detection.

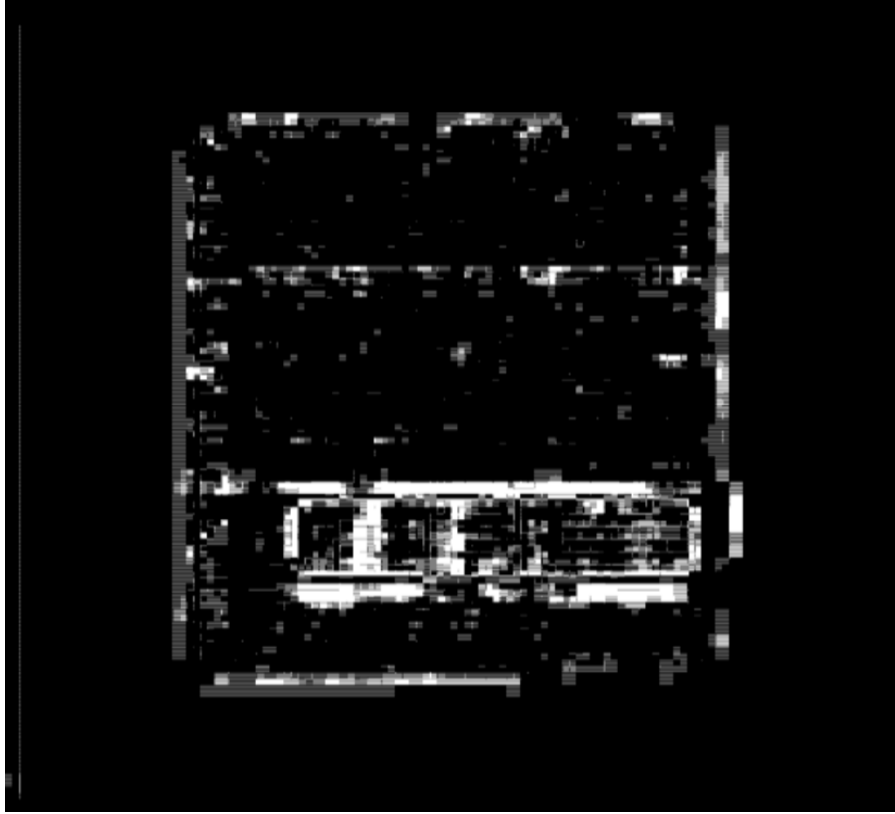


Figure 6.1: Result after Performing Bitwise XOR

6.2 Bitwise XOR

In addition to using Intersection over Union (IOU) to measure how well the table detection performs, we have also investigated the effectiveness of using Bitwise XOR operation as an evaluation method. The advantage of using Bitwise XOR as a performance metric is that it allows us to analyze the differences between the ground truth and predicted masks on a pixel level, which helps us evaluate the TableNet model effectively. This approach gives us insights into how accurately the model can locate tables and serves as a primary performance metric alongside IOU.

The process starts by converting the predicted table mask into a binary form and then applying a threshold to remove unwanted artifacts. We can use techniques like reducing noise through Gaussian blur [52] and thresholding to improve the accuracy of the predictions in the image. Next, we obtain the ground truth image mask that shows where the actual tables are located in the images. Using a Bitwise XOR operation between the predicted image mask and the ground truth binary image mask, we can generate a result Fig. 6.1 that highlights discrepancies and similarities between these two representations.

The important aspect is to count the number of white pixels in the XOR result. This allows us to evaluate how well the model performs. If the count is zero, it means that the predicted table regions align perfectly with the ground truth, indicating that the TableNet model accurately predicts the exact table locations. Conversely, if the number of white pixels equals the image's size, it indicates the model's failure to predict the correct table coordinates, necessitating further optimization and refinement. By employing the Bitwise XOR operation as an evaluation metric, we gain a direct and intuitive understanding of the model's performance in table detection. Analyzing the binary representations of predictions and ground truth allows for easy identification of areas of discrepancy and evaluation of noise reduction technique's effectiveness in improving the model's accuracy.

The Bitwise XOR operation, in conjunction with the widely used IOU metric, provides a comprehensive evaluation approach for table detection. The counting of white pixels in the XOR result offers a valuable criterion for identifying accurate predictions and areas requiring improvement. Such a holistic evaluation aids in comprehending the model's strengths and limitations, guiding us toward developing more accurate and robust table detection algorithms for practical applications.

6.3 Results

In this section, we embark on a comprehensive analysis of the predicted images generated by both our custom-trained TableNet model with 256×256 input resolution and with 1024×1024 input resolution. Our goal is to thoroughly evaluate the accuracy and effectiveness of these models in table detection.

In Fig. 6.3, we have observed a notable challenge with our custom-trained TableNet model with an input resolution of 256×256 , as it struggles to precisely detect the table's exact position within the image, even when dealing with data it has encountered during training. Regrettably, the model is unable to predict even a small section of the table region. Notably, the custom-trained model encounters difficulties in providing accurate predictions for table regions in both the dataset used for training and the subsequent test dataset. Surprisingly, this limitation persists even after the model has undergone an extensive training period of 5000 epochs. This unexpected limitation becomes evident when comparing the predicted table regions with the ground truth, revealing disparities and misalignments.

To quantitatively evaluate the model's performance, we consider two essential evaluation metrics: the Intersection over Union (IOU) and the Bitwise XOR operation. The IOU score is zero due to the absence of an intersection area between the predicted mask and the ground truth mask. As a result, the model fails to predict even a small section of a table region. The obtained IOU score of 0 indicates a significant lack of alignment between the predicted and ground truth table regions, suggesting that the model's ability to pinpoint the exact table coordinates is significantly compromised. Furthermore, the Bitwise XOR operation, which counts

6 Evaluation

fastest in the major urban centers - the Ho Chi Minh City and Hanoi regions - while underdevelopment, underemployment, and poverty will remain major problems in rural regions, where 80 percent of the population are found.

Table 1. GDP Growth Rates 1990-1996 (percent)

	1990	1991	1992	1993	1994	1995	1996 prel.
Total GDP	5.1	6.0	8.6	8.1	8.8	9.5	9.3
Agriculture, forestry, and fishing	1.5	4.2	5.1	4.5	1.3	5.1	5.0
Industry and construction	2.9	9.0	14.0	13.1	14.0	13.9	13.6
Services	10.8	8.3	7.0	9.2	10.2	10.9	9.5

Source: CIEM (1997).

Next to the high growth rates and the rapid industrialization, the most impressive economic achievements during the first half of the 1990s have been related to macroeconomic stabilization. Most importantly, inflation has been reduced drastically, as shown in Table 2 below. Restrictions on domestic credit expansion, particularly for the government sector, were the main policy measures to contain inflation in the early 1990s. This brought down the inflation rate from 67 percent in 1990 and 1991 to 5 percent in 1993. The higher inflation rates in 1994 and 1995 resulted from bottlenecks in the economy - rice production was hurt by floods in the South and drought in the North, and there were disruptions in the supply of cement to the construction industry - as well as a credit expansion directed to both SOEs and the private sector. In response to the accelerating inflation, the government again tightened its financial and fiscal policies in mid-1995. Increases in interest rates, stricter enforcement of credit ceilings, and limitations on public expenditure growth managed to curb inflation already during second half of 1995, and the inflation rate in 1996 was kept below 5 percent. Two important determinants of the low inflation level in 1996 were a bumper rice crop both in the North and the South, and that the government managed to keep prices under control during the Tet period, when there is normally a surge in demand for goods and services.

4

Figure 6.2: Marmot Dataset input image for 256x256 TableNet model

Figure 6.3: Predicted image of Marmot dataset from 256x256 TableNet model

the number of white pixels in the XOR result, provides further evidence of the model's struggles. The outcome of the Bitwise XOR is equal to the number of white pixels in ground truth mask because the output mask predicted by the 256 x 256 model does not contain any white pixels, resulting in an empty image with black pixels. The recorded count of 5824 white pixels highlights notable differences between the predicted and ground truth table regions, further emphasizing the model's inability to accurately predict the correct table positions, even for data encountered during training.

However, the input image processed by the model trained for 500 epochs produces better predictions compared to the model trained for 5000 epochs. This discrepancy could be attributed to overfitting the data. For certain input images, the model trained for 5000 epochs struggles to make accurate predictions. In Fig. 6.5, we can observe that the model trained for 809 epochs, utilizing an input resolution of 256 x 256, is capable of predicting a portion of the table region.

To quantitatively evaluate the model with 500 epochs, we consider the bitwise XOR error rate. By calculating the pixel-wise differences between the ground truth and predicted masks, we find 4892 differing pixels. The error rate is calculated by taking the ratio of the number of white pixels in the bitwise XOR result to the total number of pixels in the image, resulting in 7.46%. This indicates that 7.46% of the pixels are misclassified.

We now subject the novel test data, generated using LaTeX, as input to the 256 x 256 model. In Fig. 6.7, we have observed a notable challenge with our custom-trained TableNet model with an input resolution of 256 x 256, as it struggles to precisely detect the table's exact position within the image. Notably, this custom-trained model exhibits difficulties in clearly predicting

fastest in the major urban centers - the Ho Chi Minh City and Hanoi regions - while underdevelopment, underemployment, and poverty will remain major problems in rural regions, where 80 percent of the population are found.

Table 1. GDP Growth Rates 1990-1996 (percent)

	1990	1991	1992	1993	1994	1995	1996
Total GDP	5.1	6.0	8.6	8.1	8.8	9.5	9.3
Agriculture, forestry, and fishing	1.5	4.2	5.1	4.5	1.3	5.1	5.0
Industry and construction	2.9	9.0	14.0	13.1	14.0	13.9	15.6
Services	10.8	8.3	7.0	9.2	10.2	10.9	9.5

Source: CHEM (1997).

Next to the high growth rates and the rapid industrialization, the most impressive economic achievements during the first half of the 1990s have been related to macroeconomic stabilization. Most importantly, inflation has been reduced drastically, as shown in Table 2 below. Restrictions on domestic credit expansion, particularly for the government sector, were the main policy measures to contain inflation in the early 1990s. This brought down the inflation rate from 67 percent in 1990 and 1991 to 5 percent in 1993. The higher inflation rates in 1994 and 1995 resulted from bottlenecks in the economy - rice production was hurt by floods in the South and drought in the North, and there were disruptions in the supply of cement to the construction industry - as well as a credit expansion directed to both SOEs and the private sector. In response to the accelerating inflation, the government again tightened its financial and fiscal policies in mid-1995. Increases in interest rates, stricter enforcement of credit ceilings, and limitations on public expenditure growth managed to curb inflation already during second half of 1995, and the inflation rate in 1996 was kept below 5 percent. Two important determinants of the low inflation level in 1996 were a bumper rice crop both in the North and the South, and that the government managed to keep prices under control during the Tet period, when there is normally a surge in demand for goods and services.

4

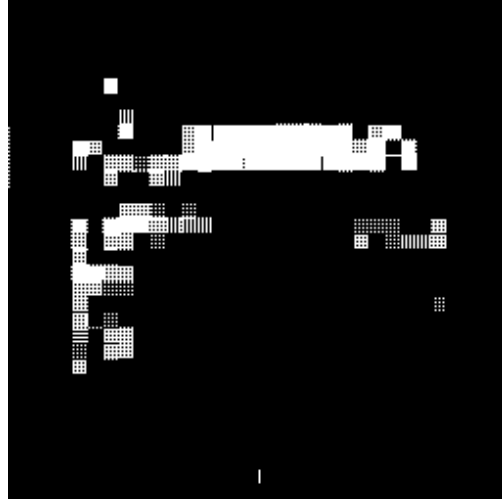


Figure 6.4: Marmot Dataset input image for 256x256 TableNet model with 500 epochs

Figure 6.5: Predicted image of Marmot dataset from 256x256 TableNet model with 500 epochs

the table regions for both trained and test datasets. Surprisingly, this limitation remains even after the model has undergone an extensive training period of 5000 epochs. This unexpected limitation becomes evident when comparing the predicted table regions with the ground truth, revealing disparities and misalignments.

To quantitatively evaluate the model's performance, we consider two essential evaluation metrics: the Intersection over Union (IOU) and the Bitwise XOR operation. The obtained IOU score of 0.164 indicates a significant lack of alignment between the predicted and ground truth table regions, suggesting that the model's ability to pinpoint the exact table coordinates is significantly compromised. This score defines the limited degree of overlap between the two masks, specifically the ground truth table mask and the corresponding mask predicted by the 256 x 256 model.

Furthermore, a comprehensive assessment of the model's performance is conducted utilizing the Bitwise XOR metric, involving the execution of the bitwise XOR operation between the resized binary representation of the ground truth table mask and the binary image of the predicted table mask. Upon examining the results, a striking observation emerges from the Fig. 6.8 post performing the Bitwise XOR operation, and the entire ground truth table mask is faithfully replicated. Furthermore, the Bitwise XOR operation, which counts the number of white pixels in the XOR result, provides further evidence of the model's struggles. The recorded count of 2530 white pixels highlight notable differences between the predicted and ground truth table regions, further emphasizing the model's inability to accurately predict the correct table positions.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut porta elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam ac mi libero, namquam eget, consectetur at, iugulata a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Maecenas ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus ac amet tortor gravida placerat. Integer sagittis est, laoreet eu, porttitor quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, euismod eu, pulvinar sit, nulla ac, nulla. Cras litor auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit nulla. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cras ante tuncque penesillat et magna dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum tristique. Pellentesque cursus lectus mauris.

Nulla malesuada pellentesque diam. Donec felis erat, congue sem, volutpat ut, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nunc nunc. Pellentesque nunc. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas laoreet. Nam ligula ligula, eleifend at, accumsan nec, sagittis a, ipsum. Morbi blandit ligula fengiat magna. Nunc eleifend consequat lectus. Sed laoreet nulla vitae enim. Pellentesque tincidunt purus vel congue. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam volutpat metus eu enim. Vestibulum pellentesque.

Quisque ut. Integer placerat.

Header1	Header2	Header3	Header4
25	12.02	4.16	color@example.com
12	55.46	10.96	gray@example.com
36	55.31	9.16	blue@example.com
21	70.12	1.94	blue@example.com
48	31.5	1.16	red@example.com

Table 1: Example Table

sed et amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Vitae facilisis. Nam elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sed dictum. Sed odio placerat quam, ac pulvinar dui purus eget enim. Nam vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae tunc porta vehicula.

Figure 6.6: (a) Novel test Dataset input image for 256x256 TableNet model

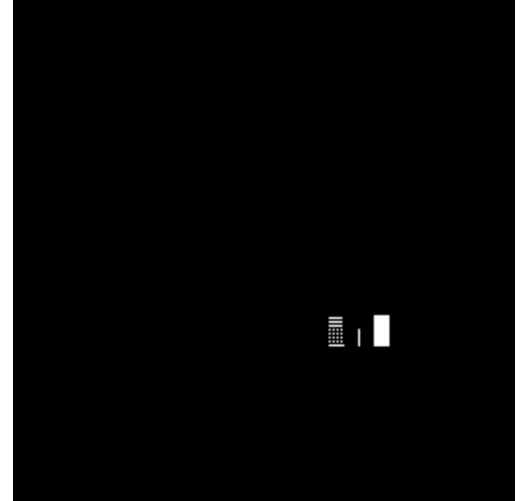


Figure 6.7: (b) Predicted image of Novel test dataset from 256x256TableNet model

The error rate can be calculated by taking the ratio of the number of white pixels in the bitwise XOR result to the total number of pixels in a predicted mask. To quantify the error rate of the 256 x 256 TableNet model, a calculated approach employs the Bitwise XOR metric, resulting in $\left(\frac{2530}{65536}\right) \times 100 = 3.86\%$. This outcome highlights the model’s relatively modest error rate, emphasizing the need for further enhancements to achieve more precise and dependable table detection outcomes.

The intriguing aspect is that the custom-trained model with an input resolution of 256 x 256 exhibits poor performance even on data it has previously encountered during training. This unexpected behavior prompts us to delve deeper into the model’s architecture and training process. Despite implementing the methodology precisely but with different input resolutions, as detailed in the TableNet research paper [9], the model’s suboptimal performance raises questions about potential factors that might have influenced its predictive capabilities.

Upon meticulously scrutinizing the findings arising from the application of the 256 x 256 TableNet model, which, unfortunately, did not yield the anticipated enhancements, a concerted endeavor was initiated to discern the underlying factors contributing to these less optimal outcomes. However, despite diligent investigation, no discernible factors that could account for these suboptimal results were ascertained. Notably, the model underwent training with varying epochs, and intriguingly, it exhibited an inability to accurately predict results even for data that had been previously encountered during its training phase. In light of this puzzling outcome, a pivotal decision was made to adhere closely to the implementation strategies detailed within the TableNet research paper. The model was reconfigured to accommodate an expanded input resolution of 1024 x 1024, thereby scaling up the scope and complexity of the analysis.

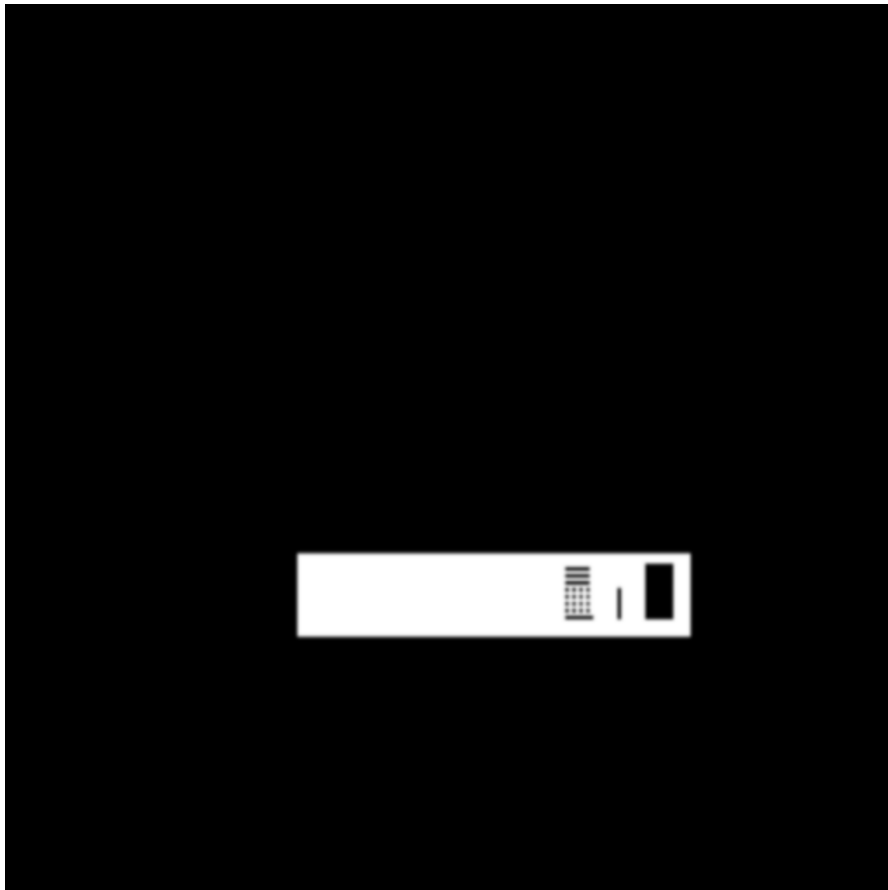


Figure 6.8: Result after Performing Bitwise XOR for novel test dataset

6 Evaluation

This revision enabled a comprehensive reevaluation of the model’s capabilities and limitations. Subsequently results were systematically gauged and documented to provide a comprehensive comparative assessment of the model’s performance across different input resolutions.

After conducting a meticulous examination of the anticipated outcomes presented in the predicted image: Fig. 6.10, discernible enhancements come to the forefront when compared to the performance of the 256 x 256 TableNet model. In this context, the TableNet model, endowed with an input resolution of 1024 x 1024, showcases improved capabilities. It exhibits a certain level of proficiency in localizing tables; however, precision remains an area warranting refinement, as evidenced by noticeable imperfections in its ability to accurately determine table locations. An evident concern arises from the prevalence of noise encircling the predicted table mask along its borders and within areas of white space.

fastest in the major urban centers - the Ho Chi Minh City and Hanoi regions - while underdevelopment, underemployment, and poverty will remain major problems in rural regions, where 80 percent of the population are found.

Table 1. GDP Growth Rates 1990-1996 (percent)

	1990	1991	1992	1993	1994	1995	1996 pred.
Total GDP	5.1	6.0	8.6	8.1	8.8	9.5	9.3
Agriculture, forestry, and fishing	1.5	4.2	5.1	4.5	1.3	5.1	5.0
Industry and construction	2.9	9.0	14.0	13.1	14.0	13.9	15.6
Services	10.8	8.3	7.0	9.2	10.2	10.9	9.5

Source: CHEM (1997).

Next to the high growth rates and the rapid industrialization, the most impressive economic achievements during the first half of the 1990s have been related to macroeconomic stabilization. Most importantly, inflation has been reduced drastically, as shown in Table 2 below. Restrictions on domestic credit expansion, particularly for the government sector, were the main policy measures to contain inflation in the early 1990s. This brought down the inflation rate from 67 percent in 1990 and 1991 to 5 percent in 1993. The higher inflation rates in 1994 and 1995 resulted from bottlenecks in the economy - rice production was hurt by floods in the South and drought in the North, and there were disruptions in the supply of cement to the construction industry - as well as a credit expansion directed to both SOEs and the private sector. In response to the accelerating inflation, the government again tightened its financial and fiscal policies in mid-1995. Increases in interest rates, stricter enforcement of credit ceilings, and limitations on public expenditure growth managed to curb inflation already during second half of 1995, and the inflation rate in 1996 was kept below 5 percent. Two important determinants of the low inflation level in 1996 were a bumper rice crop both in the North and the South, and that the government managed to keep prices under control during the Tet period, when there is normally a surge in demand for goods and services.

4



Figure 6.9: Marmot Dataset input image for 1024x1024 TableNet model

Figure 6.10: Predicted image of Marmot dataset from 1024x1024 TableNet model

When examining the predicted table mask obtained from the TableNet model at 1024 x 1024 resolution, a lot of noise is observed, which significantly complicates the evaluation process. Model estimation becomes particularly challenging given this noisy representation. Efforts to generate the coordinates of the predicted table mask in the image reveal a complex scenario due to artifacts. This complexity comes from the method used to define the coordinates of the table’s bounding box. Specifically, we consider the left white pixel and the right white pixel as the defining points of the bounding box. However, the presence of artifacts hinders the generation of accurate coordinates for the predicted table mask. To address this issue, we use a strategy to eliminate image noise. This involves applying Gaussian Blur and Thresholding techniques. First, we convert the predicted table mask into a binary image. Then we apply Gaussian blur using default parameters and a filter size of 1.5 x 1.5. The next step is to thresh-

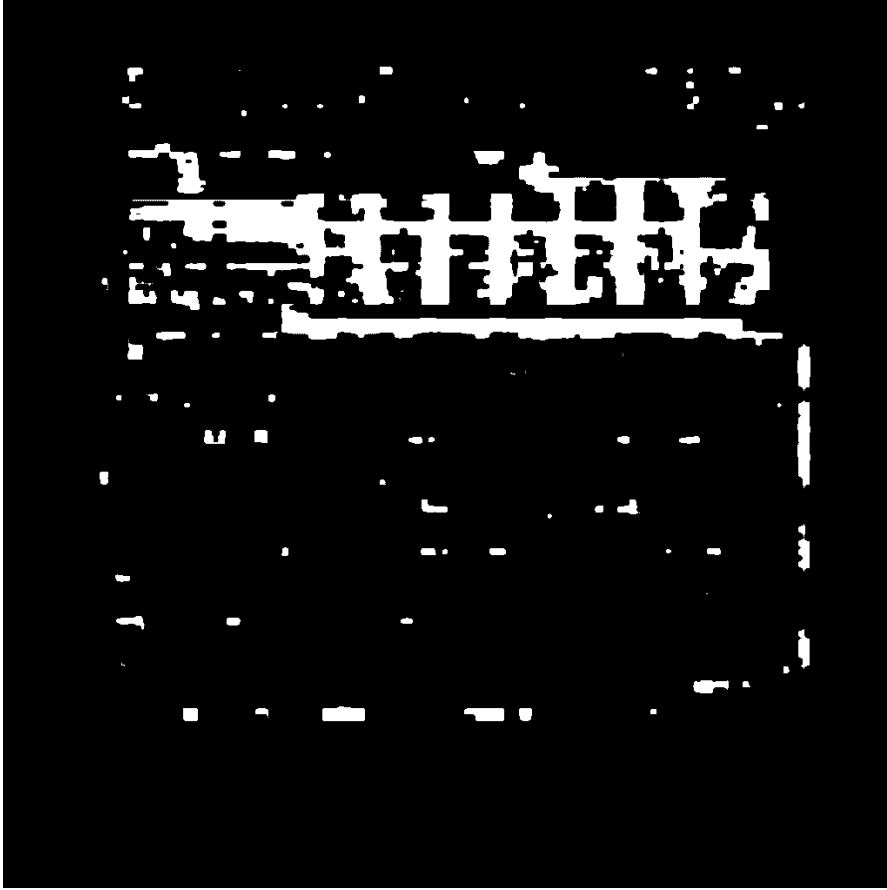


Figure 6.11: Noise removal from predicted table mask

old the output of the Gaussian blur operation.

In Fig. 6.11, it is clear that we apply Gaussian blur with a filter size of 1.5×1.5 and a threshold of 168 on the predicted table mask generated by the TableNet model with dimensions of 1024×1024 pixels. This multi-stage technique smooths out the image, reduces fluctuations at the level, and provides a cleaner representation with less noise. This improvement is crucial for determining coordinates for predicted table masks in order to achieve more precise estimation and analysis.

To quantitatively assess how well our model performs, we consider two evaluation metrics; Intersection, over Union (IOU), and Bitwise XOR operation. The IOU value of 0.65 for table detection indicates a significant similarity between the predicted and actual table masks. This value reflects a significant improvement in alignment between the two, suggesting that the model is now able to more accurately identify and locate tables in images. The increased overlap between the predicted and ground truth table regions, as compared to the 256×256 TableNet model, implies that the model is successfully delineating the borders and dimensions

of tables, contributing to a more precise and reliable table detection process.

Given the large amount of noise in the images, the Intersection over Union (IOU) metric is not considered a good performance measure. The IOU metric cannot accurately encapsulate the table area as the white pixels are enclosed across the borders, resulting in discrepancies that prevent obtaining accurate table coordinates. As a result of this limitation, evaluation by the Bitwise XOR metric becomes a viable alternative. This metric works by comparing pixel-wise values between the ground truth and the prediction masks to provide a more detailed assessment in light of the above challenges. Furthermore, a comprehensive assessment of the model's performance is conducted utilizing the Bitwise XOR metric, involving the execution of the bitwise XOR operation between the resized binary representation of the ground truth table mask and the binary image of the predicted table mask. Upon examining the results, a striking observation emerges from Fig. 6.12 post performing the Bitwise XOR operation, the entire ground truth table mask is faithfully replicated. Furthermore, the Bitwise XOR operation, which counts the number of white pixels in the XOR result, provides further evidence of the model's struggles. The recorded count of 95215 white pixels highlights notable differences between the predicted and ground truth table regions, further emphasizing the model's inability to accurately predict the correct table positions.

The error rate can be calculated by taking the ratio of the number of white pixels in the bitwise XOR result to the total number of pixels in a predicted mask. To quantify the error rate of the 1024 x 1024 TableNet model, a calculated approach employs the Bitwise XOR metric, resulting in $(\frac{95215}{1048576}) \times 100 = 9.08\%$. This outcome highlights the model's relatively modest error rate, emphasizing the need for further enhancements to achieve more precise and dependable table detection outcomes.

We now subject the novel test data, generated using LaTeX, as input to the 1024 x 1024 model. In Fig. 6.14, it is observable that the model exhibits an improved ability to detect the table region when compared with the performance of the 256 x 256 TableNet model. However, the model continues to encounter challenges in precisely determining the exact position of the table within the image. This unforeseen limitation becomes apparent when contrasting the predicted table regions with the ground truth, thereby exposing disparities and misalignments.

Similar to the prediction for the marmot dataset, this 1024 x 1024 model generates a table mask prediction characterized by a significant amount of noise along the borders and within white space regions. In a manner akin to the process described above, noise reduction techniques are implemented to address the artifacts within the image, enhancing the precision of model evaluation. Gaussian Blur and Thresholding techniques are employed for this purpose. First, the predicted table mask is converted to its binary image, and then a Gaussian blur is applied with default parameters and a filter size of 1.5 x 1.5. The next step is to threshold the output of the Gaussian blur operation.

In Fig. 6.15, it is evident that Gaussian blur is applied with a filter size of 1.5 x 1.5 and a threshold of 192 on the predicted table mask of 1024 x 1024 TableNet model. This multi-stage

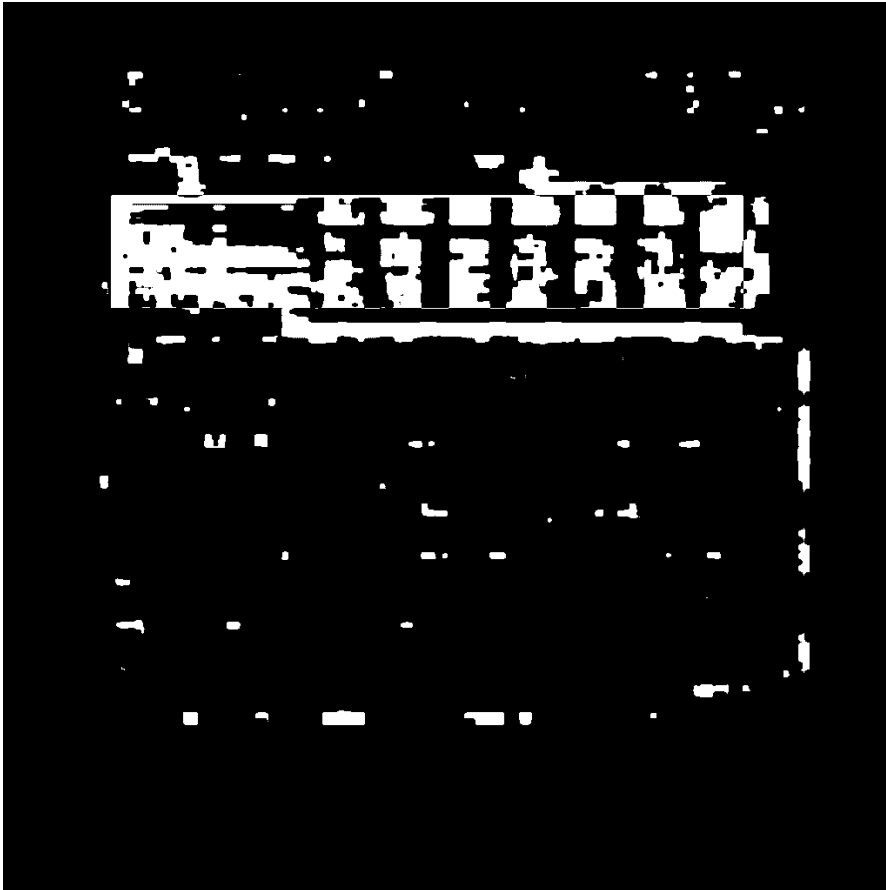


Figure 6.12: Result after Performing Bitwise XOR for Marmot dataset

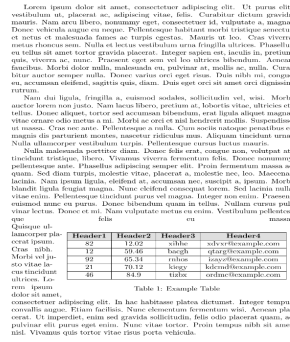


Figure 6.13: Novel test Dataset input image for 1024 x 1024 TableNet model

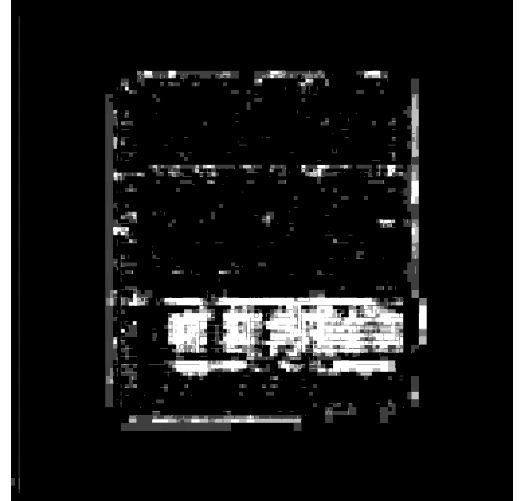


Figure 6.14: Predicted image of Novel test dataset from 1024 x 1024 TableNet model

technique smoothes the image, reduces pixel-level fluctuations, and provides a cleaner representation with less noise. This improvement is necessary to accurately generate the coordinates of the expected table mask for more accurate estimation and analysis.

To quantitatively evaluate the model's performance, we consider two essential evaluation metrics: the Intersection over Union (IOU) and the Bitwise XOR operation. The Intersection over Union (IOU) value of 0.705 associated with table detection indicates significant convergence between the predicted table mask and the ground truth table mask. This value reflects a significant improvement in alignment between the two, suggesting that the model is now able to more accurately identify and locate tables in images. The increased overlap between the predicted and ground truth table regions, as compared to the 256 x 256 TableNet model, implies that the model is successfully delineating the borders and dimensions of tables, contributing to a more precise and reliable table detection process.

Let's evaluate the predicted mask using another performance metric, the Bitwise XOR, which involves executing the bitwise XOR operation between the resized binary representation of the ground truth table mask and the binary image of the predicted table mask. Upon analyzing the outcomes, a significant observation emerges from Fig. 6.16 post the Bitwise XOR operation, the entire ground truth table mask is replicated. Furthermore, the Bitwise XOR operation, which counts the number of white pixels in the XOR result, provides further evidence of the model's struggles. The recorded count of 57146 white pixels highlights notable disparities between the predicted and ground truth table regions, thereby emphasizing the model's inability to precisely predict the correct table positions.

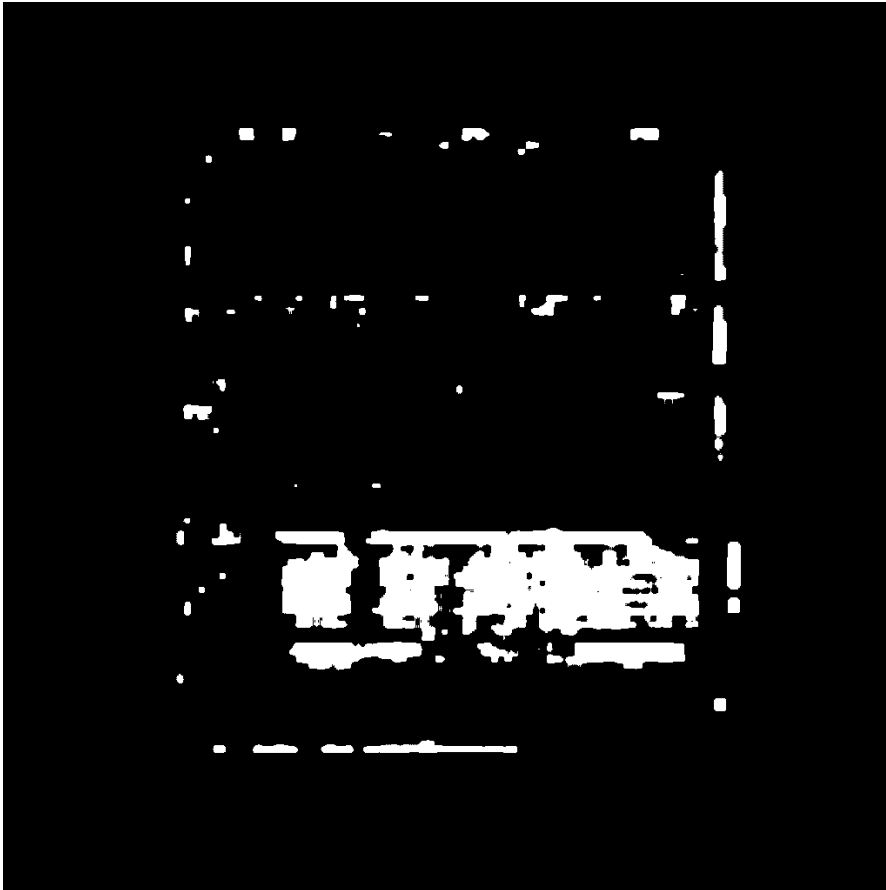


Figure 6.15: Noise removal from predicted table mask

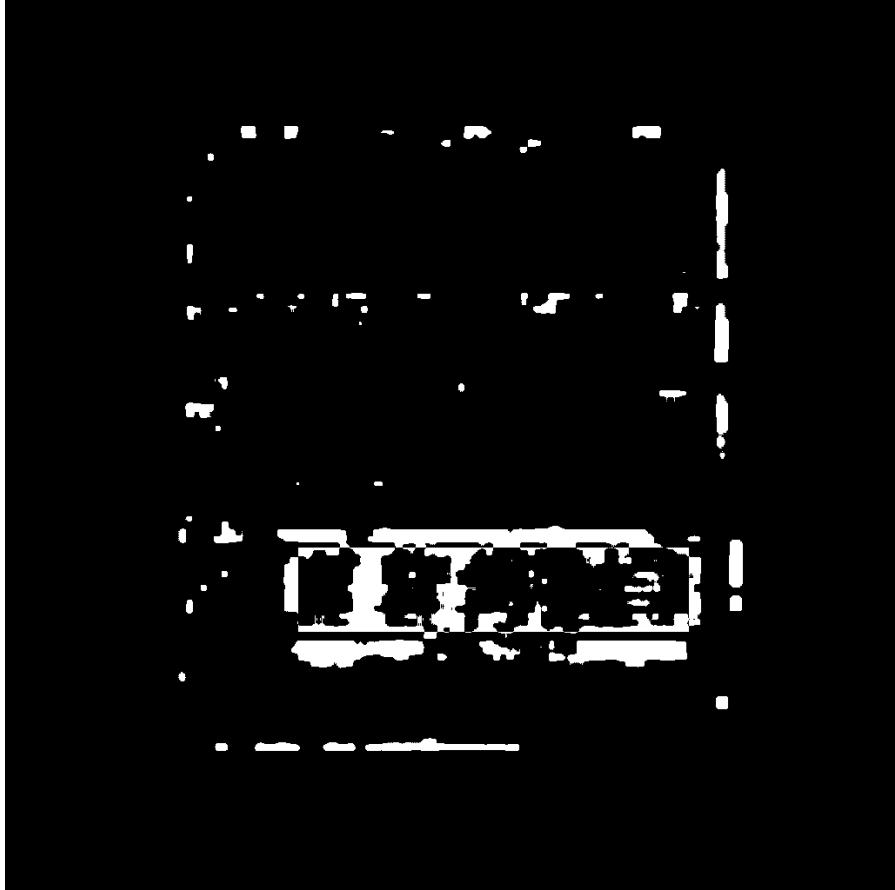


Figure 6.16: Result after Performing Bitwise XOR for novel test dataset

The error rate can be calculated by taking the ratio of the number of white pixels in the bitwise XOR result to the total number of pixels in a predicted mask. To quantify the error rate of the 1024 x 1024 TableNet model, a calculated approach employs the Bitwise XOR metric, resulting in $(\frac{57146}{1048576}) \times 100 = 5.44\%$. This outcome highlights the model’s relatively modest error rate, emphasizing the need for further enhancements to achieve more precise and dependable table detection outcomes.

Model	Bitwise XOR Error Rate	
	Marmot Test Dataset	Novel Data
256 x 256 model with 241 epochs	12.41%	10.50%
256 x 256 model with 504 epochs	9.18%	5.41%
256 x 256 model with 1540 epochs	11.69%	4.04%

Table 6.1: Bitwise XOR Error Rate Comparison across different epochs of 1024 x 1024 model

We have considered two different datasets for evaluating the existing TableNet architecture: the test dataset from Marmot and novel data from the `tables_generation` module. The Marmot test dataset comprises 50 images, while the novel data comprises 36 images. The model is evaluated using these images from both datasets. The Bitwise error rate is calculated for each image in these two datasets, and the mean value of the bitwise error rates is used as the error rate for the specific model. This process is repeated for different models trained with varying epochs and various input sizes of the model. From Table 6.1, we can conclude that the model with 256 x 256 input resolution performs better up to 1500 epochs. After 1500 epochs, the model becomes overfitted to the dataset, resulting in worse results. As the model is trained for 241, 504, and 1540 epochs, we observe a gradual decrease in the bitwise error rate. Additionally, when considering novel data, it becomes evident that the error rate consistently decreases across these new images, which were generated using LaTeX.

Model	Bitwise XOR Error Rate	
	Marmot Test Dataset	Novel Data
1024 x 1024 model with 151 epochs	13.83%	5.75%
1024 x 1024 model with 267 epochs	14.33%	4.59%
1024 x 1024 model with 764 epochs	13.91%	4.33%

Table 6.2: Bitwise XOR Error Rate Comparison across different epochs of 1024 x 1024 model

From Table 6.2, we can conclude that the model with a 1024 x 1024 input resolution performs better compared to the 256 x 256 model. The 256 x 256 model failed to predict some images from the Marmot dataset with 5000 epochs, resulting in worse results. We can also observe a gradual decrease in the bitwise error rate as the number of epochs increases for the novel

test data, which was generated using LaTeX. The 1024 x 1024 model consistently performs well for all types of images, both from the Marmot and novel datasets, producing better results in comparison with the 256 x 256 model. However, this 256 x 256 model tends to overfit the data with more epochs.

	without noise removal	with noise removal
Bitwise XOR Error Rate	13.84%	12.62%

Table 6.3: Bitwise XOR error rates with and without noise removal

We have predicted the table masks for all the images in the Marmot test dataset. However, as all the predicted masks contain noise along the borders and in white space regions, we want to evaluate the model both with and without this noise. To achieve this, we utilized noise reduction techniques such as Gaussian Blur and thresholding. All the predicted masks from the Marmot test dataset undergo Gaussian Blur processing and thresholding to eliminate the noise. Subsequently, the mean Bitwise XOR error rate is calculated using a model with an input size of 1024 x 1024. Analyzing the results in the Table 6.3, we observe a slight reduction in the mean error rate. This reduction indicates that the model is capable of better prediction for table regions compared to the earlier predictions that included noise.

From these results, we observed that the TableNet model with an input size of 256 x 256 is able to predict the table regions, although not very accurately. The model overfits the data after being trained on the Marmot dataset for 5000 epochs. The model cannot detect any table regions in some images from the Marmot dataset. However, the model with fewer epochs can detect tabular regions, though not very accurately, and the predicted masks contain a lot of noise across the borders and white space regions. In comparison to the 256 x 256 model, the 1024 x 1024 model is able to predict tabular regions more effectively. Even though this 1024 x 1024 model is trained for 800 epochs, it produces better results on the Marmot test dataset and even with autogenerated novel data. After applying noise reduction techniques, the model is able to produce better results compared to earlier predictions with noise present in the predicted masks. We have observed that the Bitwise XOR error rate plays a major role in evaluating the models in comparison to the Intersection over Union because it quantifies the pixel-wise differences between the ground truth mask and the predicted mask, resulting in the count of misclassified pixels.

7 Conclusion and Future Works

In conclusion, this thesis has introduced a novel approach to address the scarcity of annotated training data for table detection in computer vision. The method involves generating an extensive training dataset automatically, simulating scientific articles and journals with diverse table styles and data types. This dataset allows for the evaluation of state-of-the-art architectures for table detection, with promising potential for enhancing accuracy.

A significant contribution of this research lies in the creation of a diverse dataset encompassing various table styles and structures. This facilitated the evaluation of TableNet, an innovative deep-learning model that simultaneously detects tables and recognizes their structures. TableNet showcased promising results in detecting tabular regions, even amidst noise around image borders from white space regions.

Looking ahead, future research could extend this approach to generate specialized training data for column detection. This could be achieved by generating ground truth coordinates for columns present in the table, involving the number of columns per table to evaluate the column mask. Additionally, extracting data from the detected table regions using PyTesseract OCR could further enhance the methodology. In the future, we also aim to enhance TableNet's capabilities by introducing a third branch dedicated to identifying rows. Nevertheless, this endeavor necessitates a manual annotation process since the existing datasets lack provisions for row-based annotations. Additionally, further advancements could include generating images with varied backgrounds, incorporating header and footer sections, and featuring pages containing both tables and images. The dataset's expansion to include tables with different font sizes, merged cells, nested tables, multi-level headers, and other complex styles will enhance the model's performance and adaptability.

This work sets the stage for more effective and integrated solutions in table detection and structure recognition. As the field progresses, the availability of high-quality training data will play a pivotal role in driving advancements and pushing the boundaries of table detection algorithms. The proposed approach contributes significantly to the field and can serve as a solid foundation for future developments, benefiting various applications that rely on precise and efficient table detection.

Bibliography

- [1] B. Couasnon and A. Lemaitre, “Recognition of tables and forms,” in *Handbook of Document Image Processing and Recognition*, D. Doermann and K. Tombre, Eds. London: Springer London, 2014, pp. 647–677.
- [2] R. Smith, “An overview of the tesseract ocr engine,” vol. 2, 10 2007, pp. 629 – 633.
- [3] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” *ArXiv*, vol. abs/1511.08458, 2015. [Online]. Available: <https://api.semanticscholar.org/CorpusID:9398408>
- [4] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [5] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, “Object detection with deep learning: A review,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, pp. 1–21, 01 2019.
- [6] J. Yu, Y. Jiang, Z. Wang, Z. Cao, and T. Huang, “Unitbox: An advanced object detection network,” in *Proceedings of the 2016 ACM on Multimedia Conference*. ACM, 2016, pp. 516–520.
- [7] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew, “A review of semantic segmentation using deep neural networks,” *International Journal of Multimedia Information Retrieval*, vol. 7, pp. 87–93, 2018. [Online]. Available: <https://api.semanticscholar.org/CorpusID:21656238>
- [8] Y. Xing, L. Zhong, and X. Zhong, “An encoder-decoder network based fcn architecture for semantic segmentation,” *Wireless Communications and Mobile Computing*, vol. 2020, pp. 1–9, 07 2020.
- [9] S. Paliwal, V. D. R. Rahul, M. Sharma, and L. Vig, “Tablenet: Deep learning model for end-to-end table detection and tabular data extraction from scanned document images,” 09 2019.
- [10] B. Couasnon and A. Lemaitre, “Recognition of tables and forms,” in *Handbook of Document Image Processing and Recognition*, D. Doermann and K. Tombre, Eds. London: Springer London, 2014, pp. 647–677.
- [11] R. Zanibbi, D. Blostein, and J. R. Cordy, “A survey of table recognition: Models, observations, transformations, and inferences,” *International Journal on Document Analysis and Recognition (IJ DAR)*, vol. 7, no. 1, pp. 1–16, 2004.

Bibliography

- [12] A. C. e. Silva, A. M. Jorge, and L. Torgo, "Design of an end-to-end method to extract information from tables," *International Journal of Document Analysis and Recognition (IJ DAR)*, vol. 8, no. 2-3, pp. 144–171, 2006.
- [13] S. Khusro, A. Latif, and I. Ullah, "On methods and tools of table detection, extraction and annotation in pdf documents," *Journal of Information Science*, vol. 41, no. 1, pp. 41–57, 2015.
- [14] D. W. Embley, M. Hurst, D. Lopresti, and G. Nagy, "Table-processing paradigms: A research survey," *International Journal of Document Analysis and Recognition (IJ DAR)*, vol. 8, no. 2-3, pp. 66–86, 2006.
- [15] P. Pyreddy and W. B. Croft, "Tintin: A system for retrieval in text tables," in *Proceedings of the second ACM international conference on Digital libraries*, 1997, pp. 193–200.
- [16] F. Cesarini, S. Marinai, L. Sarti, and G. Soda, "Trainable table location in document images," in *2002 International Conference on Pattern Recognition*, vol. 3, 2002, pp. 236–240.
- [17] T. Kasar, P. Barlas, S. Adam, C. Chatelain, and T. Paquet, "Learning to detect tables in scanned document images using line information," in *ICDAR*, 2013, pp. 1185–1189.
- [18] A. C. e. Silva, "Learning rich hidden markov models in document analysis: Table location," in *2009 10th International Conference on Document Analysis and Recognition*, 2009, pp. 843–847.
- [19] D. B. Powell, E. A. Lee, W. C. Newman, S. S. Bhattacharyya, S. Ha, D. G. Messerschmitt, D. Genin, J. D. Moortel, D. Desmet, E. F. van de Velde, L. J. Hendren, G. R. Gao, E. R. Altman, C. Mukherjee, and W.-H. Ho, "Parts that add up to a whole : a framework for the analysis of tables," 2007. [Online]. Available: <https://api.semanticscholar.org/CorpusID:16892037>
- [20] J. Fang, P. Mitra, Z. Tang, and C. L. Giles, "Table header detection and classification," in *AAAI*, 2012, pp. 599–605.
- [21] M. Raskovic, N. Bozidarevic, and M. Sesum, "Borderless table detection engine," Patent, 2018, uS Patent No. 9,990,347.
- [22] M. Fan and D. S. Kim, "Detecting table region in pdf documents using distant supervision," Tech. Rep., 2015. [Online]. Available: <http://arxiv.org/abs/1506.08891>
- [23] T. Kieninger and A. Dengel, "The t-recs table recognition and analysis system," in *Document Analysis Systems*, ser. Lecture Notes in Computer Science, S.-W. Lee and Y. Nakano, Eds. Berlin: Springer, 1999, pp. 255–270.
- [24] Y. Wang, I. T. Phillips, and R. M. Haralick, "Table structure understanding and its performance evaluation," *Pattern Recognition*, vol. 37, no. 7, pp. 1479–1497, 2004.

- [25] A. Shigarov, A. Mikhailov, and A. Altaev, "Configurable table structure recognition in untagged pdf documents," in *Proceedings of the 2016 ACM Symposium on Document Engineering, DocEng 2016*, R. Sablatnig and T. Hassan, Eds. ACM, 2016, pp. 119–122.
- [26] L. Hao, L. Gao, X. Yi, and Z. Tang, "A table detection method for pdf documents based on convolutional neural networks," in *12th IAPR Workshop on Document Analysis Systems (DAS) 2016*. IEEE Computer Society, 2016, pp. 287–292.
- [27] D. N. Tran, T. A. Tran, A.-R. Oh, S.-H. Kim, and I.-S. Na, "Table detection from document image using vertical arrangement of text blocks," *International Journal of Contents*, vol. 11, no. 4, pp. 77–85, 2015.
- [28] M. C. Gobel, T. Hassan, E. Oro, and G. Orsi, "Icdar 2013 table competition," in *IEEE Computer Society*, 2013, pp. 1449–1453.
- [29] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "Deepdesrt: Deep learning for detection and structure recognition of tables in document images," in *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, vol. 1. IEEE, 2017, pp. 1162–1167.
- [30] I. Kavasidis, S. Palazzo, C. Spampinato, C. Pino, D. Giordano, D. Giuffrida, and P. Messina, "A saliency-based convolutional neural network for table and chart detection in digitized documents," *arXiv preprint arXiv:1804.06236*, 2018.
- [31] A. Tengli, Y. Yang, and N. L. Ma, "Learning table extraction from examples," in *Proceedings of the 20th International Conference on Computational Linguistics*. Association for Computational Linguistics, 2004, p. 987.
- [32] P. Singh, S. Varadarajan, A. N. Singh, and M. M. Srivastava, "Multidomain document layout understanding using few shot object detection," *arXiv preprint arXiv:1808.07330*, 2018.
- [33] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 11 2013.
- [34] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [35] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2015. [Online]. Available: <https://api.semanticscholar.org/CorpusID:10328909>
- [36] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2015. [Online]. Available: <https://api.semanticscholar.org/CorpusID:206594738>

Bibliography

- [37] H. Hashib, M. Leon, and A. M. Salaque, "Object detection based security system using machine learning algorithm and raspberry pi," in *2019 International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering (IC4ME2)*, 2019, pp. 1–4.
- [38] Lewis, "Object detection for autonomous vehicles," 2016. [Online]. Available: <https://api.semanticscholar.org/CorpusID:4844311>
- [39] V. Munasinghe, T.-H. Lee, H.-J. Lee, T. Sung Kim, and J.-S. Kim, "Multi-camera-based product recognition for inventory management," in *2023 International Conference on Electronics, Information, and Communication (ICEIC)*, 2023, pp. 1–3.
- [40] A. Kaur, Y. Singh, N. Neeru, L. Kaur, and A. Singh, "A survey on deep learning approaches to medical images and a systematic look up into real-time object detection," *Archives of Computational Methods in Engineering*, vol. 29, pp. 1–41, 10 2021.
- [41] L. Wen, X. Li, X. Li, and L. Gao, "A new transfer learning based on vgg-19 network for fault diagnosis," in *2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 2019, pp. 205–209.
- [42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015. [Online]. Available: <https://api.semanticscholar.org/CorpusID:206594692>
- [43] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *ArXiv*, vol. abs/1704.04861, 2017. [Online]. Available: <https://api.semanticscholar.org/CorpusID:12670695>
- [44] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2014. [Online]. Available: <https://api.semanticscholar.org/CorpusID:1629541>
- [45] Accessed on August 10, 2023. [Online]. Available: <https://www.icst.pku.edu.cn/cpdp/sjzy/>
- [46] J. Fang, X. Tao, Z. Tang, R. Qiu, and Y. Liu, "Dataset, ground-truth and performance metrics for table detection evaluation," in *2012 10th IAPR International Workshop on Document Analysis Systems*, March 2012, pp. 445–449.
- [47] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*, 2014. [Online]. Available: <https://api.semanticscholar.org/CorpusID:14113767>
- [48] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask r-cnn," 2017. [Online]. Available: <https://api.semanticscholar.org/CorpusID:54465873>
- [49] Accessed on August 10, 2023. [Online]. Available: https://de.overleaf.com/learn/latex/Wrapping_text_around_figures

- [50] L. Gao, Y. Huang, H. Dejean, J. L. Meunier, Q. Yan, Y. Fang, F. Kleber, and E. Lang, “Icdar 2019 competition on table detection and recognition (ctdar),” in *Proceedings of the 2019 International Conference on Document Analysis and Recognition (ICDAR)*, Sydney, Australia, September 2019, pp. 1510–1515.
- [51] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: <https://api.semanticscholar.org/CorpusID:6628106>
- [52] I. Agustina, F. Nasir, and A. Setiawan, “The implementation of image smoothing to reduce noise using gaussian filter,” *International Journal of Computer Applications*, vol. 177, pp. 15–19, 11 2017.