**Title: OCR for Product Labels: A Machine Learning and Machine Vision Approach**

**Problem:**

Manually extracting and entering data from product labels is a time-consuming process prone to human errors. This poses a social problem, particularly in industries such as e-commerce, inventory management, and quality control, where efficient data retrieval and analysis are essential. Traditional methods lack automation, hindering productivity and increasing labor costs. The Label OCR project addresses this problem by automating label extraction and recognition using computer vision and natural language processing techniques. By leveraging deep learning and OCR, the project aims to improve accuracy, streamline tasks, and enable faster data processing.

**Abstract**:

The Label OCR research addresses the social problem of manual data entry and extraction from product labels. This paper presents an approach that utilizes computer vision and natural language processing techniques to extract and recognize text labels from images. The implemented CRNN (Convolutional Recurrent Neural Network) model combines a CNN (Convolutional Neural Network) for image feature extraction and an RNN (Recurrent Neural Network) for sequence recognition. The Tesseract OCR engine is employed for text extraction. Extracted text undergoes preprocessing, including tokenization, lemmatization, and stop-word removal. The preprocessed labels are encoded using a label encoder and used for training the CRNN model. The trained model is evaluated for label recognition accuracy on a test set. Additionally, functionalities to display extracted and processed text, as well as visualize predictions and images, are provided. The results showcase the potential of deep learning and OCR techniques to automate label extraction and recognition, addressing the social problem of labor-intensive and error-prone manual data entry.

**Introduction:**

This journal publication presents a machine learning and machine vision approach for Optical Character Recognition (OCR) on product labels, specifically utilizing the CRNN model. The goal is to automatically extract and recognize text labels from product images, addressing the social problem of manual data entry and extraction. The CRNN model combines the strengths of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), allowing it to capture both visual features and sequential dependencies in label text. This results in accurate and reliable OCR outcomes. A comparative analysis with traditional OCR methods, including LSTM and transformer-based models, is conducted. While LSTM models excel in capturing sequential dependencies, the CRNN model outperforms them by leveraging the CNN's visual feature extraction capabilities. Transformer-based models offer high accuracy but require extensive computational resources and training data. The CRNN approach strikes a balance between accuracy and computational efficiency, making it well-suited for label OCR tasks. The experimental results demonstrate the effectiveness of the proposed approach, achieving high accuracy metrics at the character, word, and label levels. This performance evaluation showcases the potential for automation and efficiency in label extraction and recognition, benefiting industries such as e-commerce, inventory management, and quality control.

**Note: Add system diagram,**

**Study Diagram/Research Approach – explain each box in order.**

A picture containing text, screenshot, font, diagram

Description automatically generated

Methodology

2.1 Data Collection and Preprocessing

We collected a diverse dataset of product labels encompassing different industries and label designs. The dataset was manually extracted by the researcher that contains about 45 images in the format “.jpg” or “.png”. The nature of the dataset is modal since there is information extracted from ‘OpenFoodFacts’ API accessing through a free developer account. To standardize the text data, we applied various text preprocessing techniques, Natural Language Processing (NLP), including lowercase transformation, punctuation removal, and stop word removal.

2.2 Machine Learning Models

For the OCR tasks, we employed deep learning models, specifically Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). CNNs were used to extract visual features from label images, while RNNs were utilized for sequence modeling and character recognition. The models were trained on the labeled dataset using appropriate loss functions and optimization techniques.

CRNN (Convolutional Recurrent Neural Network) leverages both CNN (Convolutional Neural Networks) and RNN's strengths for image and sequence recognition. The workflow is akin to the previous approaches, with the training involving a CNN for image feature extraction and an RNN (such as LSTM or GRU) for sequence modeling. This combined model learns visual representations and captures label sequence dependencies.

**2.3 Machine Vision Techniques**

To enhance the OCR process, we incorporated machine vision techniques. Image preprocessing techniques such as image normalization, denoising, and contrast enhancement were applied to improve the quality of label images. Additionally, advanced image analysis algorithms, including contour detection and segmentation, were employed to isolate and extract individual characters or text regions from the labels.

2.3.1

**OCR algorithm: (add steps in bullet points) Description aside.**

The ‘process\_image’ method is responsible for processing an input image to extract relevant information from product labels. Additionally, the function follows a series of steps to enhance the image, perform text extraction, and extract additional data.

First, the image is loaded and converted to the LAB color space. The LAB image is then split into L, A, and B channels. The L channel undergoes Contrast Limited Adaptive Histogram Equalization (CLAHE) to enhance the image's contrast. The enhanced L channel is then merged with the A and B channels, resulting in an enhanced RGB image.

Next, sharpening is applied to the enhanced image to improve edge clarity. The sharpened image is then converted to grayscale for further processing. Noise removal is performed by applying a threshold to the grayscale image using Otsu's method.

To handle images that are predominantly black or white, the function checks the ratio of black and white pixels in the image. If the image is mostly black or white, it is inverted to ensure proper text extraction.

For color images, text regions are detected by performing dilation and finding contours. Small contours are filtered out, and the contour with the maximum area is identified as the text region. The region is cropped from the thresholded image, and text extraction is performed using the Tesseract OCR engine. The extracted text is then saved, and relevant data is extracted based on predefined keywords. In case of any exceptions during the image processing, error handling is implemented to log the error message and traceback.

Overall, the process\_image function provides a comprehensive pipeline for image preprocessing, text extraction, and data extraction from product labels, contributing to the automated extraction and analysis of label information.

**Experimental Results**

3.1 OCR Accuracy Evaluation

We evaluated the trained OCR model on a separate test dataset comprising diverse product labels. Accuracy metrics such as character-level accuracy, word-level accuracy, and label-level accuracy were calculated to assess the performance of the OCR system. The results demonstrated high accuracy rates, indicating the effectiveness of the proposed approach.

**3.2 Comparison with Traditional OCR Methods**

We compared our approach with traditional OCR methods, specifically LSTM, transformer-based models, and CRNN.

LSTM (Long Short-Term Memory): LSTM models, being a type of recurrent neural network (RNN), are well-suited for sequence modeling tasks like OCR. They capture dependencies in sequential data using memory cells that retain information for long periods. We described the LSTM workflow, including preprocessing, text extraction, feature extraction, LSTM training, model architecture, and training/evaluation steps.

Transformer-based Models: Transformer-based models, such as those used in the EasyOCR library, have demonstrated remarkable performance in OCR tasks. These models utilize self-attention mechanisms to capture contextual dependencies and enable parallel processing for efficient training and inference. We outlined the workflow, including preprocessing, text extraction, feature extraction, dataset creation, transformer-based model training, and evaluation steps.

CRNN (Convolutional Recurrent Neural Network): Our proposed CRNN approach combines the strengths of CNNs and RNNs for image and sequence recognition. The CNN component extracts visual features from images, while the RNN component models sequential dependencies in label text. We provided a detailed workflow, including preprocessing, text extraction, feature extraction, dataset creation, CRNN training, CRNN model architecture design, and training/evaluation steps.

By conducting a comparative analysis of these approaches based on accuracy, complexity, generalization, robustness, training efficiency, and inference speed, researchers can gain insights into their strengths and weaknesses. This analysis guides the selection of the most suitable approach for specific label OCR requirements, considering factors such as accuracy and computational resources.

**A bit more detailed comparison:**

LSTM (Long Short-Term Memory) is a recurrent neural network (RNN) model apt for OCR (Optical Character Recognition) tasks due to its memory cells' ability to retain long-term information. Its workflow includes preprocessing input images, text extraction using OCR techniques like Tesseract, feature extraction through keyword matching, and training an LSTM model using encoded label features. The model's architecture can include embedding layers, LSTM layers, and fully connected layers. It is trained on labeled data, validated, and evaluated for accuracy on a test set.

Transformer-based models, like those in the EasyOCR library, have shown promising results in OCR tasks due to their self-attention mechanisms and parallel processing capabilities. The workflow mirrors that of the LSTM approach, except for text extraction, where a transformer-based model like EasyOCR is used. These models require large datasets and computational resources for effective training, and their performance is evaluated through metrics like accuracy, precision, and recall.

CRNN (Convolutional Recurrent Neural Network) leverages both CNN (Convolutional Neural Networks) and RNN's strengths for image and sequence recognition. The workflow is akin to the previous approaches, with the training involving a CNN for image feature extraction and an RNN (such as LSTM or GRU) for sequence modeling. This combined model learns visual representations and captures label sequence dependencies.

In comparing these approaches, their accuracy is evaluated using metrics like precision, recall, and F1 score on a test dataset. The complexity analysis covers model architecture, training data requirements, and computational resources. The models' generalization capabilities are assessed by performance on unseen labels or different datasets, and their robustness is evaluated under challenging conditions like noisy or low-resolution images. Training efficiency includes comparisons of convergence speed and data requirements. Lastly, inference speed is measured to determine efficiency for large-scale OCR applications.

**Better comparison:**

3.2 Comparison with Traditional OCR Methods

Approaches for Label OCR Research**: LSTM vs. Transformer-based vs. CRNN**

LSTM (Long Short-Term Memory):

Approach:

The LSTM (Long Short-Term Memory) model is a type of recurrent neural network (RNN) that is well-suited for sequence modeling tasks like OCR. It has memory cells that can retain information for long periods, allowing it to capture dependencies in sequential data.

Workflow:

Preprocessing: Resize and normalize the input images, convert them to grayscale, and extract relevant regions using techniques like contour detection.

Text Extraction: Apply OCR (Optical Character Recognition) techniques such as Tesseract or pytesseract to extract text from the preprocessed image regions.

Feature Extraction: Use keyword matching or other techniques to identify specific label features, such as product names, directions, supplements, or warnings.

LSTM Training: Encode the extracted label features and use them to train an LSTM model. Prepare the data by tokenizing the text, encoding it numerically, and padding sequences to a fixed length.

LSTM Model Architecture: Design an LSTM model with appropriate input and output dimensions. It can include embedding layers, LSTM layers, fully connected layers, and softmax activation for multiclass classification.

Training and Evaluation: Train the LSTM model using labeled data, perform validation, and evaluate its accuracy on a test set.

Transformer-based Models:

Approach: Transformer-based models, such as those used in the EasyOCR library, have shown remarkable performance in OCR tasks. These models employ self-attention mechanisms to capture contextual dependencies and parallel processing for efficient training and inference.

Workflow:

Preprocessing: Similar to the LSTM approach, preprocess the images by resizing, normalizing, denoising, and converting them to grayscale.

Text Extraction: Use a transformer-based model, such as EasyOCR, to extract text directly from the preprocessed images. These models can handle multiple languages and offer high accuracy in OCR tasks.

Feature Extraction: Utilize keyword matching or similar techniques to identify relevant label features from the extracted text, similar to the LSTM approach.

Dataset Creation: Prepare the dataset by creating labeled data with the extracted label features and relevant image paths.

Transformer-based Model Training: Fine-tune a pre-trained transformer-based model on the labeled dataset. These models often require large amounts of training data and computational resources for effective training.

Evaluation: Evaluate the performance of the transformer-based model on a test set, measuring accuracy, precision, recall, and other relevant metrics.

CRNN (Convolutional Recurrent Neural Network):

Approach: CRNN combines the strengths of CNNs (Convolutional Neural Networks) and RNNs for image and sequence recognition. It leverages the CNN's ability to extract visual features from images and the RNN's capability to model sequential dependencies in label text.

Workflow:

Preprocessing: Preprocess the images by resizing, converting to grayscale, and potentially applying denoising techniques.

Text Extraction: Extract text from the preprocessed images using OCR techniques like Tesseract or pytesseract, similar to the LSTM and transformer-based approaches.

Feature Extraction: Identify relevant label features using keyword matching or other methods, similar to the previous approaches.

Dataset Creation: Prepare the labeled dataset with the extracted label features and image paths.

CRNN Training: Train a CRNN model by integrating a CNN for image feature extraction and an RNN (such as LSTM or GRU) for sequence modeling. CNN learns visual representations from the images, and the RNN captures dependencies in the label sequences.

CRNN Model Architecture: Design the CRNN model by combining the CNN and RNN components, where the CNN extracts visual features and the RNN processes the features to make predictions.

Training and Evaluation: Train the CRNN model on the labeled dataset, perform validation, and evaluate its performance on a test set, measuring accuracy and other relevant metrics.

**Comparative Analysis:**

**Accuracy:**

LSTM: Evaluate the accuracy of the LSTM approach by measuring its performance on the test dataset. Calculate metrics such as accuracy, precision, recall, and F1 score to assess its effectiveness in correctly identifying label features.

Transformer-based: Measure the accuracy of the transformer-based approach using the same evaluation metrics as LSTM. Compare the performance of the transformer-based model with the LSTM model to determine which approach achieves higher accuracy in label OCR tasks.

CRNN: Evaluate the accuracy of the CRNN approach by assessing its performance on the test dataset. Compare the accuracy of CRNN with LSTM and transformer-based models to determine the most accurate approach. – still working on this.

**Complexity**:

LSTM: Analyze the complexity of the LSTM approach in terms of model architecture, including the number of LSTM layers, hidden sizes, and input/output dimensions. Assess the training data requirements, such as the size of the labeled dataset and data preprocessing steps. Evaluate the computational resources, memory, and training time needed to train the LSTM model effectively.

Transformer-based: Assess the complexity of the transformer-based approach by analyzing the architecture of the model, including the number of transformer layers, attention heads, and model size. Evaluate the training data requirements, such as the amount of labeled data needed for effective training. Analyze the computational resources, memory, and training time required for training the transformer-based model.

CRNN: Analyze the complexity of the CRNN approach by considering the model architecture, including the configuration of CNN and RNN layers. Evaluate the data requirements, computational resources, memory, and training time needed for training the CRNN model.

**Generalization**:

LSTM: Assess the generalization capabilities of the LSTM approach by evaluating its performance on unseen labels or different datasets. Measure its ability to handle variations in label layouts, font styles, and sizes.

Transformer-based: Evaluate the generalization capabilities of the transformer-based approach by testing it on unseen labels or different datasets. Assess its performance in handling variations in label characteristics and languages.

CRNN: Assess the generalization capabilities of the CRNN approach by evaluating its performance on unseen labels or different datasets. Measure its ability to handle variations in label layouts, font styles, and sizes, similar to LSTM.

**Robustness:**

LSTM: Evaluate the robustness of the LSTM approach by testing it on challenging scenarios such as noisy or low-resolution images. Measure its performance in accurately extracting label features under such conditions.

Transformer-based: Assess the robustness of the transformer-based approach by testing it on noisy or low-resolution images. Measure its ability to handle image variations and noise while extracting label information.

CRNN: Evaluate the robustness of the CRNN approach by testing it on challenging scenarios such as noisy or low-resolution images. Measure its performance in accurately extracting label features under such conditions.

**Training Efficiency:**

LSTM: Compare the training efficiency of the LSTM model with the other approaches in terms of convergence speed. Analyze the data requirements for training the LSTM model effectively, such as the size and quality of the labeled dataset.

Transformer-based: Compare the training efficiency of the transformer-based model with the other approaches. Assess its convergence speed and data requirements for training, including the availability of pre-trained models and transfer learning possibilities.

CRNN: Compare the training efficiency of the CRNN model with LSTM and transformer-based models. Analyze the convergence speed and data requirements, including the need for pre-trained models and transfer learning possibilities.

**Inference Speed:**

LSTM: Measure the inference speed of the LSTM model to identify its efficiency for real-time or large-scale OCR applications. Evaluate the time taken by the model to process individual label images or batches of images.

Transformer-based: Measure the inference speed of the transformer-based model to identify its efficiency for real-time or large-scale OCR applications. Evaluate the time taken by the model to process individual label images or batches of images.

CRNN: Measure the inference speed of the CRNN model to identify its efficiency for real-time or large-scale OCR applications. Evaluate the time taken by the model to process individual label images or batches of images.

By performing a comparative analysis of the LSTM, transformer-based, and CRNN approaches based on accuracy, complexity, generalization, robustness, training efficiency, and inference speed, researchers can gain insights into the strengths and weaknesses of each approach. This analysis can guide the selection of the most suitable approach for specific label OCR requirements, considering factors such as accuracy, computational resources, training data availability, and real-time processing needs.

**Applications and Limitations (methods that do not work):**

**MMOCR**

MMOCR is an OCR toolbox that provides a comprehensive set of tools for optical character recognition. To use MMOCR, Microsoft Visual C++ 14.0 or greater must be installed on the system to compile the pycocotools package, a dependency for MMOCR.

The installation process for MMOCR involves installing several dependencies using the "openmim" package manager. These dependencies include "mmengine," "mmcv" (version 2.0.0rc1 or later), and "mmdet" (version 3.0.0rc0 or later). Detailed installation instructions can be found on the MMOCR documentation website.

MMOCR offers two main components for OCR: detection and recognition.It's worth noting that while MMOCR offers high accuracy in text recognition, it has some limitations. The approach struggles with images of different scales, often resulting in extremely small images. Resizing the images manually can help overcome this issue. Furthermore, based on the MMOCR official documentation, the training of the model can take up to 400 times to be trained and tested to provide high accuracy which will require much more powerful computer with high speed. Additionally, the performance of the MMOCR approach is slower compared to other OCR methods.

In conclusion, MMOCR provides a powerful OCR toolbox with efficient detection and recognition capabilities. By following the installation instructions and utilizing the provided code examples, users can leverage MMOCR to extract text from images effectively. However, it is important to consider the limitations, such as the need for manual resizing of images and the slower processing speed, when using MMOCR for OCR tasks.

**EAST and CRAFT**

The code provided implements text detection using the CRAFT (Character Region Awareness for Text Detection) algorithm. It performs text detection on both black-and-white and color images. The CRAFT algorithm involves several steps, including text region detection using the CRAFT model and subsequent preprocessing and OCR steps.

The CRAFT algorithm involves steps such as text region detection using the CRAFT model, contrast enhancement, noise removal, CLAHE for even lighting, and Otsu's thresholding.

Similarly, the EAST algorithm relies on a fully convolutional neural network (FCN) for text detection. However, the EAST approach also faces issues with optimization and modularity, as indicated by the encountered C++ errors from the OpenCV library. The limitations of the EAST algorithm for product label OCR include challenges in capturing complex designs and layouts accurately, difficulty in detecting small text or fine details, and potential interference from background noise or clutter.

Both the CRAFT and EAST algorithms lack semantic understanding of text regions, treating all regions equally without considering their specific context or structure. This limitation is particularly relevant for product label OCR, where different text regions often convey different types of information. To overcome these challenges, alternative approaches tailored for product label OCR can be explored. These approaches may involve combining text detection methods that account for the unique characteristics of product labels, employing specialized preprocessing techniques, utilizing domain-specific models, and incorporating techniques for semantic understanding of the text content. Customizing the OCR pipeline to address the specific challenges of product labels can improve the accuracy and reliability of OCR results.

**BLOB/Boxes.**

The provided code implements text detection using the BLOB (Binary Large Object) approach. It includes functions for extracting bounding boxes, cropping images, applying non-maximum suppression, rescaling boxes, and filtering boxes based on specific criteria. However, a major limitation of this approach is that it encounters an issue where there are more than 1600 boxes per image, causing the algorithm to get stuck in a loop for each box and resulting in extremely slow performance. This significantly decreases the practical usability of the BLOB approach.

To address this limitation, an alternative contour-based approach can be considered. The contour-based approach involves detecting contours in the image and filtering them based on size, shape, or hierarchy. This technique significantly reduces the number of regions to be processed, leading to improved algorithm speed. By leveraging contour properties and incorporating additional techniques like edge detection, the contour-based approach offers a more efficient and accurate solution for text detection.

In summary, the BLOB approach provides functions for text detection but suffers from poor performance due to the large number of boxes per image. To overcome this limitation, exploring the contour-based approach and incorporating contour filtering and edge detection techniques can offer a more efficient and accurate text detection solution.

**OCR Space website API.**

The OCR space website provides an OCR (Optical Character Recognition) API for extracting text from images. OCR space provides a simple and convenient OCR (Optical Character Recognition) API for parsing images and multi-page PDF documents, returning the extracted text results in a JSON format. The API offers three tiers: Free, PRO, and PRO PDF. The Free OCR API plan is suitable for users who want to explore and test the OCR functionality. It comes with a rate limit of 500 requests per day per IP address to prevent unintended spamming. This plan allows developers to get started without any financial commitment.

The OCR space website provides an OCR (Optical Character Recognition) API for extracting text from images. However, there are limitations associated with using the free version of the API. One limitation is that the API commands can only be used every 600 seconds. Additionally, the image size must be less than 1 MB to be processed by the API. However, there are other payable deals that offer great solutions to general image OCR implementations.

It is also important to note that while the Free OCR API plan has certain limitations such as the rate limit and file size limit of 1 MB, the PRO plans offer expanded capabilities, including larger file size support, higher request volumes, and customizable OCR servers.

For users seeking faster response times and guaranteed uptime, the PRO plans are available. The PRO OCR API runs on high-performance servers located in the US, EU, and Asia regions.

Additionally, the PRO OCR API can be purchased as a locally installable on-premises OCR software, providing organizations with the flexibility to have their OCR servers set up at a location of their choice. This option is suitable for users with specific data security or compliance requirements.

**Add Google Vision api, great option, but you got to pay. Complicated, need developer account, computation expensive.**

**5. Conclusion**

In this study, we have presented a machine learning and machine vision approach for label OCR using the CRNN model. By integrating CNNs and RNNs, the proposed approach achieves accurate and reliable text extraction from product labels. The combination of computer vision techniques, natural language processing, and deep learning enables the automation of label recognition tasks, leading to improved efficiency and data accessibility.

Future research directions include exploring additional enhancements to the OCR pipeline, such as incorporating advanced image preprocessing techniques, optimizing hyperparameters of the CRNN model, and expanding the dataset to encompass a wider range of label designs and languages. Moreover, the integration of domain-specific knowledge and semantic understanding could further improve the accuracy and interpretability of the OCR system.

Overall, the label OCR research opens opportunities for advancements in automation and data extraction from product labels, enabling streamlined processes and facilitating information retrieval in various industries. By harnessing the power of machine learning and machine vision, we can revolutionize the way label data is handled and utilized, ultimately driving innovation and efficiency in product-related workflows.

**Research links:**

<https://pytorch.org/docs/stable/index.html>

<https://pillow.readthedocs.io/en/stable/index.html>

<https://github.com/pytorch/pytorch>

<https://docs.opencv.org/4.x/>

<https://tesseract-ocr.github.io/tessdoc/Home.html>

<https://www.hindawi.com/journals/complexity/2019/1671340/>

<https://github.com/openfoodfacts/off-nutrition-table-extractor>

<https://stacks.stanford.edu/file/druid:bf950qp8995/Grubert_Gao.pdf>

<https://towardsdatascience.com/5-open-source-tools-you-can-use-to-train-and-deploy-an-ocr-project-8f204dec862b>

<https://arxiv.org/pdf/2103.14470.pdf>

Chauhan, Amit. “Drawing Bounding Box Method in Image Processing.” *Medium*, 26 Nov. 2022, https://pub.towardsai.net/drawing-bounding-box-method-in-image-processing-ec7487393cfa.

*Free OCR API*. https://ocr.space/OCRAPI#local. Accessed 19 June 2023.

*How to Find the Bounding Rectangle of an Image Contour in OpenCV Python*. https://www.tutorialspoint.com/how-to-find-the-bounding-rectangle-of-an-image-contour-in-opencv-python. Accessed 19 June 2023.

*Install the Gcloud CLI | Google Cloud*. https://cloud.google.com/sdk/docs/install. Accessed 19 June 2023.

*MCREU\_Label\_OCR/NotWork/*. http://localhost:8888/tree/MCREU\_Label\_OCR/NotWork. Accessed 7 July 2023.

Medicine (US), Institute of, et al. *Corporate Social Responsibility*. National Academies Press (US), 2007, https://www.ncbi.nlm.nih.gov/books/NBK53982/.

MMOCR Contributors. *OpenMMLab Text Detection, Recognition and Understanding Toolbox*. 2021. 0.3.0, 2020, https://github.com/open-mmlab/mmocr.

nathancy. “Answer to ‘How to Detect Paragraphs in a Text Document Image for a Non-Consistent Text Structure in Python OpenCV.’” *Stack Overflow*, 29 July 2019, https://stackoverflow.com/a/57262099.

Rosebrock, Adrian. “OpenCV Text Detection (EAST Text Detector).” *PyImageSearch*, 20 Aug. 2018, https://pyimagesearch.com/2018/08/20/opencv-text-detection-east-text-detector/.

“Social Responsibility.” *Wikipedia*, 24 Oct. 2018, https://en.wikipedia.org/w/index.php?title=Social\_responsibility&oldid=865554318.

Sun, Hongbin, et al. *Spatial Dual-Modality Graph Reasoning for Key Information Extraction*. arXiv:2103.14470, arXiv, 26 Mar. 2021, http://arxiv.org/abs/2103.14470.

Yon, Lai Woen. “5 Open Source Tools You Can Use to Train and Deploy an OCR Project.” *Medium*, 30 Nov. 2021, https://towardsdatascience.com/5-open-source-tools-you-can-use-to-train-and-deploy-an-ocr-project-8f204dec862b.