Document Image Cleaning using Budget-Aware Black-Box Approximation

by

Ganesh Tata

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Recent work has shown that by approximating the behaviour of a non-differentiable black-box function using a neural network, the black-box can be integrated into a differentiable training pipeline for end-to-end training. This methodology is termed “differentiable bypass,” and a successful application of this method involves training a document preprocessor to improve the performance of a black-box OCR engine. However, a good approximation of an OCR engine requires querying it for all samples throughout the training process, which can be computationally and financially expensive. Several zeroth-order optimization (ZO) algorithms have been proposed in black-box attack literature to find adversarial examples for a black-box model by computing its gradient in a query-efficient manner. However, the query complexity and convergence rate of such algorithms makes them infeasible for our problem. In this work, we propose two sample selection algorithms to train an OCR preprocessor with less than 10% of the original system’s OCR engine queries, resulting in more than 60% reduction of the total training time without significant loss of accuracy. We also show an improvement of 4% in the word-level accuracy of a commercial OCR engine with only 2.5% of the total queries and a 32x reduction in monetary cost.

Moreover, we propose a simple ranking technique to prune 30% of the document images from the training dataset without significantly affecting the system's performance. Finally, we demonstrate that the history of OCR engine predictions for each sample throughout the training process further improves the system's performance in a low query setting.

Preface

Some parts of this thesis have been submitted as a journal paper to the *Pattern Recognition* journal. The paper is co-authored by Katyani Singh, Eric Van Oeveren and Nilanjan Ray.

Acknowledgements

I would like to thank my supervisor Nilanjan Ray for his constant guidance and support. The regular discussions with him involving research ideas and experiments were crucial for the success of the project. I would also like to thank Katyani Singh for proofreading my thesis drafts and providing valuable inputs regarding the methodology. Additionally, I would like to express my gratitude to Eric Van Oeveren from Intuit Inc. for providing valuable feedback for the research work. I would also like to acknowledge the funding support provided by Intuit AI Research and NSERC.

Contents

Abstract **ii**

Preface **iv**

Acknowledgements **v**

List of Tables **ix**

List of Figures **x**

List of Algorithms **xiii**

1 Introduction **1**

1.1 Problem Statement 1

1.2 Contributions 5

1.3 Thesis Outline 6

2 Background **7**

2.1 Differentiable Bypass 7

2.1.1 Integrating Black-Box Functions in Training Pipelines 7

2.1.2 Document Cleaning using Black-Box Approximation 12

2.2 Data Subset Selection for Efficient DNN Training 16

2.2.1 Mini-Batch Sample Selection 16

2.2.2 Epoch Sample Selection 18

2.2.3 Dataset Sample Selection 19

2.3 Tracking Past Information 19

3 Methods **21**

3.1 Selecting Samples to Query OCR Engine 21

3.1.1 UniformCER 21

3.1.2 TopKCER 24

3.2 Pruning Document Images 24

3.3 Label Tracking for Utilizing History Information 25

4 Experiments and Results **29**

4.1 Experiment Setup 29

4.1.1 Datasets 29

4.1.2 Training Details 30

4.2 Results 34

4.3 Additional Experiments 37

4.3.1 Training and Testing on Different OCR Engines 37

4.3.2 Performance across Different Query Budgets 38

4.3.3 Effect of MSE Loss 40

4.3.4 Pruning Receipt Images 41

4.3.5 Label Tracking Experiments 44

5 Conclusion and Future Work **45**

References **46**

Bibliography **47**

Appendix A Additional Results and Hyperparameter Tuning **56**

A.1 Character Error Rate (CER) of OCR Engine 56

A.2 Comparison of 0% and 2.5% Query Budget 57

A.3 Hyperparameter Tuning for CRNN and UNet 57

List of Tables

4.1 Word-level accuracy (%) with Tesseract for different selection methods. No selection methods were used for 0% and 100% budgets. The OCR engine is never queried for 0% budget. 33

4.2 Word-level accuracy (%) with EasyOCR for different selection methods. No selection methods were used for 0% and 100% budgets. The OCR engine is never queried for 0% budget. 34

4.3 Test set word-level accuracy (%) of preprocessor trained with Google Vision API along with expected and actual cost of training the system. 36

4.4 Word-level accuracy (%) for OCR trained and tested on different engines using best-performing preprocessors from Tables 4.1,4.2 and 4.3. *The preprocessor was trained with 50% query budget. 38

4.5 Comparison of test accuracy (%) between noise addition and label tracking. 44

A.1 Average Character Error Rate (CER) of OCR Engine ($\times$100) for different selection methods across POS and VGG datasets. No selection methods were used for 0% and 100% budgets. Lower CER indicates better text recognition performance. 58

List of Figures

1.1 Test set performance of OCR engines on noisy and preprocessed images for the POS dataset with 4% and 100% of the total OCR queries using UniformCER selection. Even though the Google Vision API performs well for the noisy images, our methodology further improves its performance in a cost-efficient manner. We do not report results for Google Vision API with 100% budget since the monetary cost of running that experiment is too high (Table 4.3). 3

2.1 Sequential arrangement of n differentiable functions. ϕ_i and z_i are the parameters and output of function f_i, respectively. 8

2.2 Differentiable bypass for end-to-end training of system with sequentially arranged functions containing non-differentiable functions like f_i. g is a differentiable proxy that approximates the functionality of f_i. Dashed black lines indicate forward pass and dashed green lines depict backward pass during training. 10

2.3 **Training pipeline**. An overview of the training pipeline [1]. The broken green lines indicate backpropagation while the broken black lines depict only forward propagation to illustrate the training of the preprocessor using differentiable bypass. Receipt and document images are split into text strips after preprocessing. 13

2.4 UNet architecture used by Randika et al. [1] to clean document images. 14

2.5 CRNN architecture used by Randika et al. [1] to approximate the gradient of black-box OCR engines. 15

2.6 Sample Selection for accelerating neural network training. 17

3.1 Sample selection for the VGG dataset using a selection algorithm after images are passed through the preprocessor. 22

3.2 Samples selection for the POS dataset using a selection algorithm after dcoument images are passed through the preprocessor. 23

3.3 Samples selection for the POS dataset using a selection algorithm before training the preprocessor using differentiable bypass. 25

3.4 Accumulating history information for training samples and utilizing the past predictions to compute the loss function for updating the approximator using label tracking. 27

4.1 Sample Images from the POS Dataset. 30

4.2 Sample Images from the VGG Dataset. 31

4.3 Test set word-level accuracy (%) of OCR engine with noise added once (50% budget) and twice (100% budget) for POS and VGG datasets. 32

4.4 Preprocessor output for different test samples in the POS dataset. The preprocessors were trained with Tesseract OCR engine and UniformCER selection was used for 4% % budget. **Column 1:** Original Images. **Column 2:** Original system - 100% query budget. **Column 3:** 4% query budget. 35

4.5 Preprocessor output for different test samples in the VGG dataset. The preprocessors were trained with Tesseract OCR engine and UniformCER selection was used for 4% % budget. **Column 1:** Original Images. **Column 2:** Original system - 100% query budget. **Column 3:** 4% query budget. 37

4.6 (a) Test set performance, (b) CRNN accuracy with OCR predictions as Ground Truth and (c) system training time, with UniformCER selection and Tesseract across different query budgets. 39

4.7 Effect of MSE loss coefficient β on preprocessor trained with Uniform-CER selection, Tesseract and 8% budget. 40

4.8 Test Accuracy and Tesseract queries (%) across different data pruning rates. The numbers in blue indicate query % using single jitter with respect to the original system. 42

4.9 Samples with 0 *mean CER* that were pruned since they rank lowest with respect to Tesseract's CER. 42

4.10 Performance of system with original and pruned POS dataset using UniformCER sample selection for reducing queries to Tesseract. . . . 43

A.1 Preprocessor output for few randomly selected images in the POS dataset. The preprocessors were trained with Tesseract OCR engine and UniformCER selection was used for 2.5 % budget. **Column 1:** Original Images. **Column 2:** 0% query budget. **Column 3:** 2.5% query budget. 59

List of Algorithms

1 Efficient approximation of OCR engine for document image cleaning using sample selection 22
2 UniformCER Selection Algorithm 24
3 Efficient OCR approximation for document image cleaning using sample selection with and w/o label tracking 26

Chapter 1

Introduction

1.1 Problem Statement

Optical Character Recognition (OCR) is the process of extracting text in images like scanned documents and point of sale (POS) receipt images. An OCR system (also called an OCR engine) consists of two main components - a text detector to obtain bounding boxes of text in the image and a text recognizer to recognize the text in the bounding boxes. Recently, a number of text detection [2][3][4] and text recognition [5][6][7] neural networks have been proposed in literature. Based on these recent advances, several open-source and commercial OCR engines have been made available. The commercial OCR engines can be accessed through cloud APIs since they are provided as a SaaS (Software as a Service) product.

In some machine learning systems, a non-differentiable black-box function is approximated using a differentiable surrogate model to facilitate end-to-end training using gradient-based methods [1][8][9][10]. The effectiveness of this "differentiable bypass" approach can be particularly evidenced through the improvement in performance achieved by training a preprocessor for a black-box Optical Character Recognition (OCR) engine [1]. A neural network called the "approximator" is trained to approximate the behaviour of the black-box OCR engine. However, training a good black-box approximator requires several queries of the OCR engine, which can be computationally expensive for open-source engines like Tesseract[1] and EasyOCR[2], or incur a high financial cost for proprietary OCR APIs like the Google Cloud Vision API[3]. In this regard, our work focuses on reducing the number of queries of an OCR engine when training a surrogate model for efficient training of the OCR preprocessor.

For OCR, several commercial and open-source solutions have been made available. Commercial OCR systems are usually trained on many different types of documents since they are used for various client use cases, making them powerful OCR engines. However, fine-tuning commercial OCR APIs is not straightforward, while fine-tuning open-source OCR engines requires a good understanding of their re-training process, which can be cumbersome. For instance, both Tesseract and EasyOCR have separate instructions to retrain the OCR engine for a new dataset[4] [5]. The differentiable bypass [1] approach significantly improves the performance of OCR engines like Tesseract and EasyOCR without fine-tuning the engine itself. While this approach boosts OCR performance, there is an associated tradeoff regarding queries made to the OCR engine, which can incur high financial/computational costs for training, especially when APIs like Google Vision API are used. The high costs associated with training such a preprocessor underscore the need for improving the query efficiency of the differentiable bypass system to achieve good text recognition performance at a fraction of the system's total OCR queries.

Zeroth-order optimization (ZO) techniques have been used to estimate the gradient of a function by querying it at different points without using the function's first order derivative [11][12] [13]. ZO methods have been particularly effective for query-efficient black-box attacks [14][15][16]. However, such methods require *at least two evaluations* of the function for each sample to estimate the gradient of the function.

[1]https://github.com/tesseract-ocr/tesseract
[2]https://github.com/JaidedAI/EasyOCR
[3]https://cloud.google.com/vision/docs/ocr
[4]https://tesseract-ocr.github.io/tessdoc/tess4/TrainingTesseract-4.00.html
[5]https://github.com/JaidedAI/EasyOCR/blob/master/custom_model.md

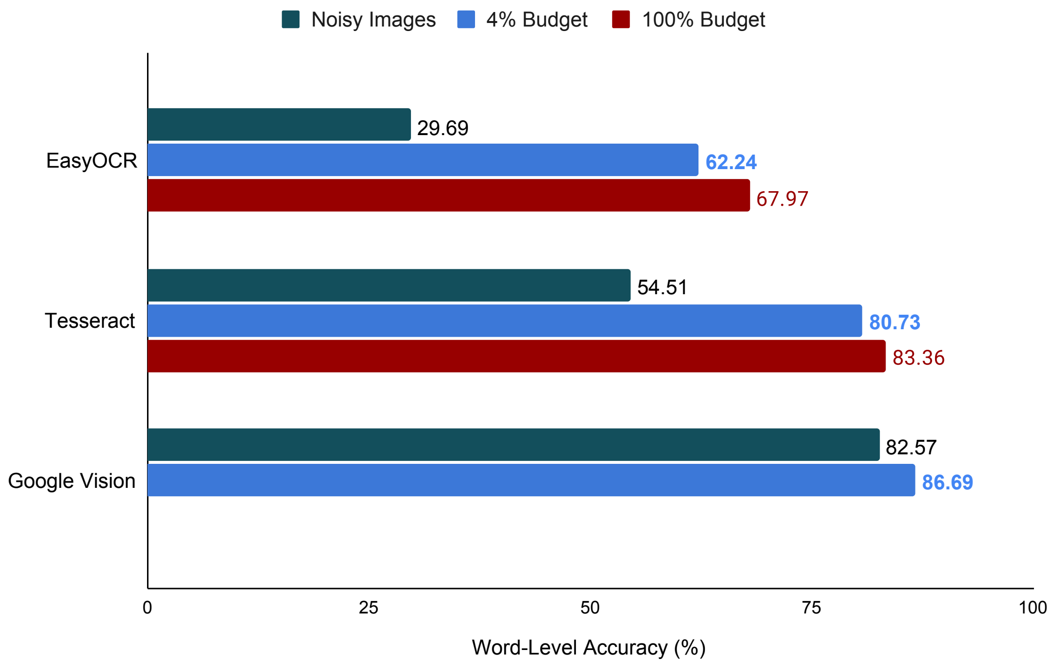


Figure 1.1: Test set performance of OCR engines on noisy and preprocessed images for the POS dataset with 4% and 100% of the total OCR queries using UniformCER selection. Even though the Google Vision API performs well for the noisy images, our methodology further improves its performance in a cost-efficient manner. We do not report results for Google Vision API with 100% budget since the monetary cost of running that experiment is too high (Table 4.3).

The convergence rate of many such methods also becomes higher as the dimensionality of the parameter space increases. Hence, ZO methods are *not suitable* for approximating the OCR with fewer queries. The score function approximator approach (SFE) [1] trains the preprocessor for the OCR by computing the gradient of the black-box OCR using the REINFORCE algorithm [17]. However, even with 10 queries of the OCR for each sample in every epoch, the system's performance does not match that of the differentiable bypass approach. Hence it is clear that the SFE method would perform worse in a low-query regime.

Recently, data-efficient training of neural networks has become a key area of research. While recent advances in deep learning have achieved state-of-the-art performance in several domains, they have also led to huge computational costs, increased carbon footprint, high financial costs, and increased training time [18] [19]. Training models with smaller subsets of data can yield faster training cycles and a quicker turnaround time for hyperparameter tuning. Some recent scholarship has focused on adaptive subset selection [20] to choose a smaller number of samples for training neural networks without degrading performance. These methods include core-set selection [21], gradient-based scoring [22], and loss-based prioritization [23]. Inspired by these methods, we propose two simple selection algorithms, UniformCER and TopKCER, which select a small subset of samples in each training mini-batch for querying the OCR engine and for subsequently updating the parameters of the approximator. Both UniformCER and TopKCER select samples by utilizing the approximator's Character Error Rate (CER), which is based on Levenshtein Distance [24], to measure the hardness of each sample. Additionally, these selection algorithms add no extra computational overhead to the system. As shown in Fig. 1.1, sample selection using UniformCER with a very low query budget (4%) can improve the text recognition performance for different OCR engines. We also propose a simple technique to prune document images before training the system to further reduce the OCR engine queries. Pruning is performed by ranking document images using the OCR engine's CER and removing a proportion of low-ranking images so that the system can be trained with the pruned dataset without significantly changing the text recognition performance.

Information accumulated over time has been used to improve sample efficiency of algorithms in reinforcement learning [25] and continual learning [26]. Similarly, we hypothesize that utilizing past OCR engine predictions can help improve the performance of the system on low query budgets. In this regard, we propose an algorithm called *label tracking* to track each sample's past OCR engine queries and incorporate them in a multi-objective loss function for training the approximator. The preprocessor is updated in each epoch using differentiable bypass, which constantly changes the input to the OCR engine. The variation in preprocessed images yields different

OCR engine outputs at different stages of training for the same sample. Hence, label tracking allows the approximator to learn from previous predictions of the same sample. Combined with UniformCER selection, label tracking improves the text recognition performance of EasyOCR and Tesseract with minimal computational overhead to the system.

1.2 Contributions

- In this work, we propose two selection methods, UniformCER and TopKCER, to query the OCR engine for a smaller subset of samples without significantly reducing OCR performance and with an overall query budget of less than 10%. Training the system with less than 10% query budget leads to more than 60% reduction in total training time. We also show that increasing the query budget beyond 10% leads to a larger training time for the system and marginal improvement in OCR performance.

- We demonstrate that UniformCER and TopKCER selection methods outperform random sampling for two low query budgets.

- We use a simple ranking technique to prune a subset of receipt images from the training dataset without significant reduction in test accuracy for the trained preprocessor. Pruning the dataset implicitly leads to fewer queries of the OCR engine.

- We propose label tracking to utilize the history information of OCR engine predictions for each sample and improve the performance of UniformCER selection in most settings.

1.3 Thesis Outline

Chapter 2 discusses different techniques for integrating black-box functions into differentiable training pipelines, specifically focusing on differentiable bypass and its application in document clearning for black-box OCR engines. As part of Chapter 2, we also discuss two sample selection algorithms for efficient training of Deep Neural networks. Chapter 3 describes our proposed sample selection techniques for reducing expensive OCR engine queries. Furthermore, the data pruning and label tracking methodology is described in the second half of Chapter 3. Chapter 4 presents the experiment setup, the text recognition performance corresponding to selection algorithms at low query budgets, and additional experiments analyzing the impact of different design choices in our setup. The last two sections of Chapter-4 provide insights into the effect of data pruning and label tracking on the performance of the system. Finally, Chapter 5 provides the conclusion for the thesis and also discusses potential future work.

Chapter 2

Background

2.1 Differentiable Bypass

2.1.1 Integrating Black-Box Functions in Training Pipelines

Machine learning systems can be constructed as a composition of different functions, where each function can either be differentiable or non-differentiable. Let us consider a system h composed of n sequentially arranged functions given by $f_1, f_2, ..., f_{n-1}, f_n$ (Fig. 2.1) where an input x is given to the first function f_1, the output of f_{i-1} is input to f_i for $i = 2, .., n-1$ and the output of f_n is input to a loss function L. In a supervised learning setting, L can be evaluated as $L(h(x), y)$ for input x and its corresponding ground-truth label y. Here, we assume that each function f_i is differentiable with respect to its parameters ϕ_i. If z_i is the output of f_i, then the gradient of loss function L with respect to ϕ_n (parameters of the last function f_n) is given by

$$\frac{\partial L}{\partial \phi_n} = \frac{\partial L}{\partial z_n} \frac{\partial z_n}{\partial \phi_n} \tag{2.1}$$

Using backpropagation, the error from the last function can be propagated to earlier functions in the sequence using the chain rule.

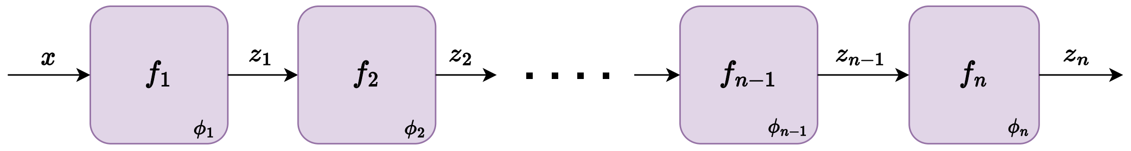


Figure 2.1: Sequential arrangement of n differentiable functions. ϕ_i and z_i are the parameters and output of function f_i, respectively.

$$\frac{\partial L}{\partial z_i} = \frac{\partial L}{\partial z_{i+1}} \frac{\partial z_{i+1}}{\partial z_i} \tag{2.2}$$

Computing the gradient with respect to z_i allows us to calculate gradient of L with respect to ϕ_i.

$$\frac{\partial L}{\partial \phi_i} = \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial \phi_i} \tag{2.3}$$

Thus, if all components of the system are differentiable, then such a system can be trained using the backpropagation algorithm, i.e., the parameters of individual components can be updated using gradient descent. A typical gradient descent update is as follows, where α is the learning rate

$$\phi_i^{t+1} = \phi_i^t - \alpha \frac{\partial L}{\partial \phi_i^t} \tag{2.4}$$

However, if a function is non-differentiable, it is not possible to compute the derivative of the function's output with respect to its parameters since the parameters of the function are unknown. The only option is to query the function at certain input points to get the corresponding output. Thus, if a system with sequentially arranged functions consists of non-differentiable functions, then it is not possible to train all the differentiable components of the system in an end-to-end fashion using backpropagation.

To alleviate this problem, we can attempt to approximate the gradient of the black-box function. The Straight Through Estimator (STE) [27] has been used to estimate the gradient of threshold functions in neural networks. However, STE is not useful for functions other than the threshold function. The Score Function Estimator (SFE) [28] allows us to estimate the gradient of a function using the log-derivative trick and the Monte-Carlo estimator. Using SFE, the gradient of function f can be estimated as

$$\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)] = \mathbb{E}_{x \sim p_\theta(x)}[f(x) \nabla_\theta \log p_\theta(x)] \tag{2.5}$$

The gradient can be computed using Monte-Carlo Estimation

$$\mathbb{E}_{x \sim p_\theta(x)}[f(x) \nabla_\theta \log p_\theta(x)] \approx \frac{1}{n} \sum_{i=1}^{n} f(x^{(i)}) \log p_\theta(x^{(i)}) \tag{2.6}$$

The policy gradient algorithm REINFORCE [17] uses SFE to optimize the parameters of a policy in a Reinforcement Learning (RL) setting. While SFE is an unbiased estimator, it also exhibits high variance. Variance reduction techniques based on control variates have been used to lower the variance of SFE. Alternatively, Salimans et.al. [29] use natural evolutionary strategies (NES) [30] to optimize the parameters of a policy to maximize reward in RL. By employing a reparameterization trick, the gradient of a black-box function can be estimated by

$$\nabla_\theta \mathbb{E}_{x \sim p_\theta(x)}[f(x)] = \frac{1}{\sigma} \mathbb{E}_{\epsilon \sim \mathbb{N}(0,I)}[f(\theta + \sigma\epsilon)] \tag{2.7}$$

Computing equation 2.7 with monte-carlo estimation yields a lower variance in comparison to SFE.

The differentiable bypass methodology approximates the gradient of a black-box function using a proxy model to integrate it into differentiable training pipelines. Let us assume that f is a non-differentiable black-box function and g is a differentiable function with parameters θ that approximates f. For instance, if f takes an image as input and outputs a class label, then g can be a Convolutional Neural Network (CNN) that receives an image input and yields a classification score for different class

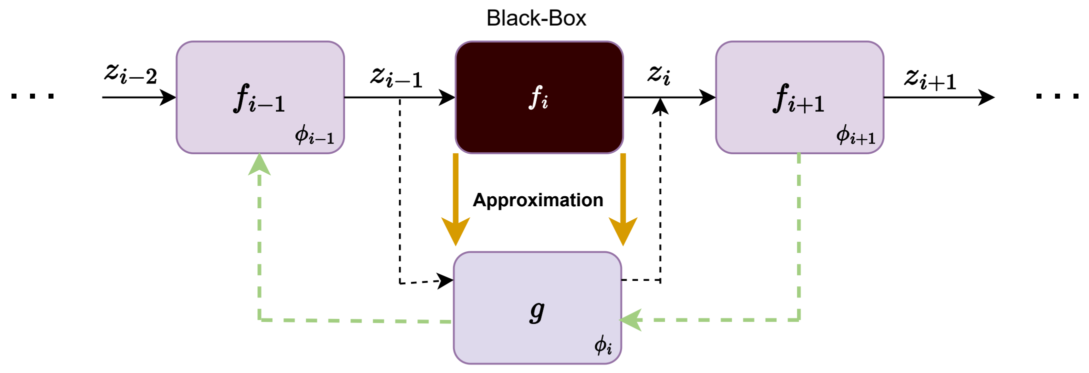


Figure 2.2: Differentiable bypass for end-to-end training of system with sequentially arranged functions containing non-differentiable functions like f_i. g is a differentiable proxy that approximates the functionality of f_i. Dashed black lines indicate forward pass and dashed green lines depict backward pass during training.

labels. The goal is to optimize θ such that g approximates f well for the given input, i.e.,

$$g(x; \theta) \approx f(x) \tag{2.8}$$

This can be achieved by optimizing θ with respect to a loss function L' that measures the fit between f and g. We can obtain optimal parameters θ^* using

$$\theta^* = min_\theta L'(g(x; \theta), f(x)) \tag{2.9}$$

A suitable choice for g are artificial Neural Networks (NN) since they are universal function approximators [31]. The universal function approximation theorem has also been shown to hold for CNNs [32]. Further, Hornik et al. [33] show that neural networks can also approximate the gradient of a function along with its output, which indicates that if g is a NN and it approximates the output of f, then, for the optimal parameters θ^*

$$\nabla_x f \approx \nabla_x g_{\theta^*} \tag{2.10}$$

Therefore, since $\nabla_x f$ cannot be computed easily, its approximation $\nabla_x g_{\theta^*}$ is used to successfully perform backpropagation through f and update the parameters of differentiable components. Figure 2.2 demonstrates the differentiable bypass methodology.

Differentiable bypass has been used in various applications to integrate a black-box into a differentiable training pipeline. Nguyen and Ray [10] propose EDPCNN, a CNN trained to perform left ventricle segmentation by approximating the behaviour of a non-differentiable Dynamic Programming (DP) module using a neural network. In EDPCNN, an MRI image is passed as input to a UNet model, which generates an output segmentation map. The output map is warped using a star pattern, and a DP module further processes the warped map to output the final segmentation map. However, the DP module is non-differentiable due to the argmin function. Hence, to facilitate end-to-end training of the system and update the parameters of the UNet, the DP module is approximated using a Neural Network. The authors show that EDPCNN can accurately perform left-ventricle segmentation with less labeled data while training only a UNet for the task needs more labeled data. They also propose the addition of random perturbations to the input so that the neural network fits the DP module effectively. Equation 2.9 can be rewritten to include exploration in the input space

$$\theta^* = min_\theta L'(g(x+\epsilon;\theta), f(x+\epsilon)) \tag{2.11}$$

EstiNet [8] is a general framework for training a differentiable estimator as a proxy for a black-box function to facilitate the composition of different black-box functions and trainable modules. Jacovi et al. [8] propose that EstiNet has two components - the *argument extractor*, which is a differentiable module that constructs the input for the black-box function, and the black-box *estimator*, which is a differentiable function that approximates the output as well as the gradient of the black-box function. The estimator is used during training to compute the gradient of the loss function with respect to the argument extractor's parameters. In contrast, the black-box function is plugged into the system during inference. The authors demonstrate the effectiveness of EstiNet on the Image Addition task, which involves finding the sum of a sequence of digit images from the MNIST dataset. Using a CNN as the argument extractor and a combination of an LSTM network [34] and a NaLU [35] cell as a proxy for the argmax and sum function, image addition is accurately performed on image sequences of unknown length. The CNN also learned to classify the digit images without any ground truth digit labels, which shows that end-to-end training using differentiable bypass can potentially eliminate the need for procuring intermediate labels for individual components in the system. Further, differentiable proxies have been used to optimize the hyperparameters of black-box image signal processing (ISPs) units and circumvent manual configuration when deploying imaging systems for different applications [36].

2.1.2 Document Cleaning using Black-Box Approximation

Preprocessing images for OCR is essential to improve its text recognition performance. Commonly used preprocessing methods involve image binarization [37][38][39], independent component analysis [40], and deep learning-based super-resolution [41] [42]. However, these methods do not preprocess the image for the specific OCR engine being used. Tuning them to work well for different OCR engines is also cumbersome. To tackle this problem, Randika et al. [1] train a customized preprocessor by approximating the gradient of a black-box OCR engine using a neural network, which significantly improves OCR performance. The authors propose two techniques to approximate the gradient of a black-box OCR engine - NN-based approximation and SFE-based approximation. For receipt images, it is assumed that the ground truth consists of bounding box coordinates and the text present in them.

An overview of the NN-based approach is illustrated in Fig. 2.3. The NN-based approach consists of two components - the *preprocessor* g with parameters θ and the black-box *approximator* f with parameters ϕ. g and f represent EstiNet's argument extractor and black-box estimator, respectively. g is trained to perform transformations on the input document image x. A UNet architecture [43] is considered for the g, which is depicted in Figure 2.4. During training, each image is first passed

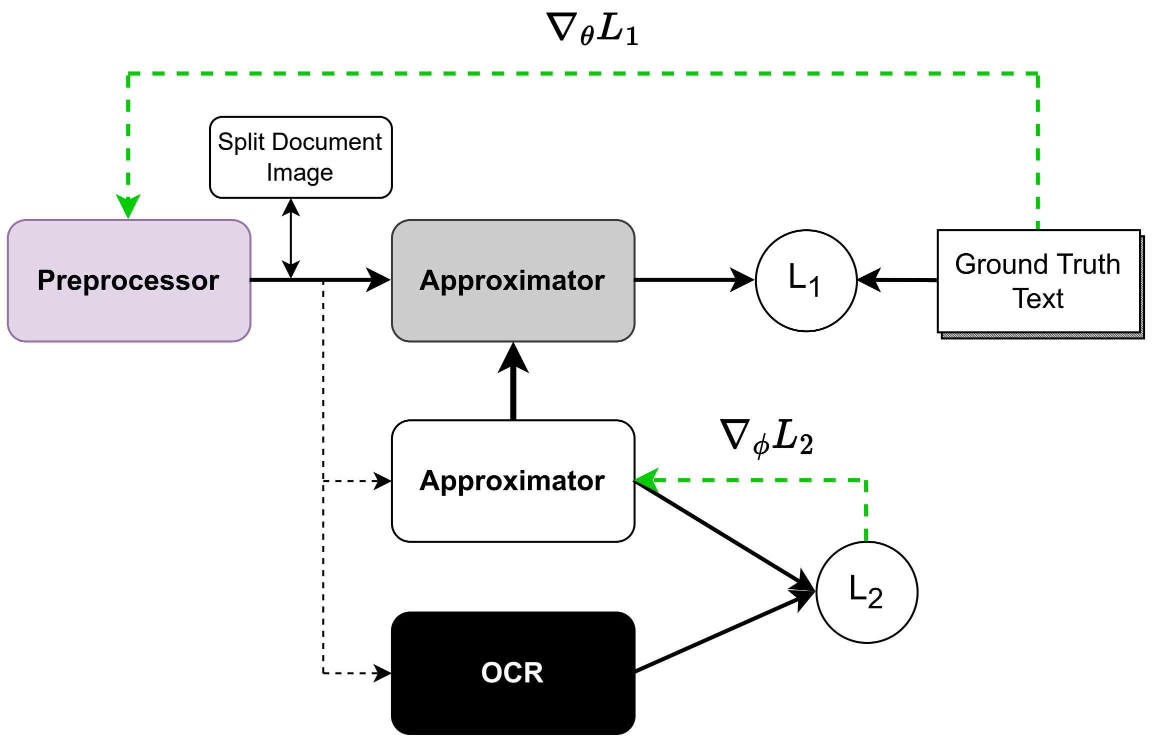


Figure 2.3: **Training pipeline**. An overview of the training pipeline [1]. The broken green lines indicate backpropagation while the broken black lines depict only forward propagation to illustrate the training of the preprocessor using differentiable bypass. Receipt and document images are split into text strips after preprocessing.

through g to obtain a preprocessed image $g(x)$, which has the same size as x. If the document image has more than one word, the words are cropped from $g(x)$ using ground-truth text bounding boxes. The cropping is performed since f is represented by the CRNN architecture [44], which can only perform text recognition on word images. It cannot extract text bounding boxes from the input image. However, it can approximate the text recognition functionality of an OCR engine. The CRNN architecture is shown in Figure 2.5. Once the word images are obtained, Gaussian noise ϵ is added to "jitter" them (2.11) before passing them through a black-box OCR engine and its differentiable approximator f. The jitter prevents overfitting in

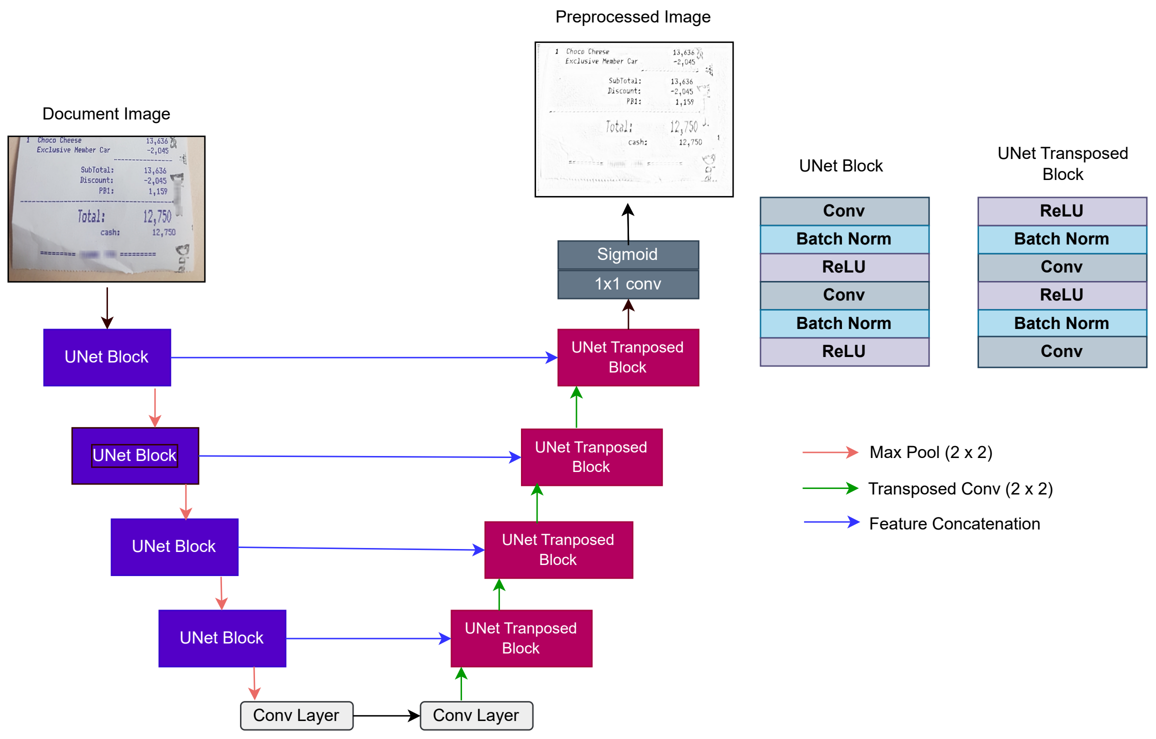


Figure 2.4: UNet architecture used by Randika et al. [1] to clean document images.

the approximator by providing some exploration in the input space. Specific hyper-parameters control the number of random perturbations and the parameters of the Gaussian noise. The output from the OCR engine is used as labels to match the output of f and update its parameters ϕ. Since the output of the OCR engine is a string, the CTC loss function [45] is used to update ϕ.

$$L_{approx}(x, y_{ocr}) = \mathrm{CTC}(f(g(x) + \epsilon), \mathrm{OCR}(g(x) + \epsilon)) \tag{2.12}$$

Subsequently, the gradient descent update rule for ϕ with learning rate η is give by

$$\phi^{t+1} = \phi^t - \eta \nabla_\phi L_{approx} \tag{2.13}$$

On the other hand, to train g, the ground truth labels are used, the parameters

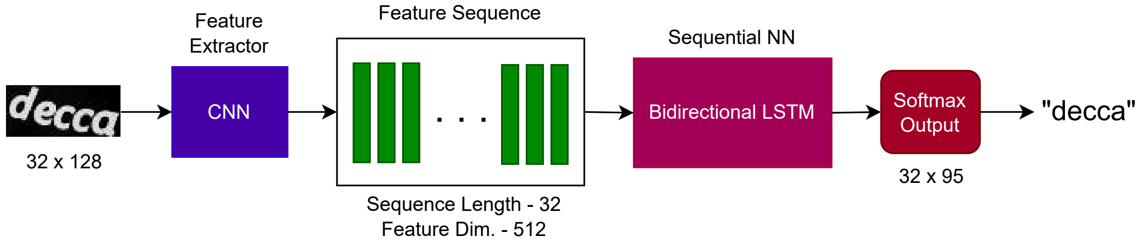


Figure 2.5: CRNN architecture used by Randika et al. [1] to approximate the gradient of black-box OCR engines.

of f, i.e., ϕ, are frozen, and f is used as a differentiable proxy for the OCR engine. Again, the CTC loss function is used to update g's parameters θ. The mean squared error (MSE) between the preprocessed image and a white image (image with all ones) is used as a secondary loss function when updating g to "whiten" the image [1]. The parameter β controls the effect of the MSE loss on the total loss. Thus, the loss function for g is given by

$$L_{prep}(x, y_{gt}) = \text{CTC}(f(g(x), y_{gt}) + \beta * \text{MSE}(g(x), J_{m\times n}) \tag{2.14}$$

The gradient descent update rule for θ with learning rate α is given by

$$\theta^{t+1} = \theta^t - \alpha\nabla_\theta L_{prep} \tag{2.15}$$

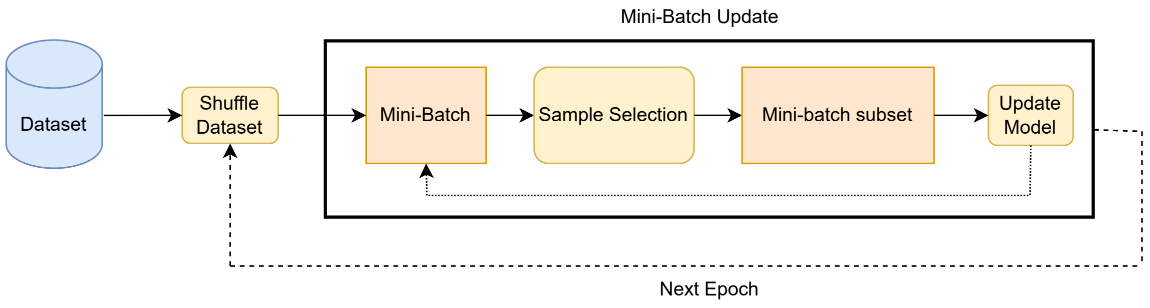
The computation of $\nabla_\theta L_{prep}$ is possible only because f acts as a differentiable proxy for the black-box OCR engine to facilitate error backpropagation. Hence, when ϕ is updated, θ is fixed, and vice versa. This alternating training scheme was proposed as *Hybrid Training* to train Estinet. Initially, f is trained separately with the output of the OCR engine to obtain good initial weights for the end-to-end system, thereby avoiding the cold-start problem. Approximating the black-box OCR engine using the differentiable approximator enables end-to-end training of the system without requiring intermediate labels to train the preprocessor.

2.2 Data Subset Selection for Efficient DNN Training

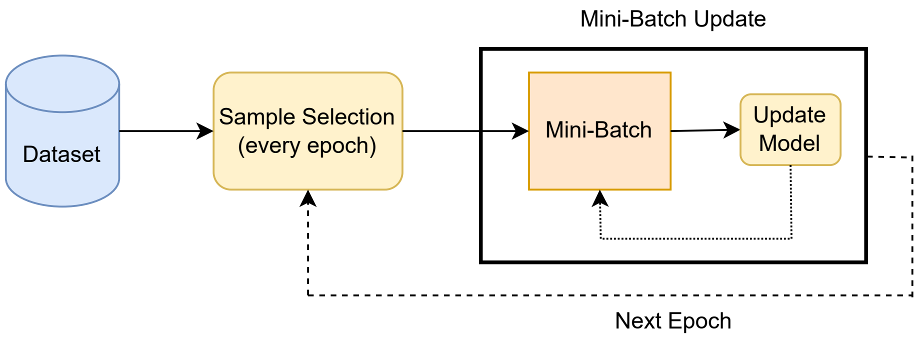
While the differentiable bypass approach is useful for training a document preprocessor to improve the performance of black-box OCR engines, it requires querying the engine for all text strips in the dataset. Due to the large computational/financial cost associated with querying commercial/open-source OCR engines, there is a need to reduce the number of OCR engine queries without compromising text recognition performance. Here, we go through recent work on efficient training of neural networks using data subset selection. The idea is to select a smaller subset of data for training a neural network such that its overall performance is the same as with the original system. We derive inspiration from these methods to select a subset of samples for querying the OCR engine and perform efficient black-box approximation to train the preprocessor. Several algorithms have been proposed to prioritize samples for accelerating the training of neural networks. Sample prioritization allows the selection of a smaller subset of samples. The sample selection can be performed in three different stages of training - mini-batch, epoch, and dataset.

2.2.1 Mini-Batch Sample Selection

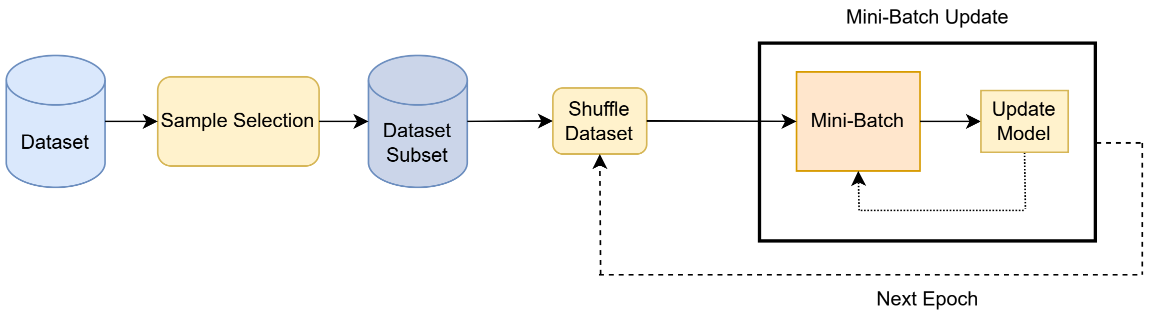
Samples are selected based on a probability value which is a function of the per-sample loss. The authors show that Selective Backprop improves training time by a factor of 3.5 compared to the standard Stochastic Gradient Descent approach. Similarly, Xu et al. [46] propose UNcertainty-aware mIXup (UNIX) for computationally-efficient Knowledge Distillation (KD) [47]. KD involves transferring the learning of a heavier (larger) model to a lighter (smaller) model. First, for each training mini-batch, UNIX performs a forward pass of the student network for the samples in the mini-batch and orders them in descending order using model uncertainty measures like entropy. Then, mixup data augmentation [48] is performed between the ordered mini-batch and the shuffled mini-batch, where the degree of mixing is controlled by



(a) Mini-Batch Sample selection



(b) Epoch Sample Selection



(c) Dataset Sample Selection

Figure 2.6: Sample Selection for accelerating neural network training.

the rank of each sample in the ordered mini-batch. More uncertain samples undergo less mixing, and vice-versa. Finally, a subset of samples is selected from the mini-batch containing mixed images, and the teacher model is queried with this mini-batch to perform KD. The authors show that UNIX reduces the computational cost of training by 30% compared to traditional Knowledge Distillation without compromising on image classification accuracy. To approximate black-box OCR engines in a query-efficient manner, we opt for sample selection within each mini-batch, i.e., selecting text strips in each CRNN mini-batch and using the smaller image subset to query the OCR engine and training the approximator.

2.2.2 Epoch Sample Selection

Sample selection at the epoch level involves selecting a subset of samples from the training dataset at the beginning of each epoch, as depicted in Figure 2.6b. Subset-selection methods based on Curriculum-learning [49] perform sample selection at each epoch. Minimax curriculum learning [50] selects hard and diverse samples using submodular maximization, and the diversity of chosen subsets is adjusted based on different stages of training. The hardness of a subset is measured using the sum of per-sample losses, and the diversity of the subset is measured with a submodular function based on image features [51]. DIHCL [52] is a curriculum learning method that uses Dynamic Instance Hardness (DIH) to quantify each sample's hardness during the training process to reduce the training time of neural networks. They propose to use the loss value, change in loss across consecutive epochs, and the number of changes in the predicted class across training as three different metrics to compute the DIH for each training sample. DIHCL selects a subset of hard samples in each epoch for training. The subset size is reduced gradually throughout training to get accurate estimates of the per-sample DIH in early epochs and subsequently allow the network to focus on more difficult samples in the later epochs. Moreover, importance sampling techniques [53] [54] have been used to select important samples for accelerating neural network training and reducing computational resource utilization.

2.2.3 Dataset Sample Selection

Sample selection from the training dataset involves selecting a representative subset of samples from the training set and using the chosen subset to train the full system instead of the original dataset, as depicted in Figure 2.6c. A coreset is a subset of the training data that can be used to train a model and achieve performance comparable to the model trained with the original data. The Coresets for Accelerating Incremental Gradient descent (CRAIG) [55] algorithm selects a weighted subset of training samples using submodular maximization of the facility location function [56] to approximately estimate the full gradient of the original training set with respect to a loss function. Selection-via-proxy [57] uses a low-capacity proxy model to perform coreset selection from a given training dataset in a computationally efficient manner for accelerating the training of a larger capacity model. The coreset selection can be performed using forgetting events [58], max-entropy uncertainty sampling [59] or greedy k-means [60] [61].

Active Learning. Beyond supervised learning, sample selection techniques are used to prioritize unlabelled data for labeling in an active learning setting. Uncertainty sampling techniques like Max Entropy and Least Confidence Sampling are used to select the most uncertain samples for annotation [59]. Further, filtered active submodular selection (FASS) [62] is a batch active learning algorithm that selects diverse samples from the set of most uncertain samples for a given model.

2.3 Tracking Past Information

There are several domains where past information accumulated by the model has been used to improve the model's performance. Experience Replay [25] is a technique that enables training a reinforcement learning agent using previous experience stored in memory, which allows the agent to learn with less real-world experience. Experience Replay has also been used to tackle catastrophic forgetting in multi-task reinforcement learning [26] and graph neural networks [63]. Moreover, using soft labels as history information to predict word-level sentiment has been shown to improve the performance of target-level sentiment classification [64]. Based on these methods, we utilize past predictions of the OCR engine to train the approximator and improve the OCR engine's text recognition performance in a low query setting.

Chapter 3

Methods

3.1 Selecting Samples to Query OCR Engine

Querying the OCR engine for each text strip can be computationally expensive (for open-source engines) or costs money (commercial software/APIs). In our setup, the preprocessed word images are passed as a mini-batch B to f and the OCR engine. The goal of subset selection is to choose a representative subset of size k from each mini-batch such that $k \ll |B|$ and obtain the OCR labels for them to train f. Querying the OCR engine for k samples instead of $|B|$ leads to a significant reduction in total OCR queries. Our approach for sample selection involves choosing a representative subset [50][62] [65] based on a measurable property of each sample. In this regard, we propose two techniques for selecting samples to obtain OCR labels - UniformCER and TopKCER. Fig. 3.1 and Fig. 3.2 provide an overview of sample selection.

3.1.1 UniformCER

Character Error Rate (CER) is a character-level metric used for evaluating OCR systems. We assume that CER, with respect to the ground truth label, quantifies the

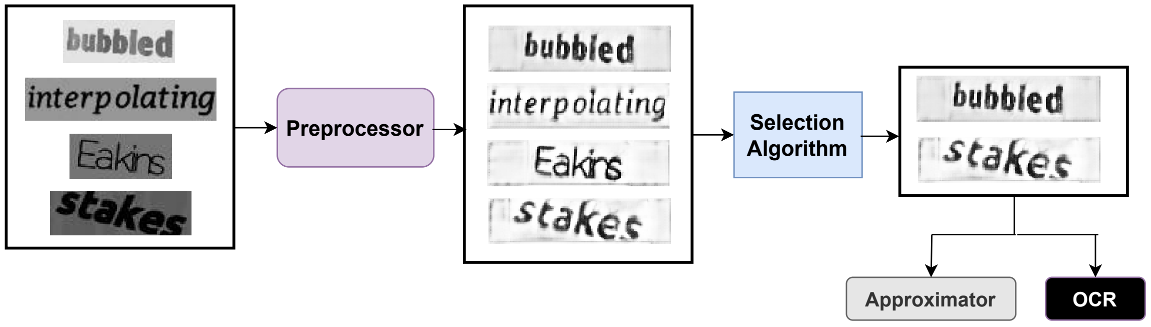


Figure 3.1: Sample selection for the VGG dataset using a selection algorithm after images are passed through the preprocessor.

Algorithm 1 Efficient approximation of OCR engine for document image cleaning using sample selection

```
**Input**: $X_{train}$, $Y_{train}$, $k$, $\eta_1$, $\eta_2$, $\sigma$
**for** $x_{batch}, y_{batch} \in \{X_{train}, Y_{train}\}$ **do**
    $g_{batch} = preprocessor(x_{batch})$
    $subset = select(g_{batch}, k)$
    **for** $x \in subset$ **do**
        Sample $\epsilon \sim \mathcal{N}(0, \sigma)$
        $L$ += $\text{CTC}(f(x + \epsilon),\ OCR(x + \epsilon))$
    $\phi = \phi - \eta_1 \nabla_\phi L$
    $\theta = \theta - \eta_2 \nabla_\theta L(f(x_{batch}), y_{batch})$
```

hardness of each sample. The CER for a sample is calculated as shown in equation 3.1, where n is the number of characters in the ground truth word, s is the number of substitutions, i is the number of insertions, and d is the number of deletions.

$$\text{CER} = \frac{(s + i + d)}{n} \tag{3.1}$$

As shown in Algorithm 1, the parameters of preprocessor g are updated by calculating the loss function between f and the OCR for *all* samples in mini-batch B, which enables the computation of CER value for each sample in B with respect to the ground truth labels. These CER values are recorded in each training epoch. The

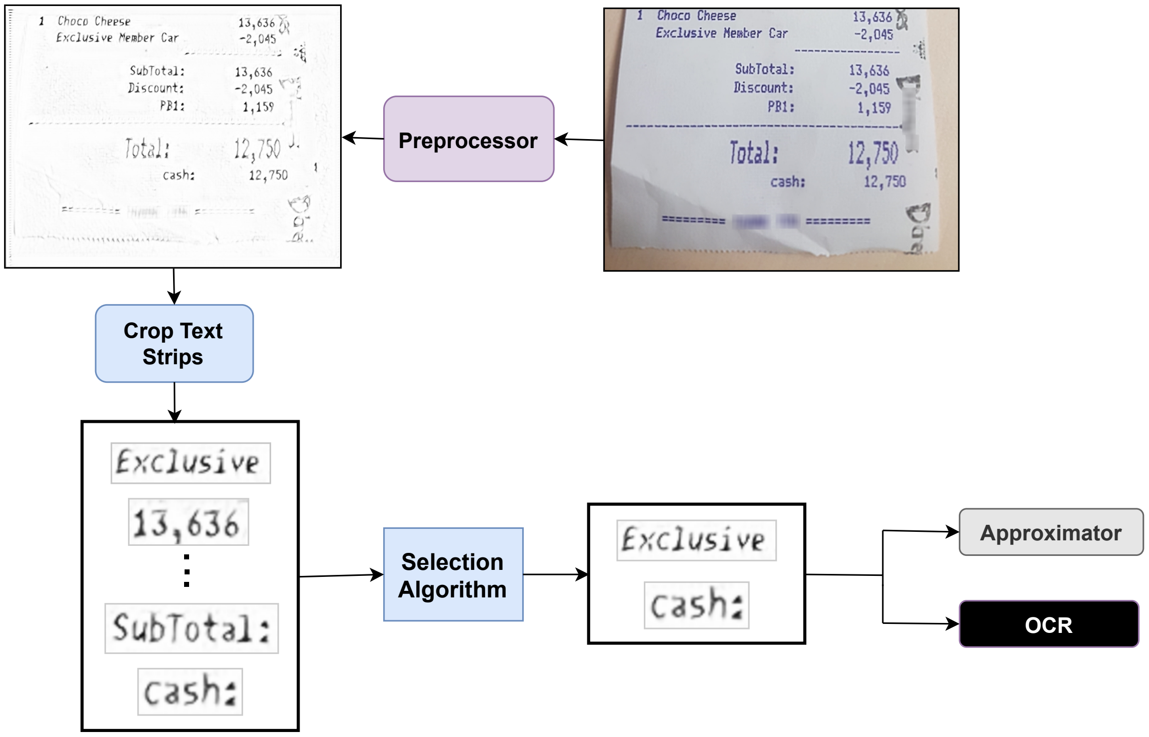


Figure 3.2: Samples selection for the POS dataset using a selection algorithm after dcoument images are passed through the preprocessor.

CERs stored in the previous epoch for each sample in B are used for sample selection to query the OCR. We obtain the minimum and maximum CER value for each mini-batch, denoted by cer_{min} and cer_{max}, respectively. Then, k values are sampled from Uniform(cer_{min}, cer_{max}). Finally, we determine the sample whose CER is closest to each of the k selected points. These k samples are passed through the OCR and the CRNN to compute the CTC loss function and update the weights of the CRNN. Algorithm 2 shows the mathematical details of UniformCER.

Algorithm 2 UniformCER Selection Algorithm

```
**Input**: $x_{batch}$, $cers_{batch}$, $k$
$cer_{max} = \max(cers_{batch})$
$cer_{min} = \min(cers_{batch})$
$c_1, c_2 \dots c_k \sim U(cer_{min}, cer_{max})$
$idx = \{\}$
**for** $c_i \in c_1, \dots c_k$ **do**
    $j = \underset{j \notin idx}{\operatorname{argmin}} (|c_i - (x_{batch})_j|)$
    $idx.\text{insert}(j)$
$subset = x_{batch}[idx]$
```

3.1.2 TopKCER

Selective Backprop and Variance Reduction Importance Sampling (VR) [54] perform loss-based sample prioritization to select a subset of samples for neural-network training acceleration. These techniques assume that samples with a higher loss are highly informative. Since CER can also be treated as a measure of informativeness because it measures the hardness of text strips, we can select samples with the highest CERs computed during training of the system (as shown in 3.1.1) to query the OCR engine.

3.2 Pruning Document Images

In section 3.1, we discussed techniques to select images from the training dataset to construct a coreset. Similarly, we propose a simple technique to prune the document image dataset and train the preprocessor using a smaller subset of documents. In our document cleaning setup, the approximator is pre-trained with the output of the OCR engine to avoid the cold start problem. The OCR engine is queried once for all text strips/word images in the dataset to obtain labels for pre-training. We use these OCR string labels to compute the CER of each text strip with respect to the Ground Truth labels. This CER quantifies the OCR engine's capability to predict the correct characters in each text strip. A text strip with a high CER can be considered a *hard* sample for the OCR engine, while a text strip with a low CER can be considered an *easy* sample. Then, we compute the *mean CER* for each document image using the CER of the text strips present in the image. The mean CER gives us a ranking of the images based on the OCR engine's CER. Images are selected by choosing the top-k samples based on mean CER. Performing sample selection from the dataset enables us to remove document images before training the system. The techniques proposed in section 3.1 can be combined with our pruning algorithm to reduce queries to the OCR engine further. Figure 3.3 provides an overview of the data pruning setup.

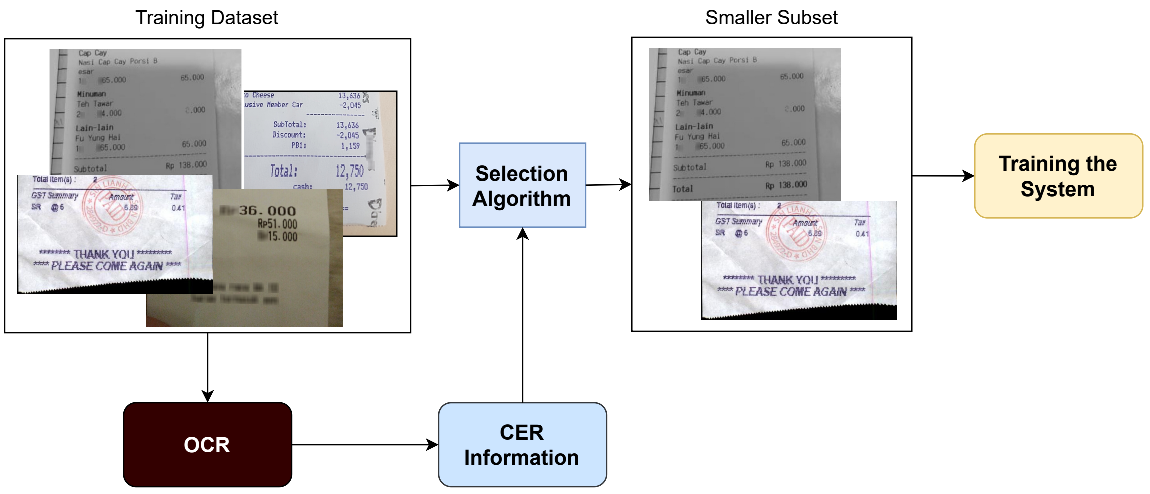


Figure 3.3: Samples selection for the POS dataset using a selection algorithm before training the preprocessor using differentiable bypass.

3.3 Label Tracking for Utilizing History Information

Label tracking aims to leverage the history of the OCR engine's predictions for individual samples during training. Figure 3.4 shows an overview of label tracking. In Algorithm 1, we show that a *subset* of preprocessed text strips is selected to query the OCR engine and subsequently update the approximator's parameters. Due to low query budgets, the size of the chosen subset is very small. Hence, we would like to utilize the predictions of all samples throughout training. For each sample x in the *subset* (in our setup, x is a text strip), the output from the OCR engine is appended to its list of predictions, denoted by $H(x)$. Here is an example of the history of OCR predictions for a given sample -

$$\mathrm{H(x)} = [\,\mathrm{stok}, \mathrm{stokee}, \mathrm{Stroke}, \mathrm{stroke}\,] \tag{3.2}$$

The i'th label from the back of the list $H(x)$ can be indexed as $H(x)_i$. For instance, $H(x)_1$ represents the most recent prediction obtained by querying the OCR engine for sample x. In our example, $H(x)_1$ is "stroke", $H(x)_2$ is "Stroke" and so on.

Algorithm 3 Efficient OCR approximation for document image cleaning using sample selection with and w/o label tracking

```
Input: $X_{train}$, $Y_{train}$, $\boldsymbol{w}$, *history* (or $H$), $k$, $m$, $\eta_1$, $\eta_2$, *doTracking*, $\sigma$
for $x_{batch}, y_{batch} \in \{X_{train}, Y_{train}\}$ do
    $g_{batch} = preprocessor(x_{batch})$
    $subset = select(g_{batch}, k)$
    if doTracking then
        $L' = 0$
        for $x \in subset$ do
            Add $OCR$(x) to $H(x)$
            for $i$ in $1..m$ do
                $L_i(\mathrm{x}) = \mathrm{CTC}(f(\mathrm{x}), H(x)_i)$
                $L' \mathrel{+}= w_i * L_i(x)$
    else
        Sample $\epsilon \sim \mathcal{N}(0, \sigma)$
        $L' = \mathrm{CTC}(f(subset + \epsilon), OCR(subset + \epsilon))$
    $\phi = \phi - \eta_1 \nabla_\phi L'$
    $\theta = \theta - \eta_2 \nabla_\theta L(f(x_{batch}), y_{batch})$
```

In the first epoch, the list of past predictions is empty for all samples. More history information is added as the training progresses. As shown in the previous example, the output of the OCR engine can change for a sample throughout training

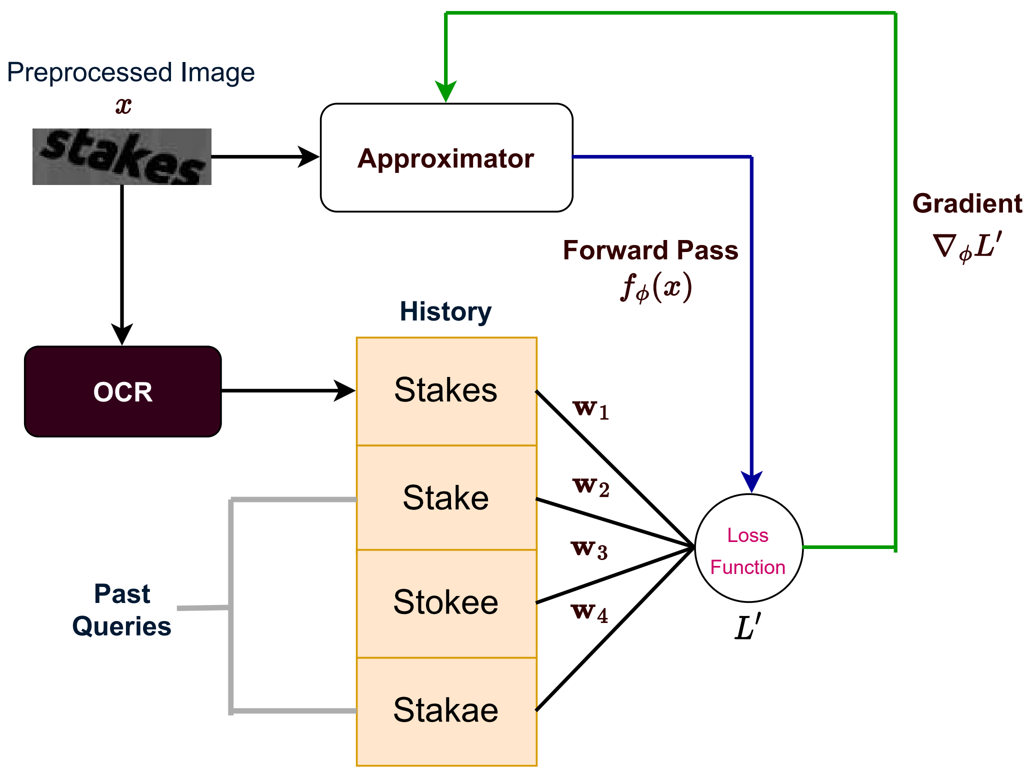


Figure 3.4: Accumulating history information for training samples and utilizing the past predictions to compute the loss function for updating the approximator using label tracking.

since the preprocessor's parameters are being updated using differentiable bypass. After a few epochs of training, the history of OCR predictions can look like this for 3 samples -

$$\mathrm{H}(x_1) = [\,abc, ab\,]$$
$$\mathrm{H}(x_2) = [\,stok, stokee, Stroke, stroke\,]$$
$$\mathrm{H}(x_3) = [\,]$$

Each sample's prediction history can be used to train the approximator, as described in algorithm 3. When the OCR engine is queried for a sample x, we check if $H(x)$

is non-empty. If it is, we pick the m most recent predicted labels for that sample, where m is the *window size*. For each of these labels, we assign corresponding weights $w_1, ..., w_m$ and these weights are set to be decaying from most recent to least recent so that older labels have smaller weights associated with them. A general form for the weights is given by,

$$w_{i+1} = \lambda w_i, \text{ s.t. } \lambda \in (0, 1]$$

We use the CTC loss function to compute the loss between the approximator f and the i'th most recent OCR prediction $H(x)_i$,

$$L_i(x) = \text{CTC}(f(x), H(x)_i) \tag{3.3}$$

To calculate the loss using all labels in history within window size m, we modify our loss function to be a multi-loss objective and use $w_1, .., w_m$ to control the impact of each label on the total loss. Hence, the new loss function L' is

$$L' = \sum_{i=1}^{m} w_i * L_i \tag{3.4}$$

As an example, we consider $m = 3$ and the label history in example 3.2 to enumerate the individual components of the summation term in equation 3.4

$$\begin{aligned} L'(x) = {} & w_1 * \text{CTC}(f(x), \text{"}stroke\text{"}) + w_2 * \text{CTC}(f(x), \text{"}Stroke\text{"}) \\ & + w_3 * \text{CTC}(f(x), \text{"}stokee\text{"}) \end{aligned}$$

Contrary to Algorithm 1, we do not add noise to the sample before querying the OCR engine in label tracking. We refrain from adding noise since the OCR engine labels obtained for noisy samples can be an unreliable source of information for label tracking. We want to track reliable labels associated with a sample so that changes in the OCR engine's predictions for a sample are caused only by the training of the preprocessor.

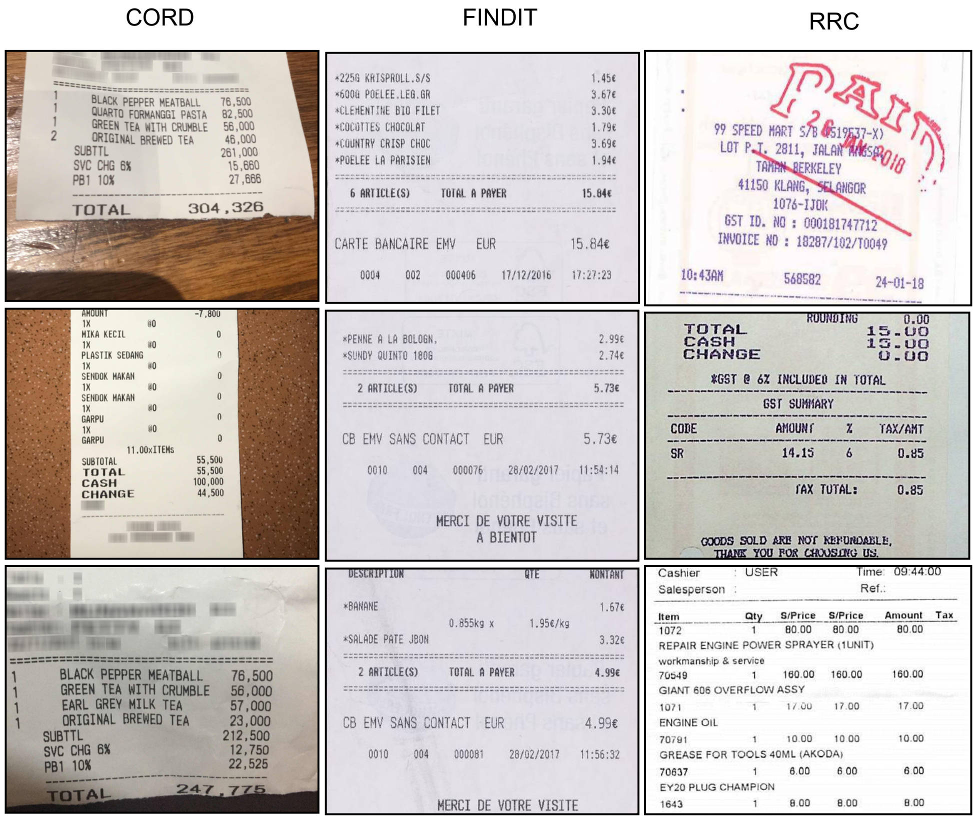
Chapter 4

Experiments and Results

4.1 Experiment Setup

4.1.1 Datasets

All experiments are performed on the "POS dataset" and the VGG dataset, curated as shown in [1], with Tesseract and EasyOCR as the black-box OCR engines. The POS dataset is a combination of three POS (Point-of-Sale receipt) datasets - Findit fraud detection dataset [66], ICDAR SROIE competition dataset [67] and CORD dataset [68]. Image patches are extracted from the receipt images and then resized such that the images in the dataset have a maximum height of 400 pixels and a width of 500 pixels. With roughly 90k word patches, the POS dataset has 3676 training images, 424 validation images, and 417 test images. The VGG dataset consists of 60k word images randomly sampled from the VGG synthetic word dataset [69]. The dataset is split into 50k training images, 5k validation images, and 5k test images. A few sample images from both datasets are shown in Figures 4.1 and 4.2.


CORD
FINDIT
RRC

```
[redacted]
================================
1    BLACK PEPPER MEATBALL    76,500
1    QUARTO FORMANGGI PASTA   82,500
1    GREEN TEA WITH CRUMBLE   56,000
2    ORIGINAL BREWED TEA      46,000
  SUBTTL                     261,000
  SVC CHG 6%                  15,660
  PB1 10%                     27,666
  --------------------------------
  TOTAL                   304,326
```

```
AMOUNT                -7,800
1X          @0
MIKA KECIL                 0
1X          @0
PLASTIK SEDANG             0
1X          @0
SENDOK MAKAN               0
1X          @0
SENDOK MAKAN               0
1X          @0
GARPU                      0
1X          @0
GARPU                      0
        11.00xITEMs
SUBTOTAL              55,500
TOTAL                 55,500
CASH                 100,000
CHANGE                44,500
[redacted]
------------------------------
[redacted]
```

```
[redacted]
=====================================
1      BLACK PEPPER MEATBALL    76,500
1      GREEN TEA WITH CRUMBLE   56,000
1      EARL GREY MILK TEA       57,000
1      ORIGINAL BREWED TEA      23,000
  SUBTTL                       212,500
  SVC CHG 6%                    12,750
  PB1 10%                       22,525
  ----------------------------------
  TOTAL                    247,775
```

```
*225G KRISPROLL.S/S                      1.45€
*600G POELEE.LEG.GR                      3.67€
*CLEMENTINE BIO FILET                    3.30€
*COCOTTES CHOCOLAT                       1.79€
*COUNTRY CRISP CHOC                      3.69€
*POELEE LA PARISIEN                      1.94€
================================================
  6 ARTICLE(S)      TOTAL A PAYER       15.84€
================================================

CARTE BANCAIRE EMV    EUR              15.84€

   0004     002     000406    17/12/2016    17:27:23
```

```
*PENNE A LA BOLOGN,                      2.99€
*SUNDY QUINTO 180G                       2.74€
================================================
  2 ARTICLE(S)      TOTAL A PAYER        5.73€
================================================

CB EMV SANS CONTACT   EUR               5.73€

   0010     004     000076    28/02/2017    11:54:14

           MERCI DE VOTRE VISITE
                A BIENTOT
```

```
DESCRIPTION                    QTE       MONTANT

*BANANE                                   1.67€
              0.855kg x      1.95€/kg
*SALADE PATE JBON                         3.32€
================================================
  2 ARTICLE(S)      TOTAL A PAYER        4.99€
================================================

CB EMV SANS CONTACT   EUR               4.99€

   0010     004     000081    28/02/2017    11:56:32

           MERCI DE VOTRE VISITE
```

```
                                 PAID
                                 26 JAN 2018
   99 SPEED MART S/B (519537-X)
   LOT P.T. 2811, JALAN ANGSA
        TAMAN BERKELEY
     41150 KLANG, SELANGOR
           1076-IJOK
   GST ID. NO : 000181747712
   INVOICE NO : 18287/102/T0049

10:43AM          568582           24-01-18
```

```
                       ROUNDING        0.00
TOTAL                                 15.00
CASH                                  15.00
CHANGE                                 0.00
        *GST @ 6% INCLUDED IN TOTAL
--------------------------------------------
               GST SUMMARY
--------------------------------------------
CODE          AMOUNT      %       TAX/AMT
--------------------------------------------
SR             14.15      6         0.85
--------------------------------------------
               TAX TOTAL:           0.85

      GOODS SOLD ARE NOT REFUNDABLE,
        THANK YOU FOR CHOOSING US.
```

```
Cashier      : USER                  Time:  09:44:00
Salesperson  :                       Ref.:
-----------------------------------------------------
Item            Qty   S/Price   S/Price   Amount   Tax
1072             1     80.00     80.00     80.00
REPAIR ENGINE POWER SPRAYER (1UNIT)
workmanship & service
70549            1    160.00    160.00    160.00
GIANT 606 OVERFLOW ASSY
1071             1     17.00     17.00     17.00
ENGINE OIL
70791            1     10.00     10.00     10.00
GREASE FOR TOOLS 40ML (AKODA)
70637            1      6.00      6.00      6.00
EY20 PLUG CHAMPION
1643             1      8.00      8.00      8.00
```

Figure 4.1: Sample Images from the POS Dataset.

4.1.2 Training Details

Similar to [1], a UNet architecture is considered for the preprocessor, while a CRNN architecture is used for the approximator. The system is trained using the Adam optimizer [70] for 50 epochs, with the learning rates for the preprocessor and approximator being 5×10^{-5} and 10^{-4} respectively. A weight decay of 5×10^{-4} is used for both the approximator and preprocessor when the POS dataset is used for training. The CRNN model is pre-trained with the OCR for 50 epochs to avoid the cold start problem [1]. For noise jitter, σ is randomly sampled from $0, 0.01, 0.02, 0.03, 0.04$, and 0.05. Most of these hyperparameter values are picked up from the best values used to train the original system [1]. The full receipt images are first passed through the

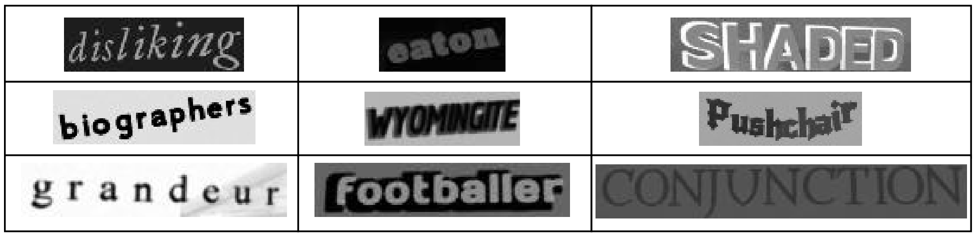
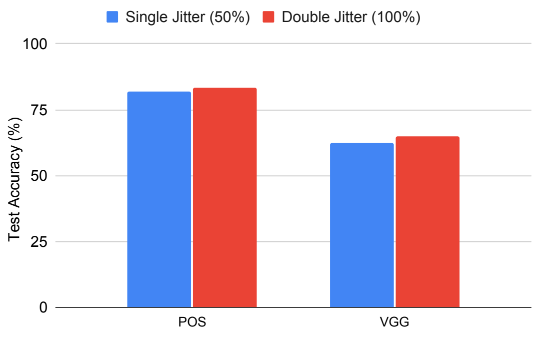


Figure 4.2: Sample Images from the VGG Dataset.

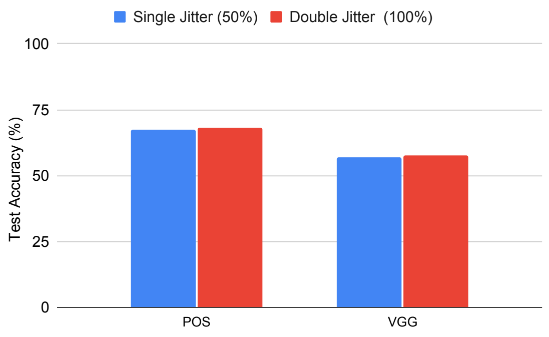
preprocessor when training the system with the POS dataset. Then, each preprocessed image is split into word images to pass them through the approximator and to query the OCR engine since our CRNN model can only work with word images as inputs. Hence, for receipt images, the batch size for the preprocessor is 1, and the batch size for the approximator is the number of text strips in the preprocessed document image. For the VGG dataset, a batch size of 64 is used for the UNet and the CRNN. For the MSE loss, We use $\beta = 1$ for our experiments [1].

The preprocessor's weights are updated by calculating the loss with respect to the ground truth labels for all samples in a mini-batch (Fig. 2.3). To train the approximator, a small subset of samples is selected from the preprocessed images (or the text strips extracted from a preprocessed image). Sample selection is performed before querying the OCR engine such that the number of queries in each epoch is $n\%$ of the queries in the original system per epoch. In the original setup, Gaussian noise is added twice for each sample, resulting in two queries per sample in each epoch. Hence, if $n = 10$, we select 20% of the total text strips in each minibatch to query the OCR and update CRNN's weights (10% of (2 |text strips|) = 20% of |text strips|). In our work, we consider $n \in \{4, 8\}$ to demonstrate the efficacy of our approach on low query budgets. There are no additional hyperparameters associated with the proposed selection methods. We consider random sampling as a baseline method for our experiments. For UniformCER and TopKCER, we compute the CER of Tesseract for all text strips before training the system and use it for performing selection in the first epoch.

Query Budgets and Preprocessor Evaluation. We refer to the results pre-



(a) Tesseract



(b) EasyOCR

Figure 4.3: Test set word-level accuracy (%) of OCR engine with noise added once (50% budget) and twice (100% budget) for POS and VGG datasets.

sented in [1] for 100% budget, while the results for 50% budget are obtained by adding Gaussian noise to the samples only once. In this work, we jitter the samples only once since it allows us to reduce the number of queries by 50% with a minimal drop in accuracy (Fig. 4.3). The OCR is not queried for experiments with a 0% budget, and the weights of the pre-trained CRNN model are frozen throughout training. Our system's evaluation metric is the OCR engine's word-level accuracy with the pre-processed images. It calculates the proportion of recognized words that correctly match the ground truth text. The word accuracy is reported using the pre-processor checkpoint with the highest validation accuracy. All models were trained on an NVIDIA V100 GPU.

Hyperparameter Tuning. Since our aim is to reduce the number of queries to the OCR engine, we do not perform extensive hyperparameter sweeps for all settings. In most cases, the hyperparameters from [1] were sufficient for the system to achieve good performance. However, for EasyOCR and POS dataset, we obtained sub-par performance with the default hyperparameters. Hence, we tuned the learning rate of UNet and CRNN for 4% query budget and TopKCER selection. Using the best-performing hyperparameters (learning rate of UNet and CRNN are 5×10^{-4} and 1.5×10^{-5} respectively), we performed the rest of the experiments. The results for EasyOCR on VGG dataset are reported with these hyperparameters. More details regarding hyperparameter tuning can be found in the appendix (A.3).

Google Vision API. Since it costs money to query the Google Vision API[1], we only perform experiments with the POS dataset, UniformCER selection, and 2.5% budget. 2.5% is the minimum budget possible in this setup since we observed that the OCR needs to be queried *for at least one sample* in each minibatch to achieve good performance on low query budgets. Further, querying the OCR for all samples in the validation set in each epoch incurs a high cost, so we randomly sample 50 images (∼800 text strips) from the validation set and use the word-level accuracy on this image subset to choose the best model checkpoint. Finally, we train this system for 41 epochs.

Dataset	**Budget**	**Selection Method**		
		Random	UniformCER	TopKCER
POS	4%	78.36	80.73	**81.60**
	8%	79.20	81.34	**81.44**
	0%		75.23	
	100%		83.36	
VGG	4%	62.04	62.86	**63.50**
	8%	62.16	**63.86**	63.56
	0%		45.60	
	100%		64.94	

Table 4.1: Word-level accuracy (%) with Tesseract for different selection methods. No selection methods were used for 0% and 100% budgets. The OCR engine is never queried for 0% budget.

[1]https://cloud.google.com/vision/pricing

4.2 Results

Tables 4.1 and 4.2 depict the word-level accuracy of different selection algorithms on Tesseract and EasyOCR, respectively, across two query budgets for both datasets. Figures 4.4 and 4.5 show the preprocessed images obtained after training the preprocessor with different query budgets.

Dataset	**Budget**	**Selection Method**		
		Random	UniformCER	TopKCER
POS	4%	62.24	**65.56**	64.22
	8%	65.27	65.02	**65.86**
	0%		59.10	
	100%		67.97	
VGG	4%	56.00	56.12	**56.84**
	8%	56.62	57.80	**57.82**
	0%		49.00	
	100%		57.48	

Table 4.2: Word-level accuracy (%) with EasyOCR for different selection methods. No selection methods were used for 0% and 100% budgets. The OCR engine is never queried for 0% budget.

Importance of querying the OCR. For most settings, it is evident that with a budget of only 4%, the performance of all selection algorithms is better than the performance of the system with no OCR engine queries (0% query budget). Further, selection using random sampling alone leads to improvement in OCR performance with less than 10% of the query budget, which shows that random sampling is a strong baseline. These results also indicate that updating the approximator with at least a few queries to the OCR engine is essential to train a better preprocessor. Figure A.1 in the Appendix shows the visual difference in preprocessed images between 0%

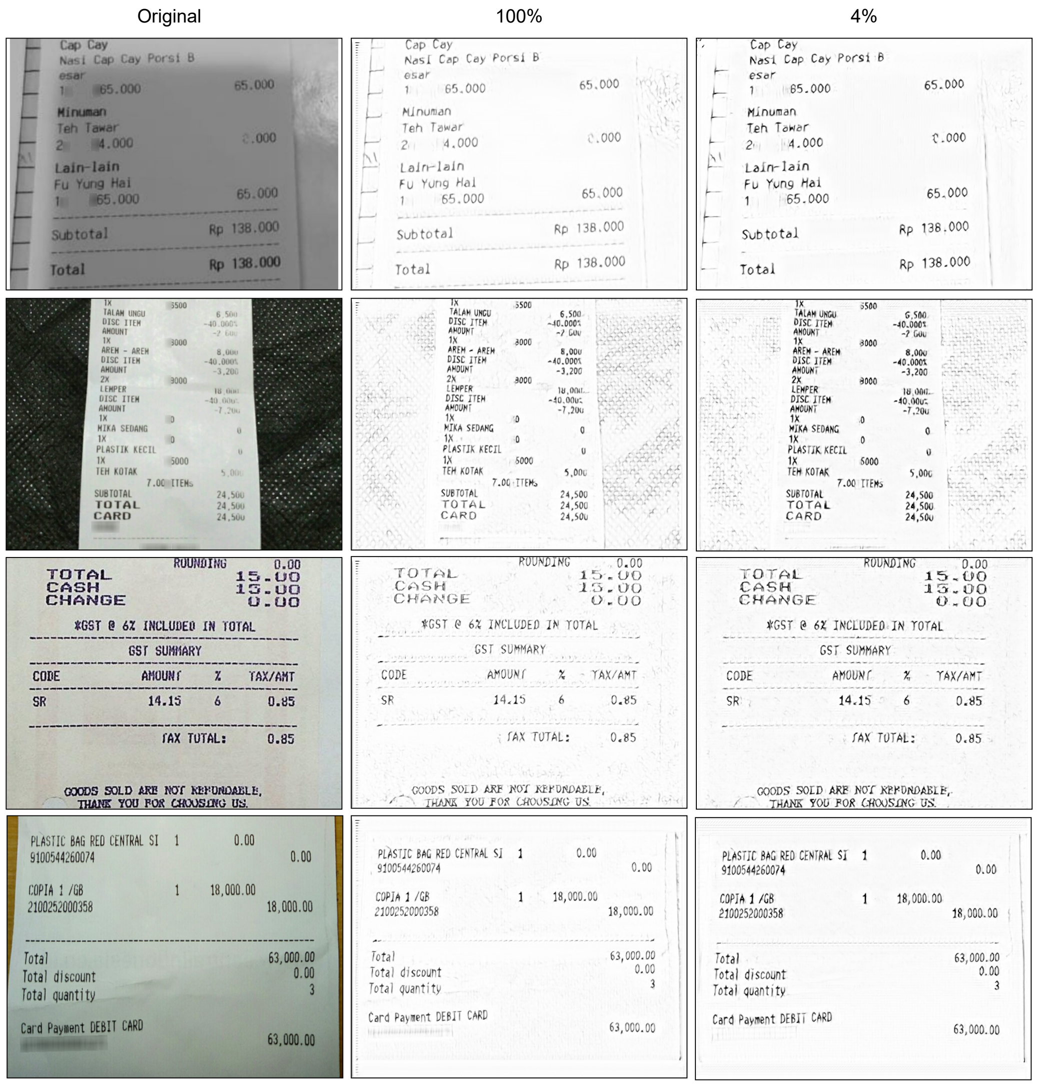


Figure 4.4: Preprocessor output for different test samples in the POS dataset. The preprocessors were trained with Tesseract OCR engine and UniformCER selection was used for 4% % budget. **Column 1:** Original Images. **Column 2:** Original system - 100% query budget. **Column 3:** 4% query budget.

Dataset	Without Preprocessing	With Preprocessing Budget=2.5%	Projected Expense Budget=100%	Actual Expense Budget=2.5%
POS	82.57%	86.69%	9030 USD	280 USD

Table 4.3: Test set word-level accuracy (%) of preprocessor trained with Google Vision API along with expected and actual cost of training the system.

budget and 2.5% to illustrate the improvement in preprocessing with minimal OCR engine queries.

Performance of selection algorithms. Across the board, both UniformCER and TopKCER perform better than random sampling. This is particularly evident for the 8% budget setting with the POS dataset for both Tesseract and EasyOCR. Compared to the original system with 100% query budget, UniformCER and TopKCER with 8% query budget results in a 3% or lower drop in accuracy for both OCR engines. For the VGG dataset, all selection algorithms display a minimum improvement of 7% when compared to the performance at 0% budget. For this dataset, all selection algorithms have 2% or lower drop in accuracy for both 4% and 8% budgets, with both CER-based algorithms performing better than random sampling. From these results, we can conclude that for lower budgets, selection using a sample measure like CER is necessary to achieve 1-2% improvement over random sampling. Fig. 4.4 and Fig. 4.5 show that the preprocessed images using 100% and 4% query look very similar for both POS and VGG datasets. This indicates that the transformations learnt by the UNet are similar across these budgets and we can perform document cleaning for a black-box OCR engine with very low number of queries to it.

Google Vision API Results. Table 4.3 shows the result for training a preprocessor with the Google Vision API for the POS dataset. We observe that there is 4% increase in the word-level accuracy with just 2.5% query budget. The increase in accuracy with a low budget shows that the performance of Google Vision API can

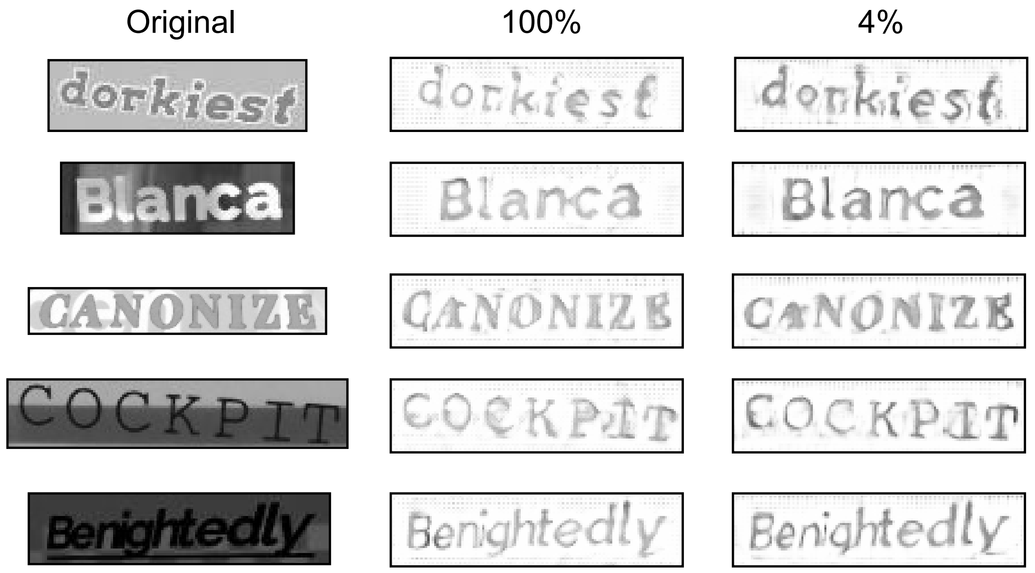


Figure 4.5: Preprocessor output for different test samples in the VGG dataset. The preprocessors were trained with Tesseract OCR engine and UniformCER selection was used for 4% % budget. **Column 1:** Original Images. **Column 2:** Original system - 100% query budget. **Column 3:** 4% query budget.

be improved by training a preprocessor using differentiable bypass. We also observe that training the preprocessor using the original system would require **32x more cost** than training it with 2.5% budget, which is a significant reduction in cost.

4.3 Additional Experiments

4.3.1 Training and Testing on Different OCR Engines

Table 4.4 shows the results for training and testing the preprocessor on different OCR engines. The results for 100% budget have been inferred from a similar experiment conducted by [1]. We observe that in most cases, the difference in performance between 4% and 100% query budget is less than 6% across all OCR combinations. These results demonstrate that a preprocessor trained with a low query budget per-

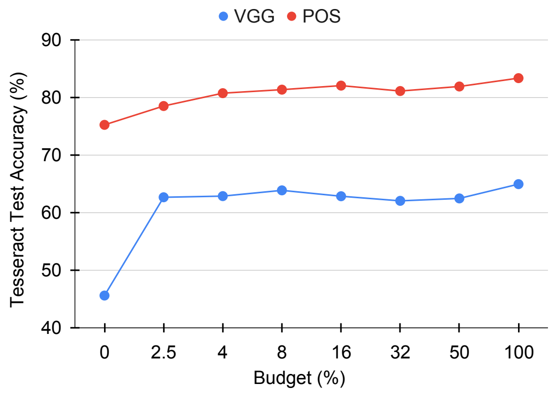
Dataset	**OCR used for training**	**OCR used for testing**	**Test Accuracy (4% Budget)**	**Test Accuracy (100% Budget)**
POS	Tesseract	EasyOCR	39.06	40.44
VGG	Tesseract	EasyOCR	43.08	47.14
POS	EasyOCR	Tesseract	63.92	60.94
VGG	EasyOCR	Tesseract	43.86	21.64
POS	Google Vision	Tesseract	66.45	-
POS	Google Vision	EasyOCR	33.04	-
POS	Tesseract	Google Vision	-	88.23*

Table 4.4: Word-level accuracy (%) for OCR trained and tested on different engines using best-performing preprocessors from Tables 4.1,4.2 and 4.3. *The preprocessor was trained with 50% query budget.

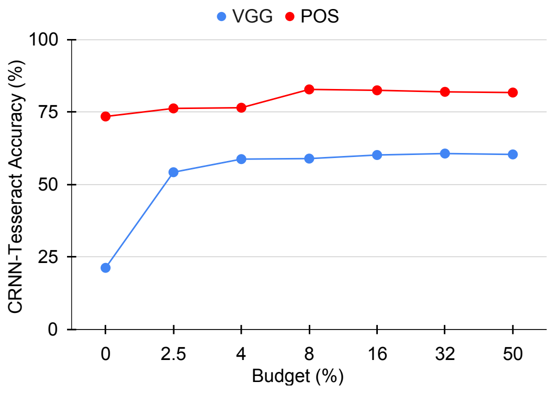
forms similarly to the original system when tested on different OCR engines. Moreover, the last row shows that a preprocessor trained with Tesseract using a 50% budget and evaluated on Google Vision API results in an accuracy improvement of 6%. Furthermore, it is important to note that the huge increase in performance (by 20%) observed in the fourth row of the table results from pre-training the CRNN with Tesseract instead of easyOCR for 4% query budget on the VGG dataset. We observed that this change improves the performance of EasyOCR on a low query budget while also improving the performance on Tesseract for the VGG dataset.

4.3.2 Performance across Different Query Budgets

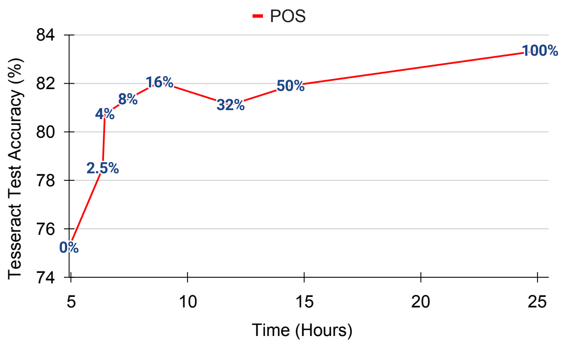
Fig. 4.6a depicts the performance of UniformCER with Tesseract across different query budgets on the POS dataset. We observe a clear jump in accuracy for both datasets when moving from 0% to 2.5% budget. However, the increase in accuracy plateaus as the budget increases beyond 4%, indicating the diminishing return of higher query budgets on preprocessor performance.



(a) Test Accuracy



(b) CRNN Accuracy

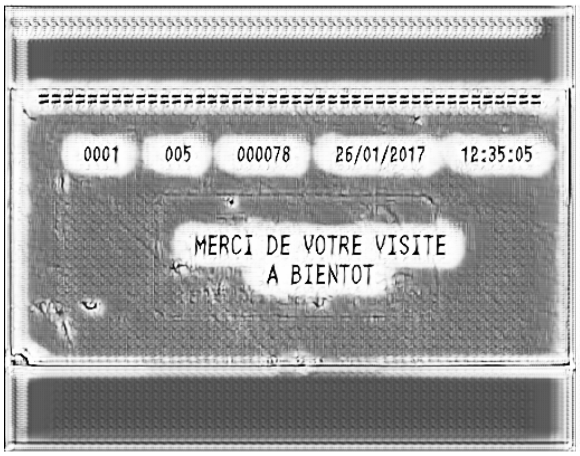


(c) Training Time

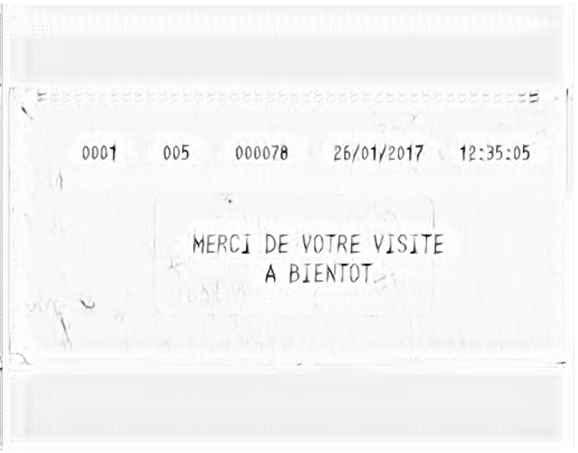
Figure 4.6: (a) Test set performance, (b) CRNN accuracy with OCR predictions as Ground Truth and (c) system training time, with UniformCER selection and Tesseract across different query budgets.

With the same experiment setup, we also aim to quantify the *approximation strength* of CRNN with respect to the OCR engine. We determine the approximation strength by computing the accuracy of CRNN on the validation set with respect to the OCR predictions. Fig. 4.6b shows the *CRNN-Tesseract* accuracy across different query budgets. We observe that the trend is similar to that of Fig. 4.6a, which shows that the CRNN's approximation strength (with respect to Tesseract and the Patch dataset) improves with a low query budget like 2.5%, which could potentially explain the increase in performance from 0-2.5% in Fig. 4.6a.

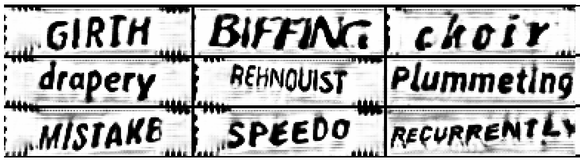
Fig. 4.6c shows the training time and accuracy of the system across different budgets for the POS dataset. We notice that the training time increases significantly with increase in query budget. However, these higher budgets do not lead to a significant improvement in OCR performance, indicating that we can improve the OCR performance by querying the OCR with a very low query budget.



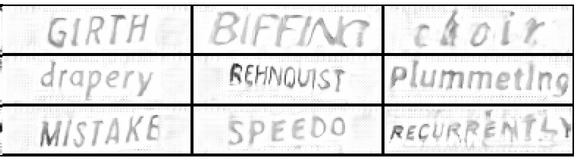
(a) POS, $\beta = 0$



(b) POS, $\beta = 1$



(c) VGG, $\beta = 0$



(d) VGG, $\beta = 1$

Figure 4.7: Effect of MSE loss coefficient β on preprocessor trained with UniformCER selection, Tesseract and 8% budget.

4.3.3 Effect of MSE Loss

In this section, we study the effect of the MSE loss coefficient β when the system is trained with a low query budget. We perform experiments on both datasets with UniformCER selection, Tesseract OCR, and 8% query budget. We vary β from 0 to 1 with increments of 0.2. For the POS dataset, we noticed that the performance does not vary significantly across different values of β. However, visual inspection of the preprocessed images produced by $\beta = 0$ (Fig. 4.7a) and $\beta = 1$ (Fig. 4.7b) indicates that they yield significantly different images. If the full POS receipt images are passed as input through an OCR engine (instead of text strips), the noise around the text in different parts of Fig. 4.7a can lead to incorrect text recognition output. Such noise is absent in Fig. 4.7b due to the MSE loss, making it amenable for preprocessed images to be passed through an OCR engine. A similar phenomenon is also observed for the VGG dataset, but the word-level accuracy at $\beta = 0$ setting is 2% lower than the word-level accuracy at $\beta = 1$.

4.3.4 Pruning Receipt Images

We perform the data pruning experiments on the POS dataset with the Tesseract OCR engine. $n\%$ of the receipt images in the POS train set are pruned, where $n \in \{10, 20, 30, 40, 50\}$. The pruned dataset is used to train the system with single jitter, and the results are shown in Figure 4.8. We can observe that the text recognition performance of Tesseract is on-par with the full dataset for 10%, 20%, and 30% of data pruning. Further, pruning 30% of the training dataset yields a 12.5% reduction in OCR engine queries without any decrease in word-level accuracy. There is an implicit reduction in the number of queries since pruning document images also removes text strips that are never used to query the OCR engine. However, pruning more than 30% of receipt images leads to a significant drop in test accuracy. Figure 4.9 shows a few sample images from the pruned set. These images have less noise, less textual content, and legible fonts. Hence, Tesseract can accurately recognize the text in them without any preprocessing.

We use the subset of data obtained by pruning 30% of the images to train the preprocessor and reduce the number of OCR engine queries using UniformCER selection. We choose 2.5%, 4%, 8%, 16%, and 32% as the query budgets. It is important to note that performing query-efficient black-box approximation with a pruned dataset leads to a further reduction in OCR engine queries. For example, the 4% query budget

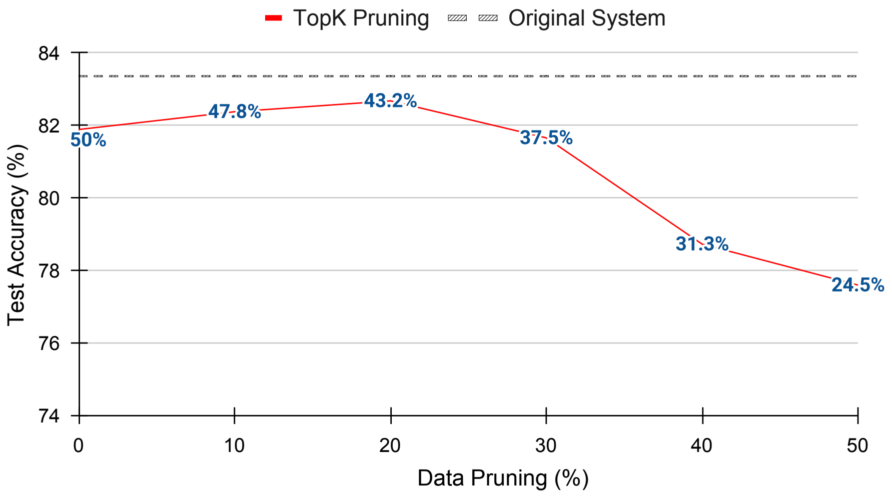
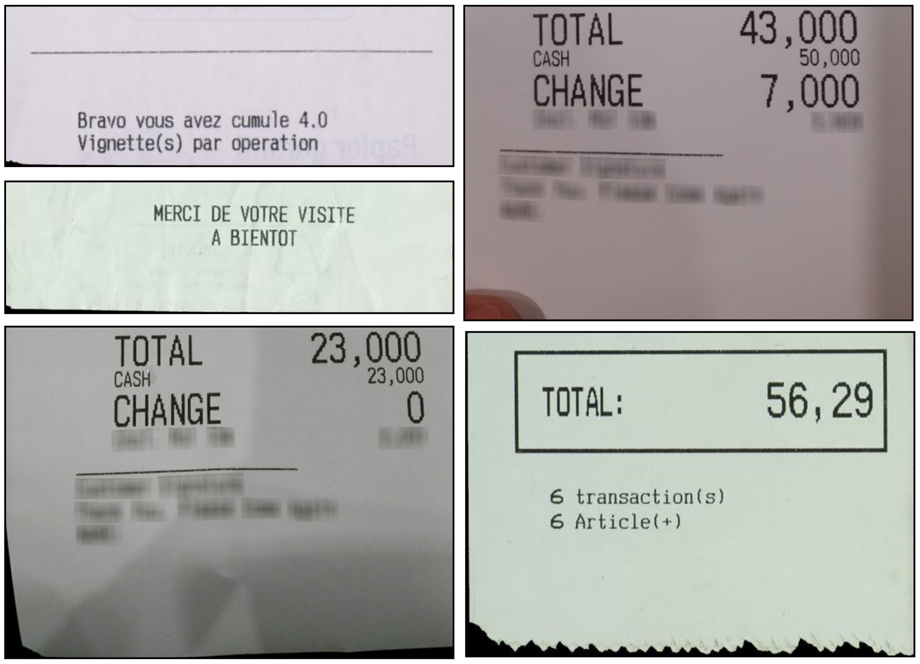


Figure 4.8: Test Accuracy and Tesseract queries (%) across different data pruning rates. The numbers in blue indicate query % using single jitter with respect to the original system.



Bravo vous avez cumule 4.0
Vignette(s) par operation

MERCI DE VOTRE VISITE
A BIENTOT

TOTAL 43,000
CASH 50,000
CHANGE 7,000

TOTAL 23,000
CASH 23,000
CHANGE 0

TOTAL: 56,29

6 transaction(s)
6 Article(+)

Figure 4.9: Samples with 0 *mean CER* that were pruned since they rank lowest with respect to Tesseract's CER.

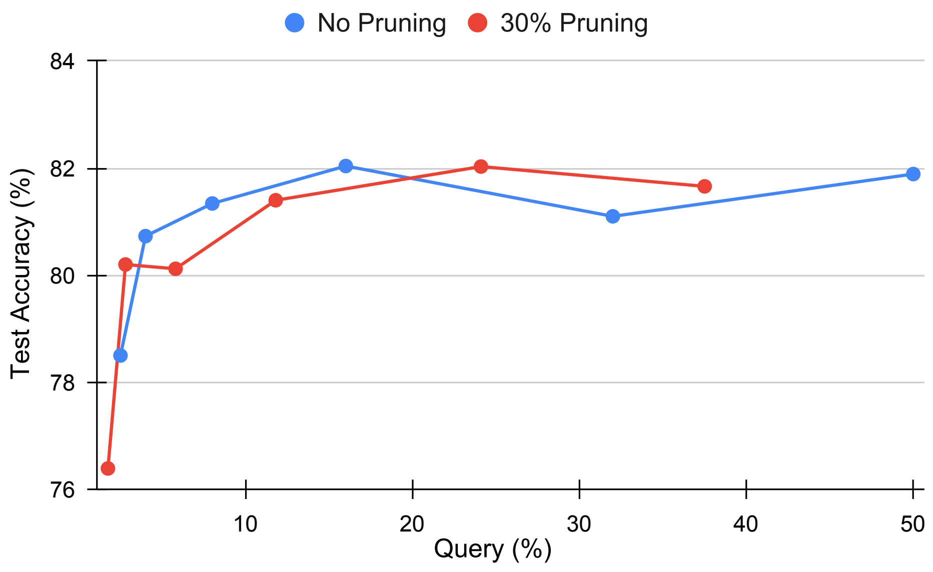


Figure 4.10: Performance of system with original and pruned POS dataset using UniformCER sample selection for reducing queries to Tesseract.

results in 2.8% of the total queries when the pruned dataset (30% pruning) is used for training. Figure 4.10 depicts the results for different query budgets with and without data pruning. We can observe that the two curves are close, and the relationship between the curves changes for different (%) query ranges. For instance, between 20% and 40% query budgets, the preprocessor trained with the pruned dataset performs better than the original dataset. Further, the system's performance with the pruned dataset at a very low query budget is better than that of the original dataset (first point in both curves). However, the pruned dataset performs slightly worse than the original dataset between 3% and 20% query budgets. This inconsistency in the results indicates that the receipt images can be selected with a better criterion so that the total OCR queries can be reduced by combining dataset pruning and query-efficient black-box approximation without compromising text recognition performance.

Dataset	Budget	Mode	Selection Method			
			Tesseract		EasyOCR	
			Random	UniformCER	Random	UniformCER
POS	4%	Noise	78.36	80.73	62.24	61.20
		Tracking	**78.77**	**81.12**	**63.57**	**62.20**
	8%	Noise	**79.20**	81.34	**65.27**	65.02
		Tracking	80.77	**82.35**	61.90	**65.59**
VGG	4%	Noise	**62.04**	62.86	**56.00**	56.12
		Tracking	57.70	**64.62**	48.20	**57.10**
	8%	Noise	**62.16**	63.86	**56.62**	57.80
		Tracking	60.60	**64.10**	55.47	**58.03**

Table 4.5: Comparison of test accuracy (%) between noise addition and label tracking.

4.3.5 Label Tracking Experiments

For the label tracking experiments, window size $m = 5$ is used and the weights $w_1, .., w_5$ are assigned as $(1, 0.7, 0.4, 0.2, 0.1)$ respectively. We use UniformCER as the sample selection algorithm with 4% and 8% query budgets. The results for both POS and VGG datasets with Tesseract and EasyOCR are shown in Table 4.5. For many settings, tracking leads to performance that is on par or better than noise-based approximation, demonstrating that the history of OCR engine queries for selected samples can compensate for querying the OCR engine with noisy samples. Moreover, tracking improves the OCR performance when the preprocessor is trained with UniformCER selection on both datasets. However, tracking leads to a reduction in text recognition accuracy for random sampling in most cases. These results indicate that label tracking can be helpful in training the system with low query budgets and UniformCER selection.

Chapter 5

Conclusion and Future Work

In this paper, we propose a sample selection scheme that drastically reduces the number of queries to an OCR engine for efficiently training an OCR preprocessor using differentiable bypass. We demonstrate that using the selection algorithms with very low query budgets can significantly boost the text recognition performance for both open-source and commercial OCR engines. Specifically, the text recognition performance of Google Vision API on receipt images improved by 4.5% with only 2.5% queries of the API. The huge reduction in the number of API queries also resulted in 32x less monetary cost when compared to the original system. For open-source OCR engines like Tesseract and EasyOCR, training the preprocessor with 8% query budget resulted in text recognition performance that was only 2% lower than the original system. Training the system with 8% query budget also decreases the training time by 60%. Furthermore, increasing the query budget beyond 8% leads to significantly larger training time with a small improvement in OCR performance. Moreover, pruning the training dataset of document images can further reduce the OCR engine queries when combined with sample selection techniques like UniformCER. Finally, we can utilize the past predictions of the OCR engine to improve the performance of OCR engine when UniformCER selection is used for training the document preprocessor.

In the original system, the OCR engine is queried many times for the POS dataset because both the OCR engine and the CRNN model are passed text strips as input. Text strips are used since CRNN can only process word images. Hence, another way to make the training query efficient is to query the OCR engine with full document images instead of text strips. Modern OCR engines can predict text bounding boxes and the text in them for a given document image. Hence, the predicted bounding boxes from a powerful OCR engine can be used to crop the document image into text strips for the CRNN model. Since the number of text strips in the training dataset is much larger than the number of document images, querying the OCR engine only for document images can reduce the total number of queries significantly.

As part of document pruning, we selected a smaller subset of the training set and trained the system with the subset. The results show that pruning 30% of images using the average CER of text strips in each image does not affect the final performance of the preprocessor. However, when training the system with the pruned dataset and UniformCER selection with low query budgets, we noticed inconsistent results. Hence, other data pruning techniques can be explored to remove document images from the training set. Further, images can be selected in each epoch instead of training the system with the pruned dataset, as discussed in section 2.2.2 (epoch sample selection).

Label tracking requires assigning values for label weights and window size. In our setup, manually tuning the weights can be time-consuming and cumbersome. Hence, the label-tracking algorithm can be improved by dynamically generating the weights during training. The effect of different window sizes on the OCR performance can also be analyzed. Further, a deeper investigation of the relationship between label tracking and the selection algorithm can be conducted to understand the improvement in performance for UniformCER selection and the decline in performance for random sampling.

Bibliography

[1] Ayantha Randika, Nilanjan Ray, Xiao Xiao, and Allegra Latimer. Unknown-box approximation to improve optical character recognition performance. In *International Conference on Document Analysis and Recognition*, pages 481–496. Springer, 2021.

[2] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z Li. Craft objects from images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6043–6051, 2016.

[3] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. East: an efficient and accurate scene text detector. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 5551–5560, 2017.

[4] Jian Ye, Zhe Chen, Juhua Liu, and Bo Du. Textfusenet: Scene text detection with richer fused features. In *IJCAI*, volume 20, pages 516–522, 2020.

[5] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[6] Darwin Bautista and Rowel Atienza. Scene text recognition with permuted autoregressive sequence models. In *European Conference on Computer Vision*, pages 178–196. Springer, 2022.

[7] Yue He, Chen Chen, Jing Zhang, Juhua Liu, Fengxiang He, Chaoyue Wang, and Bo Du. Visual semantics allow for textual reasoning better in scene text recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 888–896, 2022.

[8] Alon Jacovi, Guy Hadash, Einat Kermany, Boaz Carmeli, Ofer Lavi, George Kour, and Jonathan Berant. Neural network gradient-based learning of black-box function interfaces. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=r1e13s05YX`.

[9] Nhat Nguyen. Extending differentiable programming to include non-differentiable modules using differentiable bypass for combining convolutional neural networks and dynamic programming into an end-to-end trainable framework. Master's thesis, University of Alberta, 2019.

[10] Nhat M Nguyen and Nilanjan Ray. End-to-end learning of convolutional neural net and dynamic programming for left ventricle segmentation. In *Medical Imaging with Deep Learning*, pages 555–569. PMLR, 2020.

[11] Xiangyi Chen, Sijia Liu, Kaidi Xu, Xingguo Li, Xue Lin, Mingyi Hong, and David Cox. Zo-adamm: Zeroth-order adaptive momentum method for black-box optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

[12] Saeed Ghadimi and Guanghui Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4): 2341–2368, 2013.

[13] Sijia Liu, Pin-Yu Chen, Xiangyi Chen, and Mingyi Hong. signsgd via zeroth-order oracle. In *International Conference on Learning Representations*, 2018.

[14] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 ieee symposium on security and privacy (sp)*, pages 1277–1294. IEEE, 2020.

[15] Minhao Cheng, Simranjit Singh, Patrick Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. Sign-opt: A query-efficient hard-label adversarial attack. *arXiv preprint arXiv:1909.10773*, 2019.

[16] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In *International Conference on Machine Learning*, pages 2137–2146. PMLR, 2018.

[17] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[18] Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*, 2020.

[19] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.

[20] Krishnateja Killamsetty, S Durga, Ganesh Ramakrishnan, Abir De, and Rishabh Iyer. Grad-match: Gradient matching based data subset selection for efficient deep model training. In *International Conference on Machine Learning*, pages 5464–5474. PMLR, 2021.

[21] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020.

[22] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. *Advances in Neural Information Processing Systems*, 34:20596–20607, 2021.

[23] Angela H Jiang, Daniel L-K Wong, Giulio Zhou, David G Andersen, Jeffrey Dean, Gregory R Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C Lipton, et al. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019.

[24] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

[25] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3):293–321, 1992.

[26] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[27] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[28] Shakir Mohamed, Mihaela Rosca, Michael Figurnov, and Andriy Mnih. Monte carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, 21(1):5183–5244, 2020.

[29] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[30] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

[31] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[32] Ding-Xuan Zhou. Universality of deep convolutional neural networks. *Applied and computational harmonic analysis*, 48(2):787–794, 2020.

[33] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, 3(5):551–560, 1990.

[34] Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[35] Andrew Trask, Felix Hill, Scott E Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural arithmetic logic units. *Advances in neural information processing systems*, 31, 2018.

[36] Ethan Tseng, Felix Yu, Yuting Yang, Fahim Mannan, Karl ST. Arnaud, Derek Nowrouzezahrai, Jean-François Lalonde, and Felix Heide. Hyperparameter optimization in black-box image processing using differentiable proxies. *ACM Trans. Graph.*, 38(4), jul 2019. ISSN 0730-0301. doi: 10.1145/3306346.3322996. URL `https://doi.org/10.1145/3306346.3322996`.

[37] Sheng He and Lambert Schomaker. Deepotsu: Document enhancement and binarization using iterative deep learning. *Pattern recognition*, 91:379–390, 2019.

[38] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern recognition*, 33(2):225–236, 2000.

[39] Qiang Chen, Quan-sen Sun, Pheng Ann Heng, and De-shen Xia. A double-threshold image binarization method based on edge detector. *Pattern recognition*, 41(4):1254–1267, 2008.

[40] Utpal Garain, Atishay Jain, Anjan Maity, and Bhabatosh Chanda. Machine reading of camera-held low quality text images: an ica-based image enhancement approach for improving ocr accuracy. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[41] Ankit Lat and CV Jawahar. Enhancing ocr accuracy with super resolution. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 3162–3167. IEEE, 2018.

[42] Xujun Peng and Chao Wang. Building super-resolution image generator for ocr accuracy improvement. In *International Workshop on Document Analysis Systems*, pages 145–160. Springer, 2020.

[43] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[44] Baoguang Shi, Xiang Bai, and Cong Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[45] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[46] Guodong Xu, Ziwei Liu, and Chen Change Loy. Computation-efficient knowledge distillation via uncertainty-aware mixup. *Pattern Recognition*, page 109338, 2023.

[47] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[48] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[49] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[50] Tianyi Zhou and Jeff Bilmes. Minimax curriculum learning: Machine teaching with desirable difficulties and scheduled diversity. In *International Conference on Learning Representations*, 2018.

[51] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes. Submodular subset selection for large-scale speech training data. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3311–3315. IEEE, 2014.

[52] Tianyi Zhou, Shengjie Wang, and Jeffrey Bilmes. Curriculum learning by dynamic instance hardness. *Advances in Neural Information Processing Systems*, 33:8602–8613, 2020.

[53] Jared Fernandez and Doug Downey. Sampling informative training data for rnn language models. In *Proceedings of ACL 2018, Student Research Workshop*, pages 9–13, 2018.

[54] Angelos Katharopoulos and Francois Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2525–2534. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/katharopoulos18a.html.

[55] Baharan Mirzasoleiman, Jeff Bilmes, and Jure Leskovec. Coresets for data-efficient training of machine learning models. In *International Conference on Machine Learning*, pages 6950–6960. PMLR, 2020.

[56] Andreas Krause and Daniel Golovin. Submodular function maximization. *Tractability*, 3:71–104, 2014.

[57] Cody Coleman, Christopher Yeh, Stephen Mussmann, Baharan Mirzasoleiman, Peter Bailis, Percy Liang, Jure Leskovec, and Matei Zaharia. Selection via proxy: Efficient data selection for deep learning. *arXiv preprint arXiv:1906.11829*, 2019.

[58] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

[59] Burr Settles. Active learning literature survey. *Machine Learning*, 2009.

[60] Reza Zanjirani Farahani and Masoud Hekmatfar. *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media, 2009.

[61] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[62] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Submodularity in data subset selection and active learning. In *International conference on machine learning*, pages 1954–1963. PMLR, 2015.

[63] Fan Zhou and Chengtai Cao. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4714–4722, 2021.

[64] Da Yin, Xiao Liu, Xiuyu Wu, and Baobao Chang. A soft label strategy for target-level sentiment classification. In *Proceedings of the Tenth Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 6–15, 2019.

[65] Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *Advances in neural information processing systems*, 32, 2019.

[66] Chloé Artaud, Nicolas Sidère, Antoine Doucet, Jean-Marc Ogier, and Vincent Poulain D'Andecy Yooz. Find it! fraud detection contest report. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 13–18. IEEE, 2018.

[67] Zheng Huang, Kai Chen, Jianhua He, Xiang Bai, Dimosthenis Karatzas, Shijian Lu, and CV Jawahar. Icdar2019 competition on scanned receipt ocr and information extraction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1516–1520. IEEE, 2019.

[68] Seunghyun Park, Seung Shin, Bado Lee, Junyeop Lee, Jaeheung Surh, Minjoon Seo, and Hwalsuk Lee. Cord: a consolidated receipt dataset for post-ocr parsing. In *Workshop on Document Intelligence at NeurIPS 2019*, 2019.

[69] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Synthetic data and artificial neural networks for natural scene text recognition. *arXiv preprint arXiv:1406.2227*, 2014.

[70] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[71] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[72] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

Appendix A

Additional Results and Hyperparameter Tuning

Thus section discusses performance metrics apart from word-level accuracy, hyper-parameter tuning details and visual analysis showing significance of a small number of OCR queries.

A.1 Character Error Rate (CER) of OCR Engine

Table A.1 shows the average Character Error Rate (CER) (Equation 3.1) for Tesseract and EasyOCR with UniformCER and TopKCER selection across different query budgets. It was used by [1] as an evaluation metric, apart from word-level accuracy. The observations from this table align with the conclusions drawn from Tables 4.1 and 4.2. Specifically, the CER of OCR engines with random selection is always higher than the CER-based selection techniques. Further, the CER for both UniformCER and TopKCER with 8% query budget is only 1% lower than the CER of the original system.

A.2 Comparison of 0% and 2.5% Query Budget

Figure A.1 depicts the transformations performed by preprocessors trained with 0% and 2.5% query budget on the POS dataset using UniformCER selection and Tesseract OCR engine. We can clearly observe that making no queries to the OCR engine results in degradation of the document images, hindering the engine's text recognition performance. However, with only 2.5% queries, there is a notable difference in the quality of preprocessed images. The text is darker and sharper, and background noise is removed more effectively. The improvement in the quality of preprocessed images with a minimal number of OCR engine queries underscores the necessity to query the OCR engine, and also illustrates that a few queries are sufficient to train a good document preprocessor.

A.3 Hyperparameter Tuning for CRNN and UNet

We use the Optuna [71] open-source library for the hyperparameter tuning experiments mentioned in section 4.1.2. The learning rates of UNet (α) and CRNN (η) were tuned and the validation accuracy was used to select the best values. The search space for both hyperparameters is as follows: $\alpha \in (0.00001, 0.0005)$ and $\eta \in (0.00001, 0.001)$. The hyperparameter tuning was performed with 100 trials. The TPE (Tree-structured Parzen Estimator) algorithm [72] was used for hyperparameter optimization. The tuning was only performed for EasyOCR with POS dataset and 4% query budget using TopKCER selection.

Dataset	Budget	Selection Method		
		Random	UniformCER	TopKCER
POS	4%	9.6	9.0	**7.9**
	8%	10.1	**9.2**	9.7
	0%		11.6	
	100%		8.7	
VGG	4%	15.6	15.3	**15.1**
	8%	15.5	**14**	15.1
	0%		25.5	
	100%		14.7	

(a) Tesseract

Dataset	Budget	Selection Method		
		Random	UniformCER	TopKCER
POS	4%	19.0	**17.4**	18.0
	8%	17.9	17.5	**17.2**
	0%		21.5	
	100%		16.5	
VGG	4%	18.1	17.8	**17.5**
	8%	17.6	**17.3**	17.7
	0%		23.5	
	100%		17.2	

(b) EasyOCR

Table A.1: Average Character Error Rate (CER) of OCR Engine ($\times100$) for different selection methods across POS and VGG datasets. No selection methods were used for 0% and 100% budgets. Lower CER indicates better text recognition performance.

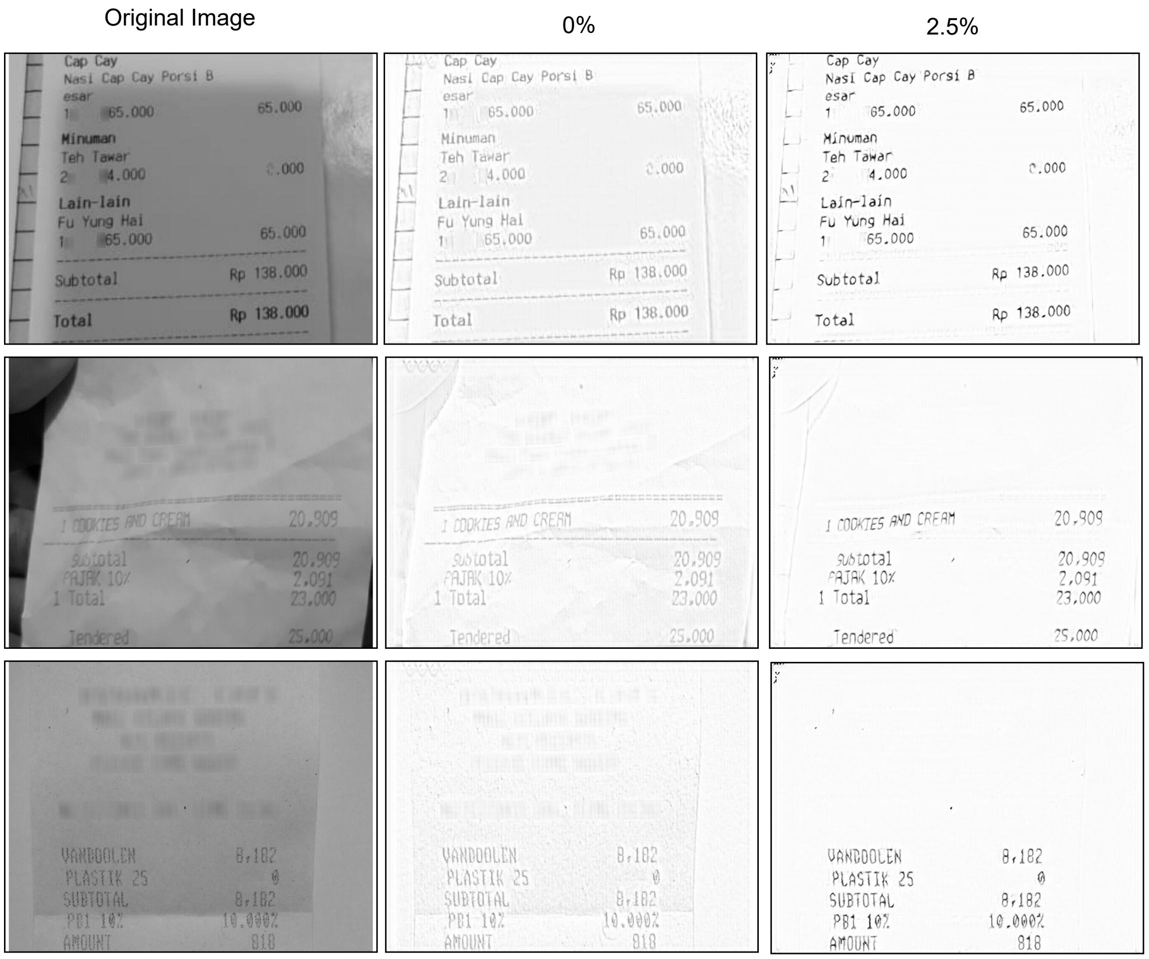


Figure A.1: Preprocessor output for few randomly selected images in the POS dataset. The preprocessors were trained with Tesseract OCR engine and Uniform-CER selection was used for 2.5 % budget. **Column 1:** Original Images. **Column 2:** 0% query budget. **Column 3:** 2.5% query budget.