Synthetic Document Generation for Form Understanding with Vision-Language Models

Angelo Roy Thompson[1,2], Andrei Alexandru Chiochiu[1,2], and Noah Scharrenberg[2]

[1] Maastricht University, Maastricht, Netherlands
{"ar.thompson, a.chiochiu}@student.maastrichtuniversity.nl
[2] Contractuo, Eindhoven, Netherlands
{angelo.thompson, andrei.chiochiu, noah}@contractuo.com
https://contractuo.com

Abstract. Form understanding remains a persistent challenge for document processing due to the scarcity of annotated training data and the sensitive nature of real-world forms. Vision–Language Models (VLMs) have shown promise in addressing the structural complexity of forms, but their performance is constrained by limited datasets. To address this gap, we propose a Synthetic Document Generation (SDG) pipeline that transforms empty templates into realistic, filled documents through a four-stage process: region detection, content generation, statistical placement modeling, and controlled degradation. The pipeline leverages YOLO-based region detection, large language model–driven content generation, and degradation frameworks such as Augraphy and Albumentations to produce high-fidelity, semantically coherent training data.
We evaluate the effectiveness of synthetic data in isolation and in combination with real-world corpora by fine-tuning the olmOCR VLM on three regimes: real-only, synthetic-only, and hybrid datasets. Results show that synthetic data alone achieves competitive performance, but hybrid training consistently outperforms both real-only and synthetic-only setups, yielding the lowest error rates and highest semantic similarity scores. Comparisons against Tesseract and PP-OCR further demonstrate the advantage of VLMs trained with SDG-augmented data for structured form understanding. These findings confirm synthetic data as a scalable and practical supplement to real-world datasets, enhancing robustness while reducing reliance on real-world datasets.

Keywords: Synthetic Data · Vision Language Models · Optical Character Recognition · Form Understanding

1 Introduction

Optical Character Recognition (OCR) technology serves as a cornerstone of modern document processing, enabling automatic extraction of textual information from images, scanned, and digital documents [29]. While traditional OCR systems excel at clean, printed text, structured documents, such as forms, present

unique challenges due to their complex layouts, multimodal elements, including checkboxes, signatures, and tables, combined with varying degrees of degradation from scanning or photocopying processes [1].

The emergence of Vision-Language Models (VLMs) has revolutionized document understanding by jointly modeling visual and textual information through transformers [37, 18, 38]. These models demonstrate superior performance on structured document tasks compared to traditional OCR approaches, as they can capture spatial relationships and contextual dependencies across document elements [38, 37, 4, 18, 27]. However, VLMs require substantial amounts of annotated training data to achieve optimal performance [3, 21], which is particularly scarce for form-centric tasks due to privacy concerns surrounding personal information and the high complexity of manual annotation [21, 16].

Form understanding faces a critical data scarcity problem that significantly impedes model development [16, 1]. Unlike general documents, forms typically contain sensitive personal information, making large-scale public datasets rare [5]. The few available datasets, such as FUNSD [16], remain limited in size and diversity [16], constraining the development of robust form understanding systems. Additionally, forms exhibit tremendous layout diversity across domains including healthcare, finance, and government administration, requiring robust training data that captures this variability [34, 1]. Manual annotation of forms is prohibitively expensive, as it requires transcribing text content, linking key-value pairs, annotating widget states, such as checkbox selections, and providing precise spatial grounding through bounding boxes [16, 3].

Synthetic Data Generation (SDG) presents a promising solution to address this data bottleneck [15, 20]. By programmatically creating realistic form images, SDG can significantly reduce reliance on real-world datasets while providing coverage of diverse layouts, content variations, and realistic degradation effects [26, 14]. This approach is particularly valuable for forms, where template structures are often publicly available even when filled instances remain confidential due to privacy or confidentiality constraints.

This paper investigates whether SDG can effectively supplement real-world training data for VLM-based form understanding. Specifically, we examine two research questions: First, can an SDG pipeline produce realistic form images that improve VLM performance on form understanding tasks? Second, how does synthetic data compare to real-world data when training VLMs for form OCR?

To address these questions, we propose a comprehensive SDG pipeline specifically designed for form documents. Our approach combines computer vision (CV)-based region detection, large language model (LLM)-driven content generation, statistical placement modeling, and realistic image degradation techniques. The pipeline transforms empty form templates into filled realistic documents.

Our primary contribution is the proposed SDG pipeline tailored for form understanding, which addresses the unique challenges of multimodal form elements and layout diversity. Through systematic evaluation using hybrid training regimes that combine synthetic and real-world data, we demonstrate improve-

ments in form understanding performance. The experimental evaluation encompasses comparisons against traditional OCR engines, and analysis of synthetic data's impact on VLM training, as detailed in Section 4 and Section 5. Our methodology is presented in Section 3, while Section 2 positions our work within the broader context of OCR, VLMs, and synthetic data research. We discuss limitations and practical implications in Section 6 and conclude with future work directions in Section 7.

2 Related Work

This section examines the research landscape that includes traditional OCR systems, VLMs for document understanding, CV techniques for document analysis, and SDG approaches. We analyze each area's capabilities, limitations, and relevance to form understanding challenges.

2.1 Optical Character Recognition and Vision-Language Models for Document Understanding

Traditional OCR systems have evolved significantly from rule-based approaches to modern deep learning architectures [33, 23, 32]. Tesseract, one of the most widely adopted OCR engines, initially relied on heuristic rules and segmentation-driven pipelines [33, 23] but has since incorporated Long Short-Term Memory (LSTM) networks with Connectionist Temporal Classification (CTC) loss for improved sequence recognition [23, 32]. While Tesseract demonstrates strong performance on clean, printed text, it struggles with complex layouts and degraded document quality, often failing to maintain proper reading order in structured documents [23, 40].

Modern Convolutional Neural Network (CNN)-based systems address many traditional OCR limitations [32, 12]. PaddleOCR's PP-OCR exemplifies this advancement, employing optimized CNN architectures combined with recurrent networks and CTC loss to achieve state-of-the-art results on multilingual benchmarks [12, 9]. These systems offer significant improvements in recognition accuracy and robustness across different languages and font variations [12, 9]. However, CNN-based approaches maintain a fundamental limitation: they operate on isolated text regions without capturing document-level context or spatial relationships between elements [37]. This constraint proves particularly problematic for structured documents like forms, where understanding field relationships and layout semantics is crucial for accurate information extraction [16, 1].

VLMs represent a paradigm shift toward holistic document understanding [37, 38]. LayoutLM pioneered this approach by augmenting BERT-style language models with two-dimensional positional embeddings, enabling joint reasoning over textual content and spatial layout [37]. This multimodal architecture demonstrates superior performance on tasks requiring spatial understanding, such as key-value extraction from forms and table parsing [37, 38]. DocFormer extends

this concept further with end-to-end transformer architectures designed specifically for document understanding [4], while Donut introduces OCR-free document parsing capabilities that directly generate structured outputs from document images [18].

The recent olmOCR model advances VLM capabilities through novel document anchoring techniques, enabling robust text extraction from complex layouts while maintaining spatial awareness [27]. However, despite these architectural improvements, VLMs face a critical bottleneck: they require substantially more annotated training data than traditional OCR systems to achieve optimal performance [3, 21]. This data need is particularly problematic for specialized domains like forms, where annotation complexity and privacy constraints limit dataset availability [16, 5].

2.2 Computer Vision Techniques for Document Analysis

Object detection frameworks play a crucial role in document analysis pipelines [19, 39]. You Only Look Once (YOLO) architectures have proven particularly effective for document layout analysis due to their balance of speed and accuracy [28, 39]. YOLO-based systems can efficiently detect and classify document regions, including text blocks, tables, figures, and form elements [28, 39]. Recent versions like YOLOv11 offer improved performance through architectural enhancements and better training procedures, making them suitable for real-time document processing applications [30].

However, YOLO-based detection faces a significant limitation in document contexts. The models struggle with highly variable layouts [39]. Detection accuracy directly impacts downstream processing quality, creating a bottleneck where missed or misclassified regions would propagate as errors through the entire pipeline [40, 30]. Additionally, YOLO models primarily focus on spatial detection without incorporating semantic understanding of document content, limiting their ability to reason about field relationships or content appropriateness [37, 39].

Layout analysis tools, such as LayoutParser, provide integrated frameworks combining multiple computer vision models for comprehensive document understanding [19]. These tools offer pre-trained models for various document types and standardized interfaces for common document analysis tasks [19]. While valuable for rapid prototyping and general document processing, such tools often lack the specialization required for specific domains like form understanding, where subtle layout variations and domain-specific elements require targeted approaches [1].

2.3 Synthetic Data Generation for Optical Character Recognition

SDG has emerged as a powerful strategy for addressing training data scarcity in OCR applications [15, 24]. Early work like SynthText demonstrated that large-scale synthetic scene text datasets could significantly improve text recognition accuracy when combined with real-world training data [15]. This foundational

work established key principles: synthetic data must exhibit sufficient realism to transfer to real domains, and careful attention to degradation and variability is essential for effective domain adaptation [15].

Document-specific synthetic generation approaches have built upon these principles [20, 26]. SynthDoG generates realistic document images with automatic annotations, targeting various document types including forms and invoices [20]. The system employs template-based generation with programmatic content filling and realistic rendering [20]. However, SynthDoG focuses primarily on general document layouts and does not address the specific challenges of form elements like checkboxes, radio buttons, and signature fields that require specialized handling [20].

DocSynth takes a different approach, emphasizing structured business documents with tabular data and standardized layouts [26]. While effective for its target domain, DocSynth lacks the flexibility needed for the diverse form layouts encountered in real-world applications [26]. The system also relies on predefined templates without the adaptive content generation capabilities needed to produce semantically coherent form completions [26].

DocCreator represents another significant contribution to synthetic document generation, offering a comprehensive framework for creating artificial documents with controlled degradation and layout variations [17]. The system provides extensive customization options for paper texture, printing effects, and scanning artifacts [17]. However, DocCreator requires substantial manual configuration for each document type and lacks automated content generation capabilities, limiting its scalability for large-scale synthetic dataset creation [17].

Degradation frameworks, such as Augraphy, complement SDG by introducing realistic artifacts that bridge the domain gap between synthetic and real-world documents [14]. Augraphy simulates various real-world effects including scanning noise, compression artifacts, ink bleeding, and paper aging [14]. These degradations are essential for effective synthetic data utilization, as clean synthetic images often fail to generalize to the imperfect conditions encountered in real-world document processing [14]. However, applying appropriate degradation requires careful tuning to avoid over-degradation that obscures important document features or under-degradation that fails to provide adequate domain adaptation [13, 14].

2.4 Form Understanding Challenges and Gaps

Forms present unique challenges that distinguish them from general document understanding tasks [1]. Unlike linear text documents, forms contain heterogeneous elements including typed text, handwritten entries, checkboxes, radio buttons, signatures, and tabular structures [34]. These elements often exhibit non-standard reading orders and complex spatial relationships that challenge both traditional OCR systems and modern VLMs [1, 40].

The FUNSD dataset represents one of the few publicly available benchmarks specifically designed for form understanding [16]. While valuable for research, FUNSD's limited size (199 forms) and narrow domain coverage constrain model

development and evaluation [16]. The dataset's focus on noisy, scanned forms also limits its applicability to modern digital form processing scenarios [16].

Template-free form parsing approaches attempt to handle layout variability without relying on predefined structures [10]. However, these methods often struggle with the semantic understanding required to properly associate field labels with their corresponding values, particularly in complex multi-column layouts or nested form structures [37, 1].

Despite advances in both synthetic generation and VLM architectures, significant gaps remain in applying these technologies to form understanding [1]. Existing synthetic generation approaches lack the sophisticated content modeling required for realistic form completion, often producing semantically inconsistent or implausible field values [20, 26]. Additionally, most systems fail to properly model the statistical distributions of human form-filling behavior, resulting in synthetic data that appears artificial to trained models [20, 17].

To address these gaps, we introduce an SDG pipeline that integrates advanced content generation, realistic placement modeling, and degradation simulation, specifically targeting the challenges of form modelling.

3 Methodology

This section presents our SDG pipeline designed to address the data scarcity problem in form understanding. The methodology transforms empty form templates into realistic, filled documents through a synthetic four-stage process.

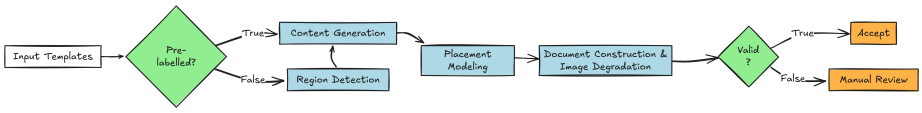

Fig. 1. High-level overview of the SDG pipeline design.

The SDG approach is built on three core principles: template preservation leverages abundant empty form templates to maintain realistic layout structures, semantic coherence ensures generated content maintains logical consistency within document context, and domain adaptation bridges the gap between synthetic and real-world document conditions through controlled degradation.

The pipeline operates through conditional branching and sequential transformations as illustrated in Figure 1. Each component can be mathematically formalized to provide precise algorithmic foundations for implementation and analysis.

3.1 Template Ingestion and Region Detection

The pipeline begins with template preprocessing, where input templates T_{input} are evaluated for existing annotations. This conditional processing can be ex-

pressed as:

$$R, L = \begin{cases} f_{existing}(T_{input}) & \text{if pre-labeled} \\ f_{detect}(T_{input}) & \text{otherwise} \end{cases} \quad (1)$$

where $R = \{r_1, r_2, ..., r_n\}$ represents detected regions with $r_i = (x_i, y_i, w_i, h_i)$, and $L = \{l_1, l_2, ..., l_n\}$ denotes corresponding class labels from the set $\{text, checkbox, radiobutton, circle - able field, signature, date, table\}$.

The detection function $f_{detect} : T_{input} \rightarrow (R, L)$ employs object detection to identify fillable regions within form templates. The quality of this detection stage directly impacts downstream performance, as missed regions result in incomplete forms while misclassified regions may receive inappropriate content.

3.2 Semantic Content Generation

Given detected regions and their classifications, the content generation stage produces contextually appropriate synthetic data through a two-phase process. First, structured attribute generation creates a coherent personal profile:

$$\mathcal{A} = f_{faker}(\text{seed}, \text{locale}) = \{name, address, phone, email, company, \dots\} \quad (2)$$

where f_{faker} samples from realistic demographic distributions while ensuring cross-field consistency. This structured approach prevents common synthetic data artifacts such as geographically inconsistent address-phone combinations.

The core content generation function operates as a conditional probability model:

$$C = f_{generate}(R, L, T_{input}, \mathcal{A}) = \{c_1, c_2, ..., c_n\} \quad (3)$$

where each content element c_i is generated by conditioning on region type l_i, spatial context within T_{input}, and the structured attributes $\mathcal{A}$. The generation function incorporates override capabilities, allowing semantic reasoning to correct misclassified regions when contextual evidence suggests errors.

3.3 Statistical Placement Modeling

Human form-filling behavior exhibits natural variability that synthetic systems must capture to avoid artificial appearance patterns. We model text placement as a stochastic process where positioning coordinates are sampled from distributions calibrated to realistic human behavior, ensuring that most samples align near field boundaries while occasional deviations mimic natural placement irregularities.

For each region $r_i = (x_i, y_i, w_i, h_i)$, the placement function computes final coordinates by treating the lower-left corner as an anchor point and sampling offsets from exponential distributions:

$$x'_i = x_i + \Delta x_i, \quad \Delta x_i \sim \text{Exp}(\sigma_{x,i}) \tag{4}$$
$$y'_i = y_i + \Delta y_i, \quad \Delta y_i \sim \text{Exp}(\sigma_{y,i}) \tag{5}$$

where the scale parameters are calibrated to ensure 95% of samples fall within 25% of the bounding box dimensions, providing realistic clustering near expected writing locations:

$$\sigma_{x,i} = \frac{f_x \cdot w_i}{-\ln(1-p)}, \quad \sigma_{y,i} = \frac{f_y \cdot h_i}{-\ln(1-p)} \tag{6}$$

with coverage probability $p = 0.95$, target fractions $f_x = f_y = 0.25$, and dimensions w_i, h_i representing region width and height respectively.

Font sizing follows adaptive scaling based on region geometry to ensure content fits appropriately within boundaries:

$$s_i \sim \mathcal{N}\left(\mu_i = 0.85 \cdot h_i, \sigma_i = \frac{0.10}{1.96} \cdot h_i\right) \tag{7}$$

where the standard deviation is chosen such that 95% of samples fall within 10% of the region height around the mean. Oversized samples that exceed width constraints are recursively adjusted downward with step size 0.001 to maintain boundary compliance. Text placement accounts for baseline differences, allowing words with descenders (e.g. g, j, p, q, y) to extend below the line naturally.

Element-specific placement modifications accommodate different form components that require specialized handling behaviors. For checkboxes and radio buttons, when the content generation produces a single token response, an "X" symbol is rendered with the font size following the same adaptive scaling as in Equation (7), but with increased mean ($\mu_i = 1.20 \cdot h_i$).

The horizontal placement distribution uses increased scatter ($f_x = 0.35$ instead of 0.25) to simulate uneven alignment, while fonts are selected from dedicated bold collections to reinforce visual weight. This combination of stochastic size increase and offset placement simulates the natural imprecision with which humans fill checkboxes compared to typed text.

Circle-able fields receive circular marks centered on bounding boxes with radius determined by:

$$r = \frac{1}{2}\sqrt{w^2 + h^2} \cdot (1 + \epsilon), \quad \epsilon \sim \mathcal{N}(0.20, 0.05) \tag{8}$$

where the base radius spans the region diagonal, and stochastic variation ϵ introduces realistic padding irregularities. Signature fields employ cursive fonts sampled from curated collections, with baseline adjustments to accommodate descender variations naturally present in handwritten signatures.

3.4 Document Construction and Quality Assurance

The placement function renders synthetic content onto templates, producing intermediate clean documents:

$$I_{clean} = f_{render}(T_{input}, C, \{(x'_i, y'_i, s_i)\}_{i=1}^{n}) \tag{9}$$

These clean documents undergo dual validation to ensure quality before final processing. The validation framework operates through parallel assessment channels:

$$v_{semantic} = f_{validate\_semantic}(I_{clean}, C, T_{input}) \tag{10}$$

$$v_{layout} = f_{validate\_layout}(I_{clean}, R, L, \{(x'_i, y'_i, s_i)\}_{i=1}^{n}) \tag{11}$$

The semantic validator assesses content plausibility, format compliance, and inter-field consistency, while the layout validator evaluates spatial placement quality, boundary compliance, and element overlap detection. Documents must satisfy both validators:

$$\text{Valid} = v_{semantic} \wedge v_{layout} \tag{12}$$

Failed documents are redirected for manual review, ensuring only high-quality samples enter the training dataset.

3.5 Degradation

The final pipeline stage introduces realistic imperfections to bridge the domain gap between clean synthetic documents and real-world scanned forms. The degradation function adapts to template characteristics:

$$I_{synthetic} = f_{degrade}(I_{clean}, T_{quality}, \mathcal{P}) \tag{13}$$

where $T_{quality} \in \{digital, scanned\}$ determines degradation strategy and $\mathcal{P}$ represents stochastic parameters controlling degradation intensity and type selection.

For digital templates, full-document degradation simulates the complete document lifecycle from printing through scanning. Scanned templates receive selective overlay degradation affecting only synthetic content, followed by harmonization to ensure seamless visual integration. Each degradation type (blur, noise, ink bleeding, compression artifacts) is applied probabilistically with parameters tuned to maintain the balance between realism and legibility.

The complete pipeline transformation can be summarized as:

$$\begin{aligned} T_{input} &\xrightarrow{f_{detect}} (R, L) \xrightarrow{f_{generate}} C \xrightarrow{f_{place}} I_{clean} \\ &\xrightarrow{f_{validate}} \begin{cases} I_{synthetic_v} = f_{degrade}(I_{clean}) & \text{if valid} \\ I_{synthetic_m} = f_{degrade}(I_{clean}) & \text{Manual Review} \end{cases} \end{aligned} \tag{14}$$

3.6 Output and Annotation Generation

The complete pipeline produces training data $I_{synthetic}$, which represents the final degraded document image. The systematic approach ensures consistent data quality across large-scale synthetic dataset generation while preserving the flexibility needed to handle diverse form layouts and content requirements.

In addition to automated LLM-based validation, additional manual inspection is necessary to address subtle errors such as nonsensical or duplicated field values within bounding boxes. The combination of automated filtering with targeted manual validation helps to accelerate the review process while maintaining the quality of the retained synthetic documents.

4 Experimental Setup

This section details the implementation of our SDG pipeline and the experimental framework used to evaluate its effectiveness for VLM training on form understanding tasks.

4.1 SDG Implementation Details

Region Detection We implemented the region detection stage using *YOLOv11*, trained using a progressive batch labeling strategy to optimize the efficiency of annotation. The construction of the training dataset followed an iterative approach.

- **Initial batch**: 50 manually labeled form templates
- **Model progression**: *YOLOv11s* for early batches, *YOLOv11l* for datasets exceeding 300 samples
- **Batch expansion**: Each subsequent batch was 1.25× larger than the previous
- **Semi-automatic labeling**: *Label Studio* integration for efficient correction of model predictions

The final detector *YOLO* was trained on 699 labeled pages with a 80/10/10 train/validation/test split, achieving $mAP@50$ of 0.842 in validation and 0.771 on test sets.

Content Generation Implementation Content generation utilized OpenAI's o3 [25] model with structured prompts and integrated components such as Faker [8] for consistent personal attributes, *custom prompts* tailored for region-specific content with context awareness (see Appendix B.1), and *fallback mechanisms* allowing LLM override for misclassified regions, with the generated content mapped to boundaries through unique identifiers to ensure precise spatial alignment during rendering.

Rendering and Degradation Document rendering utilized *Pillow* [7] with custom font management for different element types. Degradation used:

- **Augraphy [14]**: For document-specific degradation effects (scanning artifacts, compression, ink bleeding).
- **Albumentations [6]**: For additional image augmentation and layer harmonization,
- **Probabilistic application**: Each degradation type was applied with tuned probabilities for sample diversity.

Validation Implementation Quality assurance employed dual *OpenAI o3* validators with specialized prompts (see Appendix B.2, B.3) for semantic and layout validation. Documents passing both validators entered the training dataset; failures were redirected for manual review.

4.2 Datasets

Our experimental evaluation used three datasets designed to isolate contributions from synthetic data: **Internal Dataset**: Proprietary documents from Contractuo including invoices, forms, and administrative documents. Manual analysis of 200 random samples revealed 7.8% forms, 9.5% tables, and 82.7% other types of documents. The documents included both digital PDFs and scanned images with varying quality levels.

Table 1. Dataset composition and characteristics

Dataset	Pages	Languages	Source
Internal Real	12,750	Dutch, English	Contractuo documents
Synthetic	2,000	Dutch, English	SDG pipeline
Combined Hybrid	14,750	Dutch, English	Real + Synthetic

Synthetic Dataset: Generated using our SDG pipeline with form-heavy templates (see Appendix C) to complement the internal dataset's limited form coverage. Generation targeted realistic Dutch and English form completion scenarios.

Combined Dataset: Hybrid corpus with 86% real-world and 14% synthetic data, designed to evaluate the effects of synthetic supplementation while maintaining the real-world grounding.

All datasets used 80/20 train/validation splits with evaluation conducted on a separate holdout test set.

4.3 Labeling Pipeline

Converting complex PDF documents into VLM-compatible training format required high-fidelity text extraction while preserving semantic structure. Our auto-labeling pipeline combined:

- **LLMWhisperer [35]**: Primary extraction engine for document-to-text conversion with layout preservation
- **LayoutParser [19]**: Detectron2-based model for table and diagram detection
- **langdetect [22]**: Automatic language identification for multilingual support

Extracted content was formatted according to olmOCR's training schema, including metadata fields for rotation, language, and structural elements.

4.4 Model Training Configuration

Vision-Language Model We fine-tuned *olmOCR-7B-0225-preview*, selected for its document anchoring capabilities and suitability for form understanding tasks. While the base model provided reasonable extraction results on general documents, it produced inconsistent results on fillable forms due to limited form representation in the original training data.

Training Hyperparameters All models were trained with consistent configurations based on the original olmOCR paper:

Table 2. Training configuration parameters

Parameter	Value
Batch size	4
Learning rate	1e-4
Optimizer	AdamW
Schedule	Cosine annealing
Epochs	3
Hardware	Single NVIDIA H100 80GB

Three separate models corresponding to the internal, synthetic, and combined datasets were trained to allow for a direct comparison of the training regimes.

4.5 Baseline Models

We evaluated three OCR systems with different architectures: Tesseract v5, PaddleOCR v5, and the base VLM olmOCR.

All systems were used with their default configurations for document OCR tasks.

4.6 Evaluation Metrics and Protocol

Performance assessment used standard OCR metrics supplemented with semantic similarity measures. character-level metrics, including Character Error Rate (CER) and Word Error Rate (WER), are examined. Subsequently, semantic similarity metrics, namely BLEU, alongside ROUGE-1, ROUGE-2, and ROUGE-L, are assessed. All metrics were calculated based on the holdout test set.

5 Results

Table 3. Evaluation metrics for different models. Arrows indicate whether lower (↓) or higher (↑) is better.

Metric	Combined	Natural	Synthetic	olmOCR	PP-OCR	Tesseract
WER ↓	**0.1520**	0.1908	0.1709	0.661	0.878	1.0207
CER ↓	0.4245	**0.3171**	0.3243	0.654	0.817	0.8329
ROUGE-1 ↑	**0.9618**	0.9449	0.9372	0.654	0.296	0.3264
ROUGE-2 ↑	**0.8899**	0.8567	0.8557	0.533	0.189	0.1755
ROUGE-L ↑	**0.9089**	0.8762	0.8868	0.583	0.246	0.2310
BLEU ↑	**0.8440**	0.8138	0.7976	0.381	0.159	0.1809

Table 3 summarizes the evaluation metrics across all experimental conditions and baselines. The combined hybrid dataset, which integrates both real-world and synthetic samples, yielded the strongest overall performance. It achieved the lowest WER (0.152) and a low CER (0.425), alongside the highest semantic similarity scores across BLEU (0.844), ROUGE-1 (0.962), ROUGE-2 (0.890), and ROUGE-L (0.909). These results highlight the value of supplementing real-world training data with targeted synthetic forms to improve form understanding capabilities.

When considering individual datasets, training exclusively on real-world documents (Natural) produced robust results (WER = 0.191, CER = 0.317), but performance lagged behind the hybrid setting. Synthetic-only training (WER = 0.171, CER = 0.324) demonstrated that synthetic forms alone provide competitive performance, though slightly less consistent than real-world data. Importantly, both natural and synthetic regimes substantially outperformed the base olmOCR model (WER = 0.661, CER = 0.654), indicating the necessity of domain-specific training.

Finally, compared to baselines, the gap is significant: PP-OCR and Tesseract underperformed across all metrics, with WER exceeding 0.87 and 1.02 respectively, and BLEU scores falling below 0.20. This confirms that general-purpose OCR engines struggle with structured forms as compared to the VLMs, which are both more accurate at the character level and more faithful at the semantic level.

6 Discussion

While our experiments demonstrate the potential of synthetic data and VLMs for OCR on structured forms, they also reveal several limitations and challenges.

We note that all evaluation metrics are computed relative to the outputs of LLMWhisperer, which provided the training labels, rather than against absolute ground truth. Consequently, the reported improvements primarily reflect the

extent to which the models reproduce LLMWhisperer outputs. In practice, this constraint is negligible given the high reliability of its annotations.

Synthetic Data as a Supplement, Not a Replacement. The results demonstrate that synthetic data alone is insufficient to match the accuracy of models trained exclusively on real-world forms. This is expected, as synthetic templates cannot perfectly capture the complexity, noise, and diversity of real-world documents. However, when used in combination with real-world data, synthetic samples improved performance. The hybrid regime outperformed both real-only and synthetic-only training, confirming that synthetic data serves best as a **supplement** to real-world corpora, enriching the training distribution and improving generalization.

VLMs vs. Traditional OCR Engines. Comparisons between olmOCR and more traditional architectures revealed the advantages of VLMs for structured documents. While Tesseract and PaddleOCR achieved reasonable recognition rates on simple documents, they struggled with determining reading order with more complicated documents. In contrast, olmOCR leveraged multimodal representations to jointly reason about visual regions and textual content, yielding higher accuracy across all metrics. The trade-off lies in computational cost: olmOCR required high-performance GPUs (NVIDIA H100) for efficient training, whereas traditional systems are lightweight and easier to deploy on resource-constrained hardware.

Scalability and Annotation Efficiency. The use of LLMWhisperer for auto-labeling and progressive YOLO-assisted annotation proved effective in scaling the dataset with minimal manual effort. This workflow enabled us to reach the dataset sizes mentioned in Section 4.2 with substantially reduced annotation time compared to a fully manual effort. Such strategies are promising for organizations seeking to build domain-specific OCR systems without incurring prohibitive labeling costs. Nevertheless, the reliance on models for auto-labeling introduces risks of bias or hallucinations, requiring periodic human verification.

Synthetic Data Generator's Reliance on YOLO. A key limitation of our synthetic generation pipeline is its reliance on accurate region proposals from the YOLO detector. Although the LLM can, in principle, override incorrect class assignments during content synthesis, its outputs are still heavily influenced by the detector's labels and boxes; missed detections or mislabeled regions therefore propagate as omissions or misplacements. Empirically, we observed an approximately linear relationship between detector quality and downstream synthetic quality, indicating that improvements in YOLO performance translate directly into better generation outcomes. This dependence is partly rooted in the CNN-based detector's limited ability to capture global context compared to transformer-based alternatives - acting as a bottleneck. In practice, however, this limitation was not severe: the LLM frequently ignored clearly inconsistent or spurious bounding boxes and produced plausible content aligned with the

remaining reliable regions. Nonetheless, reducing detector errors remains important for maximizing yield.

No Automatic Annotation Ability. Another limitation of the synthetic data generator lies in its reliance on predefined templates. Because the generator overlays synthetic content onto existing form layouts rather than constructing documents from scratch, ground-truth labels cannot be automatically derived for input templates. This template dependence restricts the ability to generate ground-truth annotations. Future work could address this limitation by integrating generative layout models or procedural form synthesis methods capable of producing both the document image and its annotations simultaneously [20, 26].

7 Conclusion

This work examined whether synthetic data generation (SDG) can enhance Vision–Language Models (VLMs) for form understanding and how synthetic data compares to real-world corpora in OCR tasks. Our results show that while synthetic data in isolation cannot match the performance gain provided by real-world forms, the proposed SDG pipeline produces realistic samples that, when combined with real-world data, significantly improve model robustness. The hybrid training regime consistently outperformed both real-only and synthetic-only baselines, demonstrating that synthetic data serves best as a supplement, enriching training distributions and reducing generalization errors.

Looking forward, three areas stand out for future work: moving beyond template overlays to fully generative layout synthesis, which would enable automatic ground-truth annotation; reducing reliance on large proprietary LLMs by fine-tuning smaller models for cost-effective generation; and improving region detection through transformer-based approaches to minimize error propagation. Taken together, these directions point toward more scalable, accurate, and efficient pipelines, confirming SDG as a practical strategy for advancing OCR and structured document understanding.

Acknowledgments. We thank Contractuo for funding this research and providing constructive feedback and discussions. We also thank our university supervisors at Maastricht University, Prof. Gerhard Weiss and Prof. Evgueni N. Smirnov, for their guidance and constructive feedback.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abdallah, A., et al.: A comprehensive survey on form understanding: From heuristics to deep learning. Artificial Intelligence Review (2024). https://doi.org/10.1007/s10462-024-11000-0, https://link.springer.com/article/10.1007/s10462-024-11000-0

2. Aggarwal, M., Sarkar, M., Gupta, H., Krishnamurthy, B.: Multi-modal association based grouping for form structure extraction. In: The IEEE Winter Conference on Applications of Computer Vision. pp. 2075–2084 (2020)
3. Anonymous: How vision-language tasks benefit from large pre-trained models: A survey. arXiv preprint arXiv:2412.08158 (2024)
4. Appalaraju, S., Jasani, B., Kang, B., Bhunia, A.K., Xie, E.: Docformer: End-to-end transformer for document understanding. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 993–1003 (2021). https://doi.org/10.1109/ICCV48922.2021.00103
5. Bulatov, K., Emelianova, E., Tropin, D., Skoryukina, N., Chernyshova, Y., Sheshkus, A., Usilin, S., Ming, Z., Burie, J.C., Luqman, M.M., Arlazarov, V.V.: Midv-2020: A comprehensive benchmark dataset for identity document analysis. arXiv preprint arXiv:2107.00396 (2021), https://arxiv.org/abs/2107.00396
6. Buslaev, A., Iglovikov, V.I., Khvedchenya, E., Parinov, A., Druzhinin, M., Kalinin, A.A.: Albumentations: Fast and flexible image augmentations. Information **11**(2) (2020). https://doi.org/10.3390/info11020125, https://www.mdpi.com/2078-2489/11/2/125
7. Clark, J.A., Contributors, P.: Pillow: The friendly pil fork (python imaging library). https://github.com/python-pillow/Pillow (2025), accessed: 2025-09-04
8. Community, F.: Faker: A python library for generating fake data. https://faker.readthedocs.io/ (2025), accessed: 2025-09-04
9. Cui, C., Sun, T., Lin, M., Gao, T., Zhang, Y., Liu, J., Wang, X., Zhang, Z., Zhou, C., Liu, H., Zhang, Y., Lv, W., Huang, K., Zhang, Y., Zhang, J., Zhang, J., Liu, Y., Yu, D., Ma, Y.: Paddleocr 3.0 technical report (2025), https://arxiv.org/abs/2507.05595
10. Davis, B., Morse, B., Cohen, S., Price, B., Tensmeyer, C.: Deep visual template-free form parsing. arXiv preprint arXiv:1909.02576 (2019), https://arxiv.org/abs/1909.02576
11. Dooley, C.: Lcwa_gov_pdf dataset (2018), https://www.loc.gov/programs/web-archiving/about-this-program/, contributors: Jesse Johnston, Aly DesRochers, Grace Thomas, Abbie Grotke. Updated through 2019.
12. Du, Y., Chen, X., Liu, Q., Hu, M., Li, W., Zhou, B., Qiu, M., Zhou, S.: PP-OCR: A practical ultra lightweight OCR system. In: arXiv preprint arXiv:2009.09941 (2020)
13. Etter, D., Rawls, S., Carpenter, C., Sell, G.: A synthetic recipe for ocr. In: HLT-COE Workshop (2020), human Language Technology Center of Excellence, Johns Hopkins University / USC Information Sciences Institute
14. Groleau, A., Chee, K.W., Larson, S., Maini, S., Boarman, J.: Augraphy: A data augmentation library for document images. In: Proceedings of the 17th International Conference on Document Analysis and Recognition (ICDAR) (2023), https://arxiv.org/pdf/2208.14558.pdf
15. Gupta, A., Vedaldi, A., Zisserman, A.: Synthetic data for text localisation in natural images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2315–2324 (2016). https://doi.org/10.1109/CVPR.2016.254
16. Jaume, G., Ekenel, H.K., Thiran, J.P.: Funsd: A dataset for form understanding in noisy scanned documents. In: 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW). vol. 2, pp. 1–6. IEEE (2019)
17. Journet, N., Visani, M., Essoukri, I., Ogier, J.M.: Doccreator: a new software for creating synthetic datasets of documents. In: 2017 14th IAPR International Con-

ference on Document Analysis and Recognition (ICDAR). vol. 5, pp. 17–22. IEEE (2017)
18. Kim, G., Hong, T., Yim, M., Nam, J., Park, J., Yim, J., Hwang, W., Yun, S., Han, D., Park, S.: Ocr-free document understanding transformer. In: European Conference on Computer Vision (ECCV). pp. 498–517. Springer (2022). https://doi.org/10.1007/978-3-031-19815-1_29
19. Layout-Parser: A unified toolkit for deep learning based document image analysis. https://layout-parser.github.io/
20. Lee, S., Oh, T., Kim, S., Kim, D., Kim, S.: Synthdog: Synthetic document generator for ocr and nlp pre-training. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW). pp. 1336–1345 (2021)
21. Maurício, J., Domingues, I., Bernardino, J.: Comparing vision transformers and convolutional neural networks for image classification: A literature review. Applied Sciences **13**(9), 5521 (2023). https://doi.org/10.3390/app13095521
22. Mimino666: *langdetect*: Port of Google's language-detection library to Python, https://github.com/Mimino666/langdetect
23. Modi, M., Shah, A., Kothadiya, D., Rahevar, M.: Optical character recognition: Comparative analysis of tesseract and textract on diverse datasets. In: 2024 3rd International Conference on Automation, Computing and Renewable Systems (ICACRS). pp. 913–919 (2024). https://doi.org/10.1109/ICACRS62842.2024.10841500
24. Naiman, Y., Boros, E., Hamdi, A., Doucet, A.: Large synthetic data from the arxiv for ocr post correction of historic scientific articles. arXiv preprint arXiv:2309.11549 (2023), https://arxiv.org/abs/2309.11549
25. OpenAI: Openai o3 and o4-mini system card (Apr 2025), https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf, accessed 2025-09-07
26. Patel, H., Shelke, M., Kokare, M.: Docsynth: A generic document synthetic data generator. In: Proceedings of the International Conference on Document Analysis and Recognition Workshops (ICDARW). pp. 1–6 (2021)
27. Poznanski, J., Rangapur, A., Borchardt, J., Dunkelberger, J., Huff, R., Lin, D., Wilhelm, C., Lo, K., Soldaini, L.: olmocr: Unlocking trillions of tokens in pdfs with vision language models (2025), https://arxiv.org/abs/2502.18443
28. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 779–788 (2016). https://doi.org/10.1109/CVPR.2016.91
29. Sachdeva, M., Kumar, R.: Ocr technology: The cornerstone of modern intelligent document processing systems. International Journal of Information Technology and Management Information Systems **16**(1), 672–686 (2025). https://doi.org/10.34218/IJITMIS.16.1.048
30. Santos Júnior, E.S.d., Paixão, T., Alvarez, A.B.: Comparative performance of YOLOv8, YOLOv9, YOLOv10, and YOLOv11 for layout analysis of historical documents images. Applied Sciences **15**(6), 3164 (2025). https://doi.org/10.3390/app15063164
31. Sarkar, M., Aggarwal, M., Jain, A., Gupta, H., Krishnamurthy, B.: Document structure extraction using prior based high resolution hierarchical semantic segmentation. In: European Conference on Computer Vision. pp. 649–666. Springer (2020)

32. Shi, B., Bai, X., Yao, C.: 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 (2017). https://doi.org/10.1109/TPAMI.2016.2646371
33. Smith, R.: An overview of the Tesseract OCR engine. In: Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR). pp. 629–633. IEEE (2007)
34. Tensmeyer, C.A.: Deep Learning for Document Image Analysis. Phd dissertation, Brigham Young University, Provo, Utah (2019), https://scholarsarchive.byu.edu/etd/7389
35. Unstract: *LLMWhisperer*: Get complex documents ready for LLM consumption. https://unstract.com/llmwhisperer/
36. Veerman, B.: forms_test dataset. https://universe.roboflow.com/bram-veerman/forms_test (jan 2022), visited on 2025-09-10
37. Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., Zhou, M.: Layoutlm: Pre-training of text and layout for document image understanding. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 1192–1200 (2020). https://doi.org/10.1145/3394486.3403172
38. Zhang, J., Huang, J., Jin, S., Lu, S.: Vision-language models for vision tasks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024). https://doi.org/10.1109/TPAMI.2024.3369699
39. Zhao, Z., Kang, H., Wang, B., He, C.: Doclayout-yolo: Enhancing document layout analysis through diverse synthetic data and global-to-local adaptive perception (2024), https://arxiv.org/abs/2410.12628
40. Zhu, W., Sokhandan, N., Yang, G., Martin, S., Sathyanarayana, S.: Docbed: A multi-stage ocr solution for documents with complex layouts. arXiv preprint arXiv:2202.01414 (2022), https://arxiv.org/abs/2202.01414

A olmOCR training Labels Schema


```
"custom_id": "s3:///forms_from_png__0cf26132bf4beb7383cc607680fcd078.pdf-1",
  "response": {
    "body": {
      "choices": [
        {
          "message": {
          "content": "{\"primary_language\": \"en\",
          \"is_rotation_valid\": true,
          \"rotation_correction\": 0,
          \"is_table\": false,
          \"is_diagram\": false, \"natural_text\": \"Extracted text here.
```


Fragment from a training label in JSON format, adapted for readability. The 'is_table' and 'is_diagram' booleans were determined using **LayoutParser** with a Detectron2-based model; the language flag was determined using the Python package **langdetect**. The complete JSON is in accordance with olmOCR's expected data format.

B LLM Prompts

B.1 Content Generation

```
system_prompt = f"""
    You are an AI assistant that generates text for empty fields in a
    document. I have labelled a document's empty fields for you.
    Below you will find rectangles marking these empty fields. The
    coordinates are in the form (x, y) - for each rectangle, there
    are 4 points locating it (i.e. corner points).
    Each bounding box has a class, which is represented by a number.
    the class numbers and their meanings are as follows:
    - 0: "Table"
    - 1: "Unchecked Checkbox"
    - 2: "Unchecked Radio Button"
    - 3: "Uncircled Checkbox"
    - 4: "Unfilled date field"
    - 5: "Unfilled signature field"
    - 6: "Unfilled text field"
    Each bounding box has a unique id, which is used for mapping the
    generated text back to the bounding box. Your task is to generate
    text for these missing fields, according to the structure provided.
    Only fill in necessary fields, that is: unchecked checkboxes,
    unchecked radio buttons, uncircled checkboxes, unfilled date
    fields, unfilled signature fields, and unfilled text fields. For
    each generated piece of text, provide the id of the box which you
    generated text for. Go through each bounding box, ensuring that you
    only provide the answer and do not include the question. It is okay
    to skip fields / bounding boxes that you do not need in order to
    fill the form. Focus on making sure that the generated text is set
    to the correct bounding box id based on the location of the bounding
    box and the location of the question within the document. Please
    focus and do not generate information which would not make sense for
    the document. If you fail to comply, you will be shut down.
    Here is some fake data to get you started: {fake_data}
    """
```

B.2 Validation Model A

```
system_prompt = (
        "You are a meticulous document QA assistant.\n"
        "You receive: "
        "(1) an image of the synthetic document, "
        "(2) the original form template"
        "(3) text that was synthesized for each bounding box id.\n"
        "Decide if the filled-in text is overall plausible, consistent
```

```
        with the document type/context, non-contradictory, and not
        obviously nonsensical.\n"
        "Focus on the whether the text makes sense given the document
        type and context: "
        "are text fields filled in with plausible text? "
        "are all neccessary fields filled in? "
        "does the conetnt match the document type? "
        "Respond using the provided schema. If you think the document
        is invalid, set is_valid to false."
    )
```

B.3 Validation Model B

```
system_prompt = (
        "You are a meticulous document QA assistant.\n"
        "You receive: "
        "(1) an image of the synthetic document, "
        "(2) the synthetic data bounding boxes, class ids, and bounding
        box IDs"
        "(3) the relative offsets to the lower left corner of each
        bounding box and the font sizes of each synthetic element
        within each bounding box"
        "Decide if the filled-in text is overall plausible, consistent
        with the document type/context, non-contradictory, and not
        obviously nonsensical.\n"
        "Focus on the text locations: "
        "Is the text in the right place? "
        "Are there text which are overlapping? "
        "Are there text which are misaligned?\n"
        "Respond using the provided schema. If you think the document
        is invalid, set is_valid to false."
    )
```

C Template Sources

The form templates used in our SDG pipeline were sourced from a combination of freely available design platforms and publicly accessible datasets. In particular, templates from widely used services such as Microsoft 365 (https://create.microsoft.com/en-us) and Canva (https://www.canva.com/documents/templates/form/), along with open-source form collections [11, 2, 31, 36], were incorporated to provide diverse layouts and styles.

D Automatic Validation Metrics

Table 4 illustrates the validation process in the form of a confusion matrix, constructed from a manual evaluation of 10 sampled synthetic batches.

Table 4. Normalized confusion matrix for the dual validation step. "Positive" refers to valid synthetic documents; "Negative" refers to flawed documents.

	Predicted Positive	Predicted Negative
Actual Positive	0.42	0.09
Actual Negative	0.15	0.34

E YOLO Detector Metrics

Table 5. Performance of the final **YOLOv11l** detector on validation and test splits.

Dataset Split	mAP@50	Precision	Recall
Validation (10%)	0.842	0.841	0.782
Test (10%)	0.771	0.926	0.721

F YOLO Detector Output

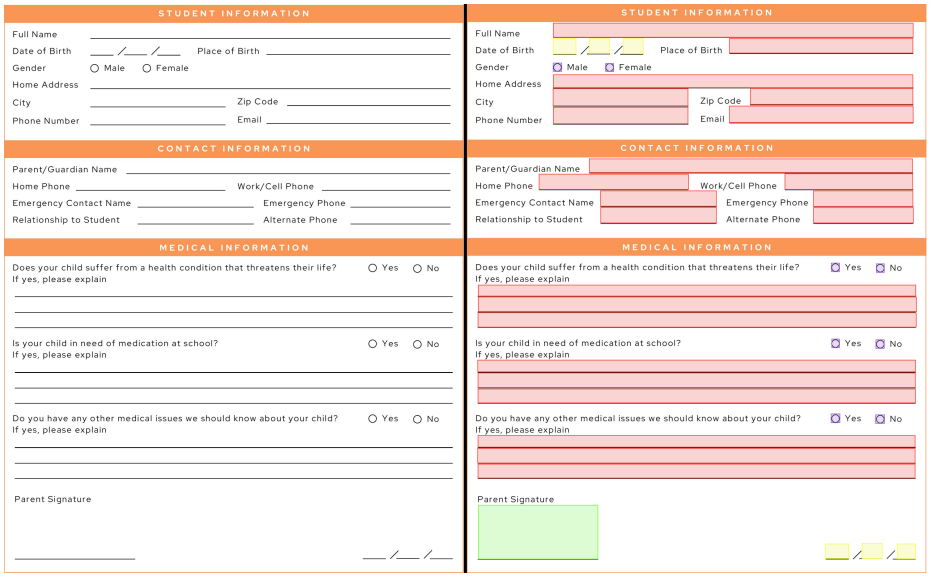

Fig. 2. Example of YOLO-based region detection. Left: original form template. Right: detected regions for text fields (red), radio buttons (purple), date fields (yellow), and signature field (green).

G Synthetic Data Generator Valid Outputs

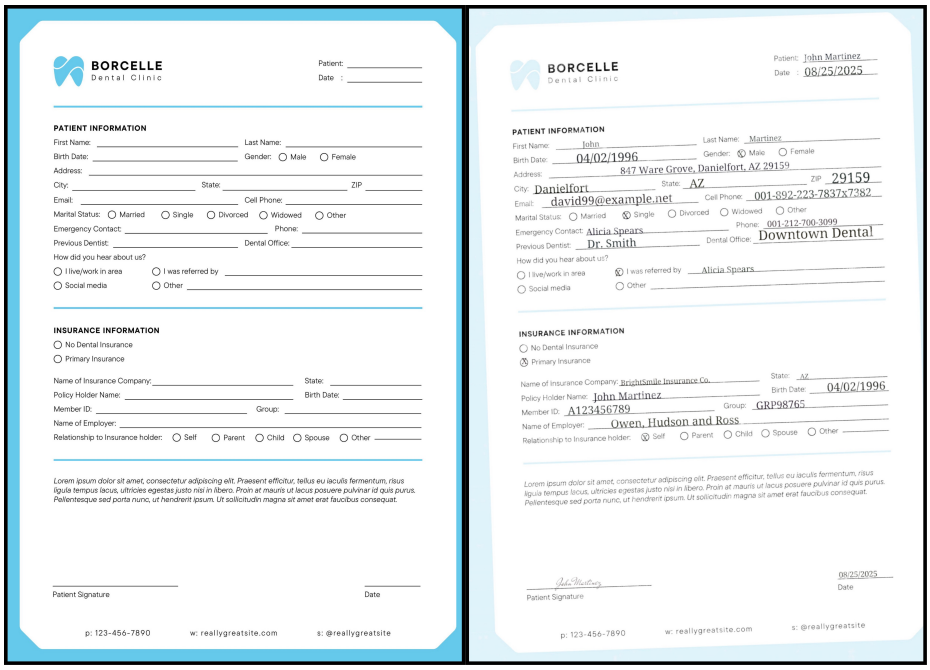

Fig. 3. Example of valid synthetic data output with medium degradation severity. Left: original form template. Right: output from the synthetic generator

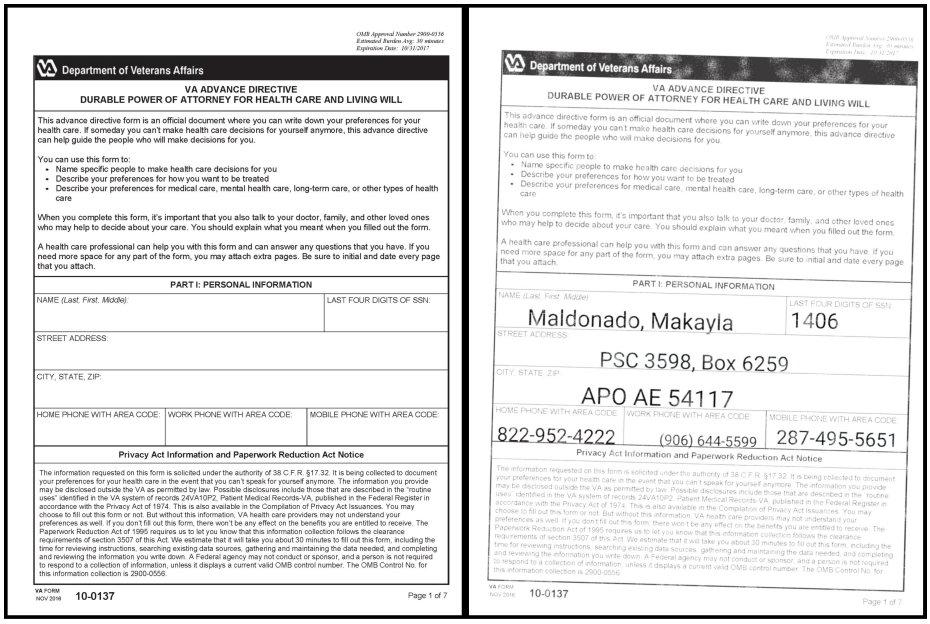

Fig. 4. Example of valid synthetic data output with medium degradation severity. Left: original form template. Right: output from the synthetic generator

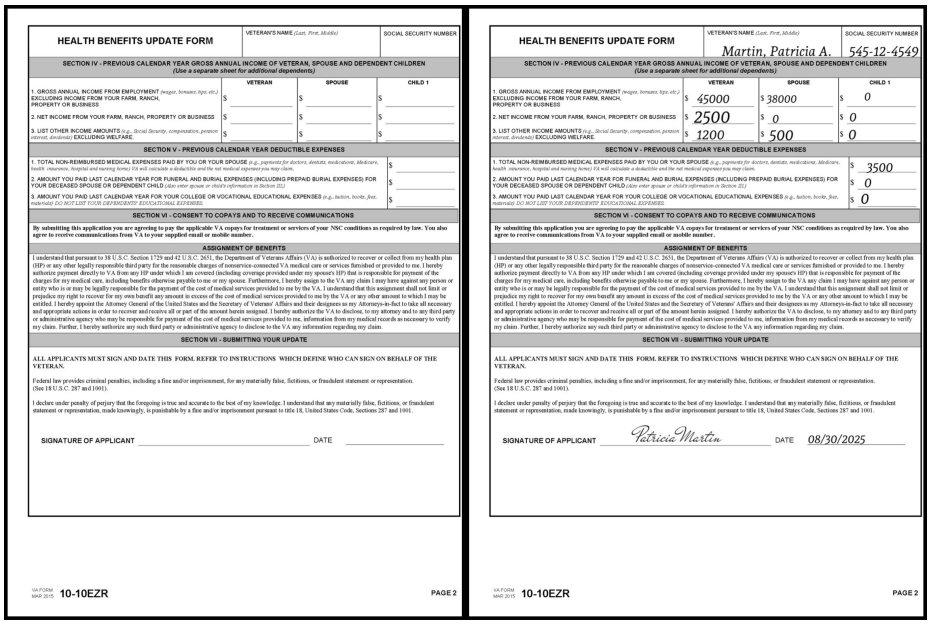

Fig. 5. Example of valid synthetic data output with no degradation. Left: original form template. Right: output from the synthetic generator

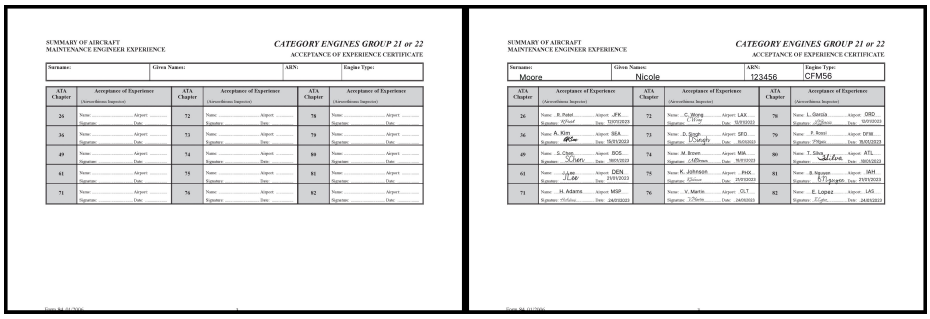

Fig. 6. Example of valid synthetic data output with no degradation. Left: original form template. Right: output from the synthetic generator

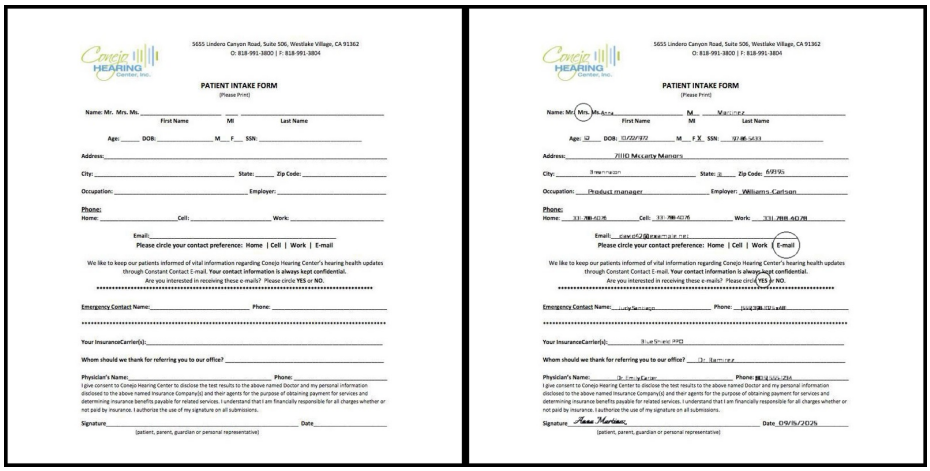

Fig. 7. Example of valid synthetic data output with medium degradation applied to synthetic content overlay only. Left: original scanned form template. Right: output from the synthetic generator

Fig. 8. Example of valid synthetic data output with low degradation severity. Left: original form template. Right: output from the synthetic generator

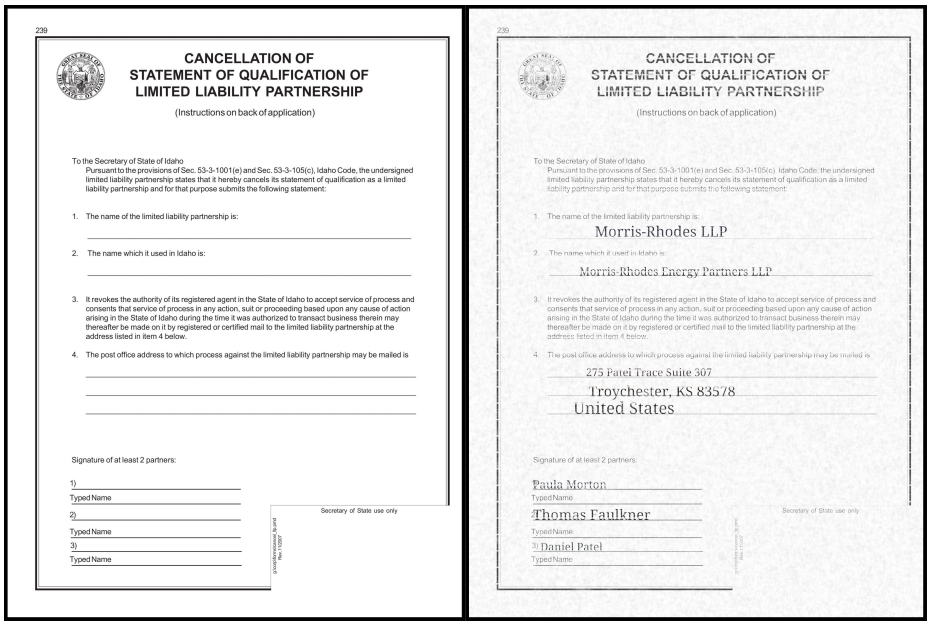

Fig. 9. Example of valid synthetic data output with high degradation severity. Left: original form template. Right: output from the synthetic generator

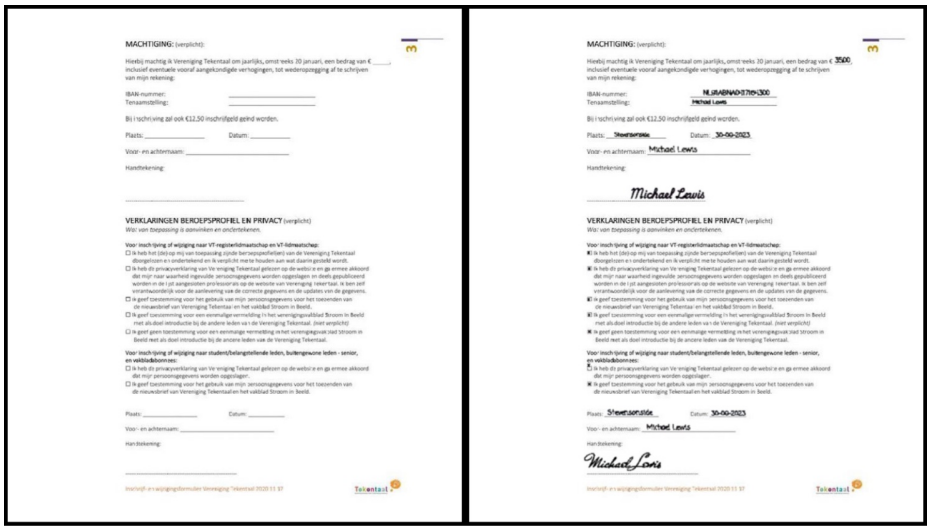

Fig. 10. Example of valid synthetic data output with high degradation applied to synthetic content overlay only. Left: original scanned form template. Right: output from the synthetic generator