A document is worth a structured record: Principled inductive bias design for document recognition

Benjamin Meyer[1,3*], Lukas Tuggener[1,5], Sascha Hänzi[2], Daniel Schmid[2], Erdal Ayfer[1], Benjamin F. Grewe[3], Ahmed Abdulkadir[1†], Thilo Stadelmann[1,4†]

[1*]Centre for AI, Zurich University of Applied Sciences, Winterthur, Switzerland.
[2]Institute of Product Development and Production Technologies, Zurich University of Applied Sciences, Winterthur, Switzerland.
[3]Institute of Neuroinformatics, University of Zurich and ETH Zurich, Zurich, Switzerland.
[4]European Centre for Living Technology, Venice, Italy.
[5]RWAI Schweiz AG, Winterthur, Switzerland.

*Corresponding author(s). E-mail(s): mebr@zhaw.ch;
Contributing authors: lukas.tuggener@rwai.ch; haez@zhaw.ch; scdd@zhaw.ch; ayfererd@students.zhaw.ch; bgrewe@ethz.ch; abdk@zhaw.ch; stdm@zhaw.ch;
†These authors contributed equally to this work.

Keywords: End-to-end document transcription, learnable symbol assembly, engineering drawing recognition, document foundation models

Many document types use intrinsic, convention-driven structures that serve to encode precise and structured information, such as the conventions governing engineering drawings. However, many state-of-the-art approaches treat document recognition as a mere computer vision problem, neglecting these underlying document-type-specific structural properties, making them dependent on sub-optimal heuristic post-processing and rendering many less frequent or more complicated document types inaccessible to modern document recognition. We suggest a novel perspective that frames document recognition as a transcription task from a document to a record. This implies a natural grouping of documents based on the intrinsic structure inherent in their transcription, where related document types can be treated (and learned) similarly. We propose a method to design structure-specific relational inductive biases for the underlying machine-learned end-to-end document recognition systems, and a respective base transformer architecture that we successfully adapt to different structures. We demonstrate the effectiveness of the so-found inductive biases in extensive experiments with progressively complex record structures from monophonic sheet music, shape drawings, and simplified engineering drawings. By integrating an inductive bias for unrestricted graph structures, we train the first-ever successful end-to-end model to transcribe mechanical engineering drawings to their inherently interlinked information. Our approach is relevant to inform the design of document recognition systems for document types that are less

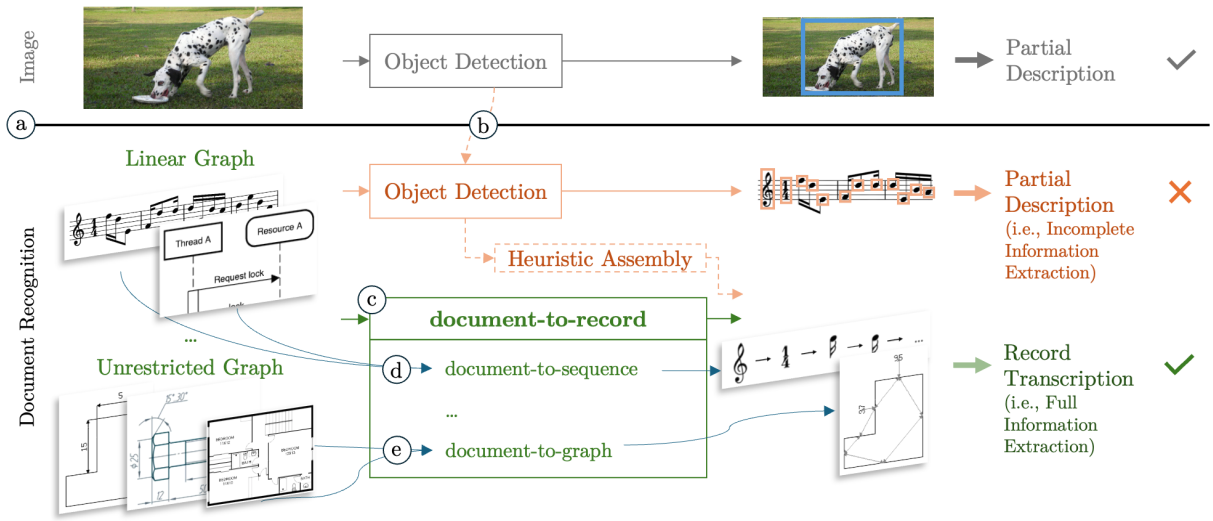


Fig. 1: Document recognition and natural image analysis are often approached with similar methods, however: (a) Image analysis is about extracting partial descriptions of complex and ambiguous content (e.g., detecting the "dog" object doesn't account for the shadows/dog breed/... in the image). In contrast, document recognition is about the complete transcription of explicitly encoded information. (b) Viewing document recognition merely as a computer vision task results in incomplete extraction, failing to fully capture all the information a document is intended to convey. (c) We propose the *document-to-record transcription* task, formulating document recognition as the extraction of complete information (i.e., a record). The record structure is specific to each document type, but archetypes of record structures exist that span multiple document types. Thus formed document classes give rise to a principled way of designing inductive biases for the underlying learning systems, e.g.: (d) Notes in monophonic music or exchanged messages in a sequence diagram form linear graphs, implying the use of a structure-specific inductive bias for *document-to-sequence* transcription; (e) more complex record types, such as for engineering drawings or floor plans, form unrestricted graphs and require *document-to-graph* transcription.

well understood than standard OCR, OMR, etc., and serves as a guide to unify the design of future document foundation models.

1 Introduction

Deep learning-based computer vision systems have become increasingly powerful in processing natural images [1, 2], which has led to their application in various types of digital document recognition tasks that process pixel-based representations of said documents [3]. While this adoption has proven successful [4–8], there are many highly relevant document types for which success remains limited, such as engineering drawings [9], floor plans [10], and others [11–13], despite such documents being easily understood by domain experts. We postulate that overcoming existing performance gaps requires a shift in perspective on automated document analysis that formulates the analysis task differently than an isolated computer vision task (e.g., object detection), and instead as transcription (cp. Fig. 1). This new perspective implies that specific inductive biases can be designed that reflect the intrinsic structures found in broad classes of documents. The remainder of this section will establish this perspective, starting from the status quo.

Many of the methods employed in both natural image analysis and document recognition were first developed for natural image datasets [14–16] and subsequently adapted for use in document analysis tasks (e.g., [17, 18]). Those methods, by design, align with natural images by extracting *partial descriptions*, necessitated by the granular, rich, and dense image content: While extracting *everything* contained in an image is usually infeasible and never attempted, different descriptions

enable different downstream tasks. For example, object detection may identify and locate a set of predefined common objects, making downstream tasks possible such as object counting [19], object tracking [20], or robotic grasping [21] (Fig. 1 (a)), but might not suffice to enable image localization [22]. However, those partial-description methods do not naturally align with documents. Documents encode specific (and comparatively little) information that they were explicitly designed to convey, and analyzing documents is about *precisely extracting* this information *fully*. Enforcing a partial description, e.g., through object detection to recognize symbols [10], results in incomplete information extraction, as, for example, the essential relationships between symbols are ignored by a pure object detection method (Fig. 1 (b)). Of course, such a partial extraction can be (and in practice usually is) amended by individual, heuristic post-processing to recover, e.g., said object relationships. But a more principled, document-centric approach seems possible.

For this, we first introduce necessary terminology (cf. Section 3.2 for formal definitions): We refer to the information encoded in a document as its *record*. This record is both the source and the transcription of a document, includes all information, and thus forms a natural interface to execute downstream tasks. It does not contain visual attributes resulting from visually representing the record as a document, such as stylistic choices (e.g., line thickness, color, font) and non-semantic layout options (e.g., line or symbol spacing). This record-document duality gives rise to viewing document understanding as an end-to-end, *document-to-record transcription* task (Fig. 1 (c)).

The exact schema of a record is document-type-specific, but, in general, it contains interlinked data, most generally represented by a property graph. We refer to the nodes in that graph as *record nodes*, which have a node type and type-specific properties. Record nodes find their correspondence in the respective document in primitive shapes and symbols. Relationships (edges) between nodes are typically visualized by clues such as intelligible spacing and additional primitives like arrows (cf. Fig. 2).

It is interesting to consider the overall structure of the record (henceforth called the *record structure*): For example, notes in monophonic sheet music form a linear graph (or: a sequence) where nodes are connected in a staff, having only one predecessor and one successor – a sequential record structure, as with standard text. Record nodes in the record of mathematical expressions form ordered trees, in which nodes may have nested child nodes, but children form an ordered sequence – a hierarchical sequential structure. Document types with less rigid content order form general graphs where nodes have interlinked relations – engineering drawings fall into this category, containing primitive shapes and annotations that, through visual cues, are interlinked in complex ways to form a discrete 3-dimensional shape with manufacturing annotations. Floor plans are another example.

Interestingly, record structures thus naturally group otherwise unrelated document types together, e.g., monophonic sheet music and plain text by means of their sequential record structure. This warrants the specialization of document-to-record into tasks such as document-to-sequence, document-to-set, and document-to-graph (others exist that haven't been exemplified above, e.g., document-to-tree for diagrams like mind maps or organizational charts). Each possible specialization makes different assumptions about the restrictions present in the record structure, mapping to known types of graphs from graph theory [23, Chapter 2.3] (Fig. 1 (d) and (e)).

This structural grouping explains the performance gap in document recognition between certain document types: Recent successes in end-to-end methods are limited to documents with an inherently sequential record structure, such as in OCR (Optical Character Recognition, e.g, [6]) and OMR (Optical Music Recognition, which is sequential in the sequence of notes when processing polyphony and multiple staves in a predefined order, e.g. [4]). These methods address the document-to-sequence problem by adopting sequence-to-sequence architectures designed with a sequential bias in their next-token generation [24]. In contrast, most document types currently lacking complete information extraction solutions carry an inherently non-sequential, more general record structure, requiring the extraction of a graph structure, where common methods lack adequate inductive biases.

This analysis also highlights a blind spot in attempts to build foundation models for documents such as those in [8, 25–27]: Present approaches exclusively employ sequence-to-sequence methods and hence focus on document types that carry a mostly sequential record structure. Our perspective offers a path to incorporating suitable inductive biases for non-sequential document types into such models by adapting them to more general structures or by motivating the normalization of training targets to align with current inductive biases, e.g., through heuristic graph linearization strategies [28].

Strictly speaking, the above does not hold for all types of documents (e.g., not for comic books or line art). In this work, we instead focus on documents designed for *precise* information exchange (see Fig. 2 for examples). We term such documents *convention-bound* as they follow strict conventions that govern the contained information and emerged to ensure unequivocal information exchange among domain experts (see Section 3.2 for a formal definition of convention-bound documents). These include sheet music or engineering drawings, but exclude imprecise documents like (ambiguous) illustrative diagrams. Due to their non-ambiguity, the document-to-record framing is well-posed for convention-bound documents.

From this consideration, we derive our main conceptual advance: Document recognition on convention-bound documents is best seen as document-to-record transcription, and benefits from incorporating the intrinsic record structure as an inductive bias in the model's architecture and training process. This enables end-to-end learning for more rare and less-understood document types, particularly those with a non-sequential nature.

In this paper, our contributions are: (1) *"Document recognition as transcription" perspective:* By viewing document recognition as complete record transcription of convention-bound documents rather than computer-vision-based extraction of partial descriptions, a meaningful and efficient grouping of document types by record structure emerges naturally that explains performance gaps in state-of-the-art approaches; it is later leveraged for learning. (2) *Method for systematically designing inductive biases for document recognition systems:* Utilizing said record structures in a principled, systematic way leads to a novel approach in designing document-centric model architectures applicable across document types. This establishes a bridge from document-to-record transcription to methods for graph prediction. (3) *Implementation of a practical respective machine learning framework:* We introduce an end-to-end, domain-agnostic learning framework based on a unified transformer backbone with document-group-specific adapted inductive biases, useful for sample-efficient document foundation models. (4) *Comprehensive experimental evaluation of proofs of concept:* The postulated principles are valid and effective in monophonic sheet music (document-to-sequence), shape drawings (document-to-set), and simplified engineering drawings (document-to-graph). For mechanical engineering drawings, we show the first ever working implementation of an end-to-end learned document recognition approach.

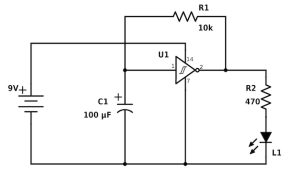
2 Related Work

2.1 Information extraction from convention-bound documents

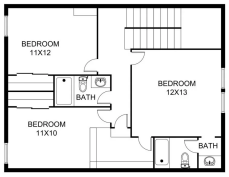
Document recognition of convention-bound documents can be viewed as information extraction. Solutions include:

Incomplete information extraction. Computer vision methods such as object detection (symbol spotting) and semantic segmentation have been widely applied to convention-bound documents, including newspaper pages [5], sheet music [17, 29, 30], mathematical expressions [31], engineering drawings [9], circuit diagrams [11], floor plans [10, 32, 33], UML diagrams [12], and P&ID diagrams [13]. However, these approaches are inherently incomplete as they typically detect individual symbols or regions but fail to capture the semantic relationships or structure between symbols directly, which is essential for fully understanding and utilizing such documents.

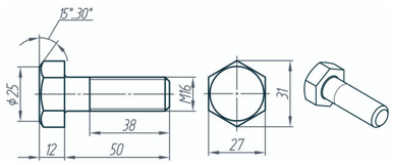
Heuristic full information extraction. Full information extraction (i.e., transcription) involves not just extracting relevant symbols and shapes but also their corresponding relationships. To achieve this, incomplete information extraction methods have been extended by post-hoc heuristic symbol assembly steps, for example, in UML diagrams [34], or sheet music [29].



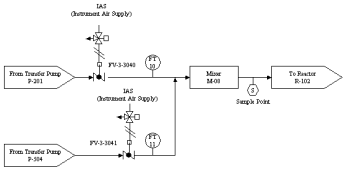
(a) *Electric circuit diagrams:* A graph of electronic symbols (e.g., diodes, resistors) with properties (e.g., resistance value).



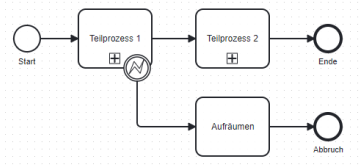
(b) *Floor plans:* A graph of shapes (e.g., walls, doors, windows, stairs, fixtures) with lengths to form and dimensionalize rooms.



(c) *Engineering drawings:* A graph of shapes (e.g., lines) and manufacturing information (e.g., dimensions, tolerances).



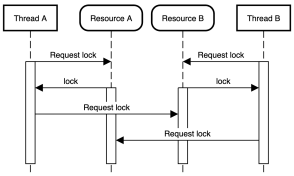
(d) *P&ID diagrams:* A directed graph of primitives (e.g., valves, tanks, pumps) with properties (e.g., numerator, control, device).



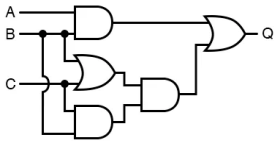
(e) *Business Process Models and Notation:* A directed graph of business process elements (e.g., activities, events, gateways).

$$\frac{df}{dt} = \lim_{h \to 0} \frac{f(t+h) - f(t)}{h}$$

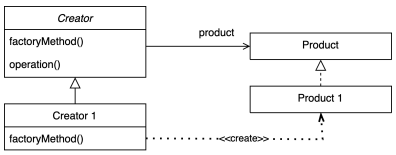
(f) *Mathematical expressions:* An ordered tree of mathematical primitives (e.g., fractions, variables, and numbers).



(g) *Sequence diagrams:* A sequence of text messages with a source and target entity.



(h) *Logic circuit diagrams:* A graph of logic gates (e.g., `AND` gate, `OR` gate) connected by wiring.



(i) *Class diagrams:* A graph of classes with, e.g., names, methods, and relationships (e.g., inheritance).

Fig. 2: Examples of convention-bound documents with brief descriptions of the contained information (record) and its structure (record structure). Document types with similar record structures can be analysed with a unified document recognition system under the perspective of our transcription framework.

Narrow end-to-end information extraction. Many methods employ sequence-to-sequence approaches to learn robust record extraction end-to-end for *single* types of documents. These records follow either an explicit sequential structure, as in text recognition [6], or an implicit sequential structure inherent to ordered tree structures through their universal address system [35, chapter 11.3], as in mathematical expressions [7] and visual language understanding [36]. The authors of [4] apply end-to-end information extraction to general sheet music, where their record representation follows an (approximately) sequential structure. Moving beyond sequential biases, a few recent approaches have begun to apply image-to-graph architectures [37, 38] to directly extract nonlinear relational structures from technical drawings, such as general graphs from piping and instrumentation diagrams (P&IDs) [39] and simplified geometric graphs from floor plans [40].

Broad end-to-end information extraction. Recent works also include early attempts at building document foundation models [41], which can extract text, lists, tables, equations, music notation, and charts from entire documents [8, 25–27] or parts of documents [42, 43]. Those models follow the natural reading order, assuming an approximately sequential record structure, explaining the success of adapting sequence-to-sequence methods to respective document types. Scaling such models to less sequential document

types would require either massive amounts of training data (with massive compute due to a noisy training signal) to learn the respective structure from the data alone [44, 45], or else need to be equipped with appropriate but still broad inductive biases to be efficient [46].

Our classification of document types according to their intrinsic record structure reveals a research bias towards documents with inherently sequential organization in the current literature, where document-to-sequence methods are commonly used. In contrast, complete information extraction remains lacking for inherently non-sequential document types. By shifting the research perspective and introducing a principled approach for developing more comprehensive models, supported by the experimental evidence presented below, we aim to address the uncharted territory left by recent methods, enabling data-efficient document foundation models for a broader array of document types.

2.2 Relational inductive biases

The concept of relational inductive biases [47] provides a framework for understanding how neural architectures exploit dependencies in data. Historically, this perspective was driven by intuitive architectural designs tailored to inherent data structures (e.g, CNNs for grids, DeepSets for sets). More recently, these architectural biases have been formalized, notably through group theory for symmetric relations [48] and category theory for a broader mathematical foundation [49, 50]. While these frameworks primarily focus on the processing of relational *inputs*, document-to-record transcription extends this challenge to predicting structured (relational) *outputs* from rasterized images (in which relations are conveyed visually). As the concern for appropriate relational inductive biases is a shared one, we now review related literature and position the proposed sequence, set, and graph prediction methods within this broader research landscape.

Graph generation. A common strategy for learning graph distributions is to linearize graphs into flat sequences. However, because arbitrary graphs lack a natural sequential order, this process yields up to $(n+m)!$ valid permutations for a single graph with n nodes and m edges. Naively training on these permutations forces the model to learn a highly diluted distribution. To mitigate this exponential explosion, methods typically restrict the permutation space using heuristics [51, 52] or by learning custom linearization strategies [53]. Still, such heuristics must be applied carefully, as sequence ordering impacts learning dynamics [54]. Alternatively, diffusion-based methods avoid imposing sequential structures by iteratively editing noisy graphs to match a target distribution—an approach that enables the use of permutation-equivariant architectures [55].

In contrast to graph generation, document transcription is a deterministic, many-to-one graph prediction task, as will be formalized in Equation (1). This bypasses any need to learn distributions over graphs: While many-to-one prediction can be framed and potentially learned as a conditional Dirac target distribution, we suspect the complex distributional modeling used in graph generation is unhelpful for document transcription. Instead, our graph prediction bias design will incorporate teacher forcing based on a node-first linearization, yielding a permutation space of $n!m!$ valid sequences (see Section 3.4). The results will show that despite this high variance in the decoder inputs during training, the decoder learns the transcription task by successfully following its remaining-node order during inference (see Section 4.6), and future work will investigate if more strictly enforcing the model's dynamically learned remaining-node order during training leads to more robust transcription capabilities.

Sequence prediction. Left-to-right autoregressive sequence prediction [24] has achieved immense success [56, 57], particularly via causal self-attention. Building on this foundation, we adapt a standard causal (sequence) transformer to support more diverse structural biases through targeted modifications to only the attention mask and loss function.

Our proposed sequence prediction aligns very well with the literature (see Section 3.5).

Set prediction. DETR [58] reformulates object detection as a set prediction problem, implemented using learnable "object queries" and a set loss based on bipartite matching between predictions and ground truth. While the methodology is applicable to general set prediction, DETR and many subsequent works [59–61] focus on

object detection and use domain-specific terminology (e.g., "object queries" rather than general "element queries"). To align with this established literature, we adopt this terminology here. Recent related approaches include Slot Attention [62] and Set Transformer [63]; both use a form of cross-attention to bottleneck a set of inputs into a pooled set of representations, referred to as "slots" or "seed vectors", that are functionally analogous to learned object queries.

Learnable object queries have two known issues: First, they converge slowly due to the instability of bipartite graph matching in early training stages [60]. Second, they require a number of queries significantly larger than the maximum number of objects in an image, incurring substantial computational overhead during both training and inference [58]. For example, DINO [61] uses 900 queries on the COCO dataset, which contains only up to 80 objects per image. Recent works have attempted to address these limitations. Convergence instability has been mitigated by injecting noisy ground truth into the object queries [60], a technique that functions as a surrogate for teacher forcing by providing the model with ground-truth hints that accelerate convergence. Additionally, dynamic queries have been introduced to prune the set of object queries throughout the forward pass, reducing the computational footprint of deeper decoder layers [64].

While these are working improvements, we propose a more fundamental rethinking of set prediction by replacing bipartite matching entirely with autoregressive remaining-node prediction. Exhibiting architectural similarity to XLNet [65], our approach will natively enable teacher forcing (see Section 3.6). Furthermore, this formulation will require exactly as many prediction queries as there are objects per sample, providing a strictly minimal query count.

Graph prediction. Several works extend the set prediction paradigm to graph prediction by incorporating a relation prediction module. Much of this research focuses on the domain of scene graph generation [37, 38, 66–69], embedding domain-specific terminology (e.g., framing graphs as "relations" or "interactions" between "subjects" and "objects"). To align with this established literature, we adopt this terminology here. HOTR [67] utilizes object and interaction queries as two parallel set predictions, deriving multi-label interactions from similarity scores between object representations and projected interaction representations. RelTR [37] detects subjects and objects using shared entity representations, which are then used to predict directed relations. Relationformer [38] introduces a distinct relationship query encoded together with the learnable object queries. For each contextualized object query pair and the contextualized relationship query, a relationship (or absence thereof) is predicted independently. This method has been applied to document transcription [39].

In contrast, our graph prediction method simply models the graph as a set of nodes and edges, reducing graph prediction to a set prediction. Compared to existing methods that predict relations independently for each object pair, our autoregressive design automatically enables the model to account for the emerging graph topology during prediction.

3 Methods

3.1 Overview

We first formalize the concept of convention-bound documents in Section 3.2 and establish that document-to-record transcription is well-posed for this class of documents. We then introduce a general learning framework for document-to-record transcription in Section 3.3, accompanied by a foundational architectural design in Section 3.4. Subsequently, we specialize this architecture for document-to-sequence, document-to-set, and, more generally, document-to-graph tasks in Sections 3.5 to 3.7; each of which is exemplified with an example use case in the next section.

3.2 Convention-bound documents

Let $\mathcal{R}$ be the set of records and $\mathcal{D}$ the set of documents for a specific document domain. A record $r \in \mathcal{R}$ is a property graph, defined as a set of nodes related by a set of edges [71]. Specifically, we represent each record as interlinked *(record) nodes*, with each node having a concrete type and a set of type-specific properties. For example, a music symbol is a record node with a type (e.g., clef or note) and properties (e.g., clef type or note pitch). Graph theory classifies a (property) graph into

certain types based on restrictions present in the graph's relations [23, Chapter 2.3]. Based on these types, we define three *record structures* for this work: The sequential record structure for linear graphs and ordered trees (e.g., for sheet music or mathematical expressions), the set record structure for empty graphs (e.g., for shape drawings), and the graph record structure for unrestricted graphs (e.g., for engineering drawings).

To share the information contained in a record $r \in \mathcal{R}$ through a document $d \in \mathcal{D}$, a *write function* $f : \mathcal{R} \to \mathcal{D}$ first encodes the record into a document (this can also be a human creating the document). A *read function* $g : \mathcal{D} \to \mathcal{R}$ decodes the document back to the record (can be a human reading the document). We call the set of all valid write functions for this document domain its *write function space* $\mathcal{F} \in \mathcal{P}(\mathcal{R} \to \mathcal{D})$ and the set of all valid read functions its *read function space* $\mathcal{G} \in \mathcal{P}(\mathcal{D} \to \mathcal{R})$, where $\mathcal{P}(\mathcal{S})$ denotes the power set of S.

To make the exchange of records via documents precise and efficient, domain-specific conventions exist, such as norms and standards. These conventions introduce restrictions to (i) what can be expressed (restricting $\mathcal{R}$), (ii) how information must be encoded (restricting the write function space $\mathcal{F}$), and (iii) how information must be decoded (restricting the read function space $\mathcal{G}$), such that records can always be exchanged losslessly via documents, meaning the following equation holds:

$$\forall r \in \mathcal{R}, \forall f \in \mathcal{F}, \forall g \in \mathcal{G} : (g \circ f)(r) = r \qquad (1)$$

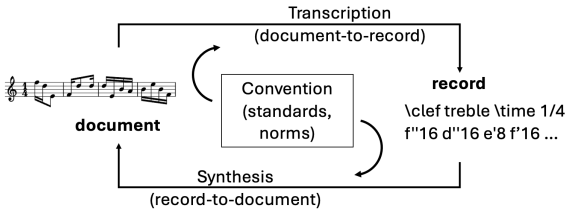


Fig. 3: Transcription and synthesis on the example of monophonic sheet music. The record is here represented with LilyPond code [70].

For example, a piece of sheet music can only express what is part of musical notation (restricting $\mathcal{R}$), and standards like the modern staff notation exist to make the exchange lossless for all expressible music pieces (restricting $\mathcal{F}$ and $\mathcal{G}$).

We call documents from domains for which such domain-specific restrictions apply *convention-bound documents*. From Equation (1) it follows that any convention-bound document $d \in \mathcal{D}$ has only a single, valid reading $r = g^*(d)$ (otherwise Equation (1) would be violated for the respective records, see Section A for the full proof). Thus, the read function space for d reduces to a single function $\mathcal{G} = \{g^*\}$ in the case of d being a convention-bound document. This implies that reading convention-bound documents is a many-to-one problem, which permits the use of function approximation techniques that directly estimate the single valid output, rather than having to approximate the (multi-modal) probability distribution over many possible outcomes as in many-to-many problems [72]. Importantly, there can still exist many write functions, as any write function $f \in \mathcal{F}$ is valid that encodes the complete record r into the document d in an unambiguously extractable way given g^*.

Thus, many valid convention-bound documents $\mathcal{F}(r) = \{f(r) \mid f \in \mathcal{F}\}$ exist for one record, but any of these documents is derived from the same single record, the one causing it. We hence call the process implemented by any $f \in \mathcal{F}$ *synthesis* and the process implemented by the single read function g^* *transcription* (cp. Fig. 3). This terminology is analogous to speech synthesis and transcription: Synthesis indicates the addition of representational variation that does not alter the information content – there it is a speaker's voice, here it is visual attributes in documents (e.g., stylistic choices like font, line thickness, or semantically neutral symbol placement choices). Meanwhile, transcription indicates that we ignore these attributes and only extract the core information, the record r.

3.3 Document-to-record transcription

To put our perspective into practice, we assume the following components in a document recognition as transcription-framework (our concern herein is to machine-learn the transcription

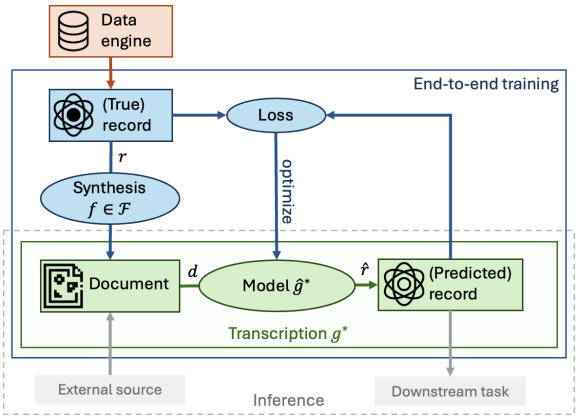


Fig. 4: Overview of the proposed document-to-record transcription framework.

model): A *data engine* provides representations of valid records $r \in R$ as training data for supervised learning, e.g., source code describing the record. A *synthesis function* $f \in \mathcal{F}$ converts such a representation of the record r into a rasterized document $d = f(r)$, i.e., into a document created for human consumption. When synthesizing training data later, we select synthesis functions strategically from a variety of domain-specific rendering routines to ensure visual diversity, leveraging tools originally developed to visualize records for humans. A *transcription model* $\hat{g}^*$ converts the document d into a record prediction $\hat{r} \in \mathcal{R}$. This transcription model is learned end-to-end (document-to-record), approximating the unknown transcription function g^*. The loss is defined in record space $\mathcal{R}$ (cf. JEPA [73, 74]) and measures the dissimilarity between the original record r and its prediction $\hat{r}$. Measuring the dissimilarity between r and $\hat{r}$ provides a stable learning signal as r is unambiguous for a convention-bound document d, meaning any dissimilarity is a mistake of the transcription model (see Section 3.2, Equation (1)).

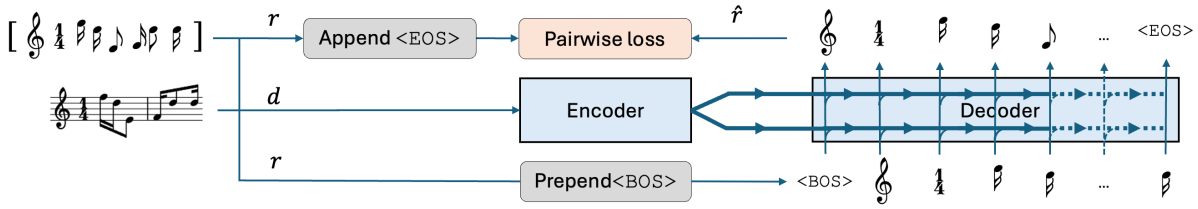
Fig. 4 illustrates the proposed framework. For any downstream task, the trained transcription model outputs records that capture all information in a semantically accessible form, omitting all visual by-products of the document and simplifying subsequent processing. The predicted record can also be synthesized back to a human-friendly document representation via $f(\hat{r})$ for manual transcription validation.

3.4 Base neural net architecture

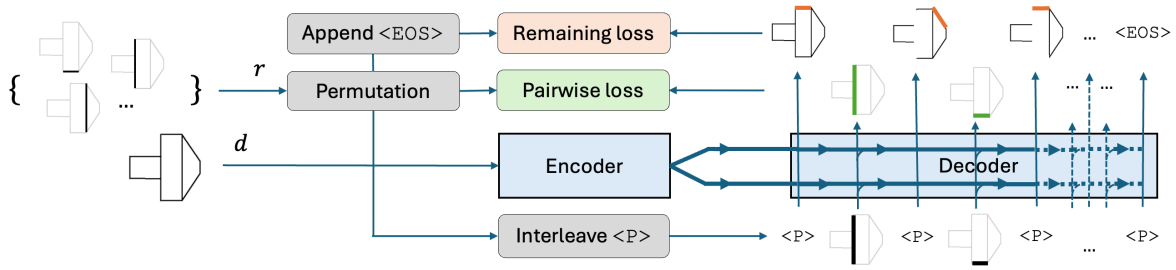
We follow the encoder-decoder paradigm [24, 75, 76], encoding the document d into an intermediate representation to then decode (transcribe) it into the record prediction $\hat{r}$. In our experiments, documents contain exactly one page, represented as a single rasterized image. We make this simplification without loss of generality, as the encoder can be trivially extended to multi-page documents. The encoder setup is unchanged across all experiments. We then add a suitable inductive bias to the decoder and training process based on the record structure as described in the following sections. Each section describes an adaptation in detail (see summary in Fig. 5). All adaptations are domain-agnostic by being transferable to any convention-bound document type with the same inherent record structure as the exemplary document type.

Encoder. We use a standard vision transformer [77] with a minor improvement for digital documents: Similar to the idea of masked autoencoders, we remove uninformative patches that contain only white pixels [78]. This reduces the memory footprint and increases computational efficiency without loss in performance.

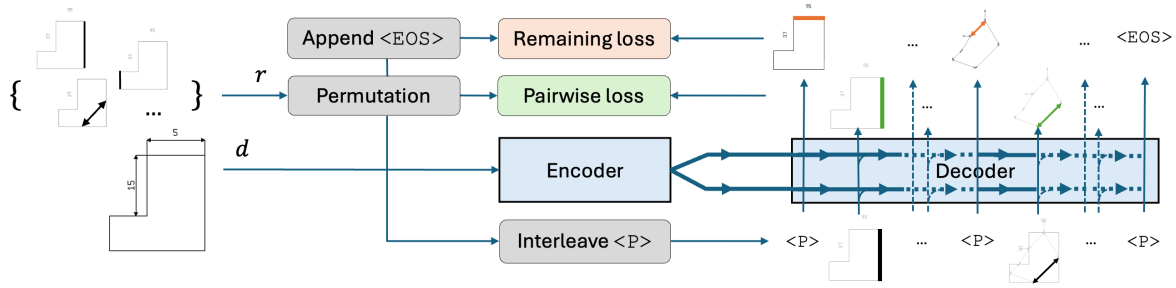
Decoder. We use the transformer architecture [76] for the decoder. Disregarding implementation details like layer normalization and positional encoding, the base transformer is equivalent to a fully connected graph attention network [79] well-suited for learning *a priori* unknown relations [63, 80]. While we introduce causal attention to enable autoregressive generation, the attention over the preceding context remains unconstrained, ensuring the model captures relations without imposing order-dependent priors. To form the decoder inputs, we embed each record node into a *node embedding* by mapping the record node's type and type-specific properties into a single embedding vector. Thus, the decoder operates at the level of node embeddings, which is necessary to enable the straightforward inductive bias adaptations discussed in subsequent sections. After the decoder, output heads predict the type and properties of a record node from each final node embedding.



(a) The training process for next-node prediction in the *document-to-sequence* model (see Section 3.5) implements the standard procedure used in sequence-to-sequence training [24] of teacher forcing [81] by prepending a <BOS> token to the decoder input and appending a <EOS> token to the end of the target sequence. Left: Input is the document (d); the true record (r) serves as ground truth. Right: The decoder predicts, given the current node (bottom), the next node in the sequence (top, thin up-arrows in the decoder) through a pairwise node loss. Thick arrows in the decoder indicate the information flow through attention masking.



(b) The training process for remaining-node prediction in the *document-to-set* model (Section 3.6) employs teacher forcing by generating a random permutation of the nodes. To facilitate this, a <P> token is interleaved within the decoder inputs, and a <EOS> token is appended to the targets (see main text for explanation). Thick arrows in the decoder indicate the information flow through attention masking; no attention on <P> embeddings. The remaining-node loss respects that there is no order in prediction, meaning any non-"taken" node can be predicted. Pairwise loss on "taken" nodes encourages information preservation.



(c) The remaining-node prediction training process for *document-to-graph* (Section 3.7): It is equivalent to the *sequence-to-set* training process in Fig. 5b except it additionally guarantees that relationship nodes occur after record nodes in the permutation step. Relationship nodes are visualized by double arrows that link two record nodes as in Fig. 9c.

Fig. 5: Overview of the core components of the architecture and specific adaptations to the training designed for handling distinct record structures. <BOS>: beginning of record, <EOS>: end of record, <P> prediction token. Note: For visual clarity, mapping from tokens and nodes to embeddings is not shown.

3.5 Document-to-sequence bias

For document types with an inherently sequential record structure, the document-to-record task is reduced to the well-known document-to-sequence task. This section discusses how an inductive bias for sequential structure is built into standard sequence-to-sequence methods, serving as a basis for integrating more complex record structures in later sections.

The seminal transformer paper [76] applied the transformer in a sequence-to-sequence setting (text translation) with the decoder and training process including a sequential inductive bias: *next-token prediction*. We do the equivalent *next-node prediction* for our sequential record structure. Specifically, at each position, the model predicts the next node in the record sequence based on the current and previous node embeddings.

We do this following the common approach: The decoder input is the node sequence prepended with a special *beginning-of-record* token `<BOS>`, and the decoder output target is the node sequence appended with a special *end-of-record* token `<EOS>`. The loss is a node dissimilarity measure (node type and properties) at each position (see Equation (2)). Causal masking, meaning embeddings can never attend to the right, is applied throughout the decoder, preventing trivial label leakage solutions. One-dimensional positional encoding is added to the initial node embeddings to enable the decoder to learn and use the sequential structure of the nodes. Fig. 5a illustrates this training process. Inference happens autoregressively, predicting one next node at a time (see Fig. 6a).

Formally, the sequence bias loss is defined as follows: Let r be a sequence of m nodes $(n_1, ..., n_m)$. Let n_0 be the `<BOS>` token and n_{m+1} be the `<EOS>` token (which, for notational convenience, is treated as a property-less node with unique type). Let $\hat{n}_i = \hat{g}^*(d, n_0, \ldots, n_{i-1})$ be the prediction of the next node n_i based on the document d, the `<BOS>` token n_0 and previous nodes $(n_1, \ldots, n_{i-1})$; $\hat{g}^*$ representing our model. Let $l(n, \hat{n})$ be a dissimilarity measure between a node n and a prediction $\hat{n}$ based on their type and properties (see Section D for definition). Then, the sequence bias loss $\mathcal{L}_{seq}$ is defined as:

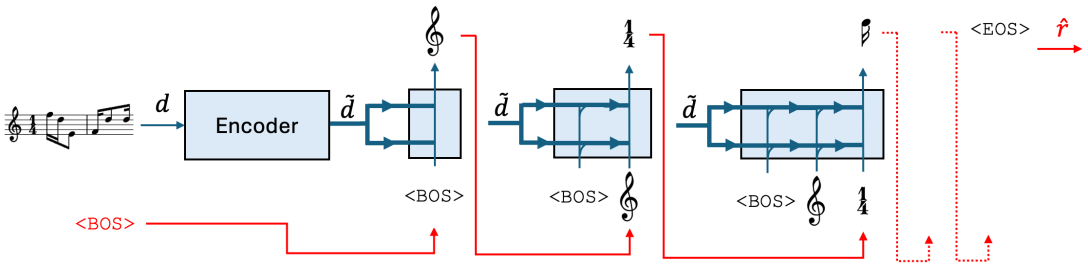
$$\mathcal{L}_{seq} = \sum_{i=1}^{m+1} l(n_i, \hat{n}_i) \tag{2}$$

3.6 Document-to-set bias

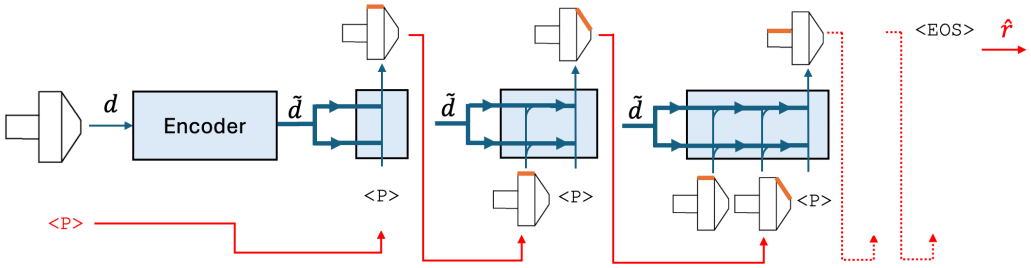
For document types with a set record structure, the document-to-record task becomes a document-to-set task. This section explains how to adapt the document-to-sequence model to the document-to-set task by replacing the sequence bias of next-node prediction with a set bias, termed here *"remaining-node prediction"*. We later apply this method to shape drawings as an exemplary document type.

For a set structure, the sequential bias of next-node prediction makes no sense as there is no natural notion of a "next node" in an unordered set of nodes. Therefore, we replace next-node prediction with remaining-node prediction: This means that from each final node embedding (after attending to the encoded document and previous node embeddings within the decoder), we predict *any* record node visible in the document but not "taken" by the previous node embeddings. Previous nodes are defined by the decoder's input, which is a random permutation of the record nodes during training and autoregressive predictions during inference. This is the main idea of the adaptation to a set record structure; however, for it to successfully work in a transformer-based decoder, a decisive technical detail is needed.

A critical part of the success behind transformer-based next-token prediction is to significantly speed up training by predicting the next token at each position simultaneously in a single forward pass, based on the previous ground-truth tokens, a technique known as teacher forcing [81]. However, this becomes problematic for (naive) remaining-node prediction: Due to this parallel prediction, each node embedding is transformed, layer by layer, to become its required prediction (here, one of its remaining nodes), while, at the same time, previous node embeddings are transformed to become their required predictions (here, one of their remaining nodes). Yet, if the previous node embeddings are transformed, the original "taken" node information is lost. Thus, it becomes impossible for later node embeddings to



(a) Sequence-to-sequence inference process: Predict one *next* node at a time.



(b) Sequence-to-set inference process: Predict one *remaining* node at a time.

Fig. 6: The proposed sequence-to-set inference process (bottom) compared to the well-known sequence-to-sequence inference process (top). Inference is basically the same autoregressive mechanism, where sequence-to-sequence prepends the `<BOS>` token, while sequence-to-set appends the `<P>` token.

know which nodes are remaining, that is, which nodes they can predict, and training fails. This issue highlights that embeddings have two roles in a transformer-based decoder: (a) they must be transformed to become their required prediction and (b) they must carry the (contextualized) original information for the other embeddings. He et al. [82] empirically validate this conflict, showing through probing that token embeddings gradually transition from representing input information to representing the prediction target as they move through the layers.

For remaining-node prediction to work successfully, we must separate these two roles into two separate embeddings: one prediction embedding that, in our case, becomes a remaining node (role "a") and another node embedding that carries the "taken" node information for the other embeddings (role "b"). This happens to be similar to the architectural adaptations of XLNet [65]; however, we do it for (remaining-node) set prediction, whereas XLNet is motivated to learn (autoregressive, bidirectional) sequence prediction (cf. [83]). Specifically, we add one prediction token `<P>` in front of each node, allowing for parallel prediction at the cost of doubling the decoder's sequence length, and add one extra prediction token `<P>` at the end to predict a special end-of-record token `<EOS>` as a stop signal during autoregressive inference. Each prediction token is then mapped to a prediction embedding, which, with the node embeddings, forms the decoder's input. Inside the decoder, prediction and node embeddings can only attend to previous or "taken" node embeddings; prediction embeddings are never attended to as they carry information irrelevant to other embeddings. From each final prediction embedding, we predict a remaining node by being subject to a *remaining-node loss* that is the minimum dissimilarity over all remaining nodes (see Equation (3b)). From each final node embedding, we predict its original input node, encouraging the model to preserve (or pass through) the "taken" node information necessary for other embeddings (see Equation (3d)). Fig. 5b illustrates this training process. For the set inductive bias, we do not add positional encoding, as there is no sequential order in the node embeddings. Inference happens

autoregressively, predicting one remaining node at a time (see Fig. 6b).

Formally, the set bias loss is defined as follows: Let r be a permutation of a set of m nodes $(n_1, ..., n_m)$. Let p be the prediction token `<P>` and n_{m+1} be the `<EOS>` token (which, for notational convenience, is treated as a property-less node with unique type). Let $\hat{n}_i = \hat{g}^*(d, p, n_1, \ldots, n_{i-1})$ be the prediction of a remaining node based on the document d, the prediction token p and the taken nodes $(n_1, \ldots, n_{i-1})$. Let $\tilde{n}_i = \tilde{g}^*(d, n_1, \ldots, n_{i-1}, n_i)$ be the prediction (or pass-through) of a taken node n_i based on the document d, the actual node n_i and previous taken nodes $(n_1, \ldots, n_{i-1})$; $\hat{g}^*$ and $\tilde{g}^*$ representing our model. Let $l(n, \hat{n})$ be the same dissimilarity measure between a node n and a prediction $\hat{n}$ as above. The loss $\mathcal{L}_{set}$ is defined as the loss for the remaining-node predictions $\mathcal{L}_{rem}$, the loss to predict the `<EOS>` token correctly $\mathcal{L}_{eos}$, and the loss for passing through the taken nodes $\mathcal{L}_{taken}$:

$$\mathcal{L}_{set} = \mathcal{L}_{rem} + \mathcal{L}_{eos} + \mathcal{L}_{taken} \tag{3a}$$

$$\mathcal{L}_{rem} = \sum_{i=1}^{m} \min_{j=i}^{m} l(n_j, \hat{n}_i) \tag{3b}$$

$$\mathcal{L}_{eos} = l(n_{m+1}, \hat{n}_{m+1}) \tag{3c}$$

$$\mathcal{L}_{taken} = \sum_{i=1}^{m} l(n_i, \tilde{n}_i) \tag{3d}$$

3.7 Document-to-graph bias

For document types with records best captured in an unrestricted graph structure, the document-to-record task becomes a document-to-graph task. In this section, we adapt the document-to-set model of the previous section to the document-to-graph task by forming an implicit graph from a set structure containing nodes for both record nodes and their relationships (edges). We assume no order in the nodes or edges, but differentiate between them by first predicting all record nodes. We later apply this method to simplified engineering drawings as an exemplary document type.

We represent the record graph as an implicit graph induced by a relational model [84, 85]: Each record node gets a unique identifier and relations between nodes are represented as additional *"relationship nodes"* with properties to link record nodes by their identifier. For example, an undirected graph of three nodes $\{A, B, C\}$ and an edge from A to B will be represented as a set of four nodes including the additional relationship node Z with two nodes as properties: $\{A, B, C, Z(A, B)\}$. This represents a graph as a set of nodes indirectly related through their properties, allowing the use of the same architecture as in Section 3.6 for the processing of any graph.

During training, when generating permutations of the record, we ensure that all record nodes appear before all relationship nodes, guaranteeing that the record nodes and their identifiers are known when predicting their relationships (see Fig. 5c). We use the position in the decoder sequence as the identifier of a node and add one-dimensional positional encoding to provide this positional information to the model with a prediction embedding sharing the same positional encoding as the node embedding to its right. Given the above graph example, we generate, e.g., the following permutations during training: $[A, B, C, Z(0, 1)]$, $[C, B, A, Z(1, 2)]$. This setup allows the model to first predict the set of record nodes visible in the document (symbol recognition) and then predict their specific relationships (symbol assembly).

Formally, the graph bias loss is defined as follows: Let r be two concatenated sequences: A sequence of m record nodes $(n_1, ..., n_m)$ and a sequence of o relationship nodes $(n_{m+1}, ..., n_{m+o})$. Let p be the prediction token `<P>` and n_{m+o+1} be the `<EOS>` token (which, for notational convenience, is treated as a property-less node with unique type). Let $\hat{n}_i = \hat{g}^*(d, p, n_1, \ldots, n_{i-1})$ be the prediction of a remaining node based on the document d, the prediction token p and the taken nodes $(n_1, \ldots, n_{i-1})$. Let $\tilde{n}_i = \tilde{g}^*(d, n_1, \ldots, n_{i-1}, n_i)$ be the prediction (or pass-through) of the taken node n_i based on the document d, the actual node n_i and previous taken nodes $(n_1, \ldots, n_{i-1})$; $\hat{g}^*$ and $\tilde{g}^*$ representing our model. Let $l(n, \hat{n})$ be the same dissimilarity measure between a node n and a prediction $\hat{n}$ as above. The loss $\mathcal{L}_{graph}$ is defined as the loss for the remaining-record-node predictions $\mathcal{L}_{rem}^{nodes}$, the remaining-relationship-node predictions $\mathcal{L}_{rem}^{rel}$, the loss to predict the `<EOS>` token correctly $\mathcal{L}_{eos}$, and the loss for passing through the taken nodes

$\mathcal{L}_{taken}$:

$$\mathcal{L}_{graph} = \mathcal{L}^{nodes}_{rem} + \mathcal{L}^{rel}_{rem} + \mathcal{L}_{eos} + \mathcal{L}_{taken} \tag{4a}$$

$$\mathcal{L}^{nodes}_{rem} = \sum_{i=1}^{m} \min_{j=i}^{m} l(n_j, \hat{n}_i) \tag{4b}$$

$$\mathcal{L}^{rel}_{rem} = \sum_{i=m+1}^{m+o} \min_{j=i}^{m+o} l(n_j, \hat{n}_i) \tag{4c}$$

$$\mathcal{L}_{eos} = l(n_{m+o+1}, \hat{n}_{m+o+1}) \tag{4d}$$

$$\mathcal{L}_{taken} = \sum_{i=1}^{m+o} l(n_i, \tilde{n}_i) \tag{4e}$$

4 Experiments

4.1 Overview

To show the validity of our approach, we conduct experiments on three archetypal document types for the progression of simple to more advanced record structures discussed above (i.e., sequence, set, and graph): We introduce how we represent records and create respective training data for monophonic sheet music (Section 4.2), shape drawings (Section 4.3), and simplified engineering drawings (Section 4.4). We then detail the common experimental setup for model training and evaluation in Section 4.5. We report results in Section 4.6 and provide an ablation study that sheds light on the influence of an appropriate record structure bias in Section 4.7.

4.2 Sequence bias: Sheet music

We use monophonic sheet music documents as an exemplary document type for the document-to-sequence task.

Record representation. The sheet music record is a sequence of one clef node, one time signature node, followed by music note nodes. The clef node and the time signature node have one discrete property: the clef type and the time signature. Each music note node has two discrete properties: duration and pitch (vertical note position).

Data engine. For training, we generate 40 000 synthetic, simplified (monophonic, four-bar, single-staff) music sheets. We generate two possible clef types (treble 𝄞 or bass 𝄢), four possible time signatures (1/4 $\frac{1}{4}$, 2/4 $\frac{2}{4}$, 3/4 $\frac{3}{4}$, and common time 𝐂), five possible note durations (whole 𝅝, half 𝅗𝅥, quarter ♩, eighth ♪, and sixteenth 𝅘𝅥𝅯) and nine possible pitches spanning five staff lines and four in-between staff spaces. All property values are chosen uniformly at random, with note duration respecting the remaining duration of the current bar.

Synthesis. We render each synthetically generated staff onto a raster image. The image is 100 pixels high and has a flexible width to guarantee that all music notes are rendered on one single staff line. Fig. 7 shows an example.

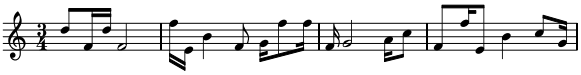


Fig. 7: Example of a simple sheet music document used for training. Fig. F.2 shows more examples.

4.3 Set bias: Shape drawings

We use shape drawings as an exemplary document type for the document-to-set task.

Record representation. We represent shape drawings as a set of record nodes representing simple geometric shapes. Without loss of generality, our simplified proof-of-concept implementation will use a line node and a circle node; others could be added. Each line node contains two properties, representing the coordinates of the start point and the end point in a tuple. Each circle node contains two similar properties as well: the circle's leftmost point and rightmost point (i.e., the extreme points on the horizontal middle axis, implicitly defining also the circle's height). Each coordinate tuple consists of continuous values of normalized x- and y pixel coordinates.

Data engine. The ABC dataset [86] contains one million 3D parts as 3D scene graphs. We randomly project 300 000 of those to 2D edges as seen from one of the six principal views using the hidden line removal algorithm [87]. We clean the projection by removing superimposed edges and rejoining 2D edges that were originally joined in 3D. We only take projections containing up to 10 circles or lines and no other edges like arcs or b-splines. The final dataset consists of about 900 000 projections.

Synthesis. We render all lines and circles onto a 280×280 pixel image. During training, we augment each record with random margins, random translations, random scaling, and axis mirroring while ensuring all lines and circles remain fully visible (see Fig. 8).

4.4 Graph bias: Engineering drawings

We use simplified engineering drawings as an exemplary document type for the document-to-graph task.

Record representation. We represent simplified engineering drawings as a record of connected line nodes and dimension annotation nodes (numbers indicating relative line lengths). Each line node contains two properties, representing the coordinates of the start point and the end point in a tuple. Each dimension annotation contains a single property, representing the coordinates of its center in a tuple (stored in the same format as described above). Relationship nodes are the connections between two lines or the link between a dimension annotation and its line.

Data engine. We create synthetic L-shaped drawings with random augmentations, specifically, mirroring, rotations with multiples of 90 degrees, and translations. Each L-shape has six lines, each connected to its neighbors and each having a 30 % chance of having a dimension annotation. Figures 9b and 9c visualize an example.

Synthesis. We render each L-shape onto a 140×140 pixel image. Dimension annotations are rendered as a random number between 0 and 100, accompanied by visually guiding helplines and arrows. Fig. 9a shows an example. The rendering simulates multiple synthesis functions through randomized line thicknesses, grayscale variations, 3 arrow styles, 4 fonts, 10 font sizes, and random blur augmentation. Fig. F.8 shows examples.

4.5 General experimental setup

All experiments and models follow the same general setup, and all models have around 65 million learnable parameters.

Encoder: The encoder is a basic vision transformer. We divide the input image into patches of size 10×10 pixels, add 2D positional encodings [88], remove all uninformative patches (i.e., background-only), and linearly map the flattened informative patches to form the encoder's input. The encoder is a vision transformer with 3 layers, a 512-dimensional embedding space, and 8-head residual self-attention (unmasked) followed by a residual feed-forward network. The feed-forward network consists of, respectively, a 2048-dimensional hidden layer with no bias term and the GELU activation function [89], a layer norm layer, a 512-dimensional output layer with no bias term and linear activation function, and another layer norm layer.

Decoder: The decoder follows the same setup as the encoder, except that it uses cross-attention on the image embeddings from the encoder and masked self-attention based on the above bias adaptations. *Node encoder:* The beginning-of-record token `<BOS>` and prediction token `<P>` are each mapped to 512-dimensional embedding vectors. A record node is mapped to a node embedding in the following way: The node type and discrete properties are each mapped to a 64-dimensional vector. Each coordinate property value is linearly mapped to a 64-dimensional vector. The node type and properties are concatenated and linearly mapped into the decoder's embedding space. *Node prediction head:* From a final decoder embedding, we predict the node type and properties. The node type and each discrete property are linear maps followed by a softmax activation function. Coordinate properties use a

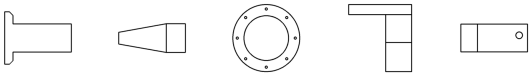
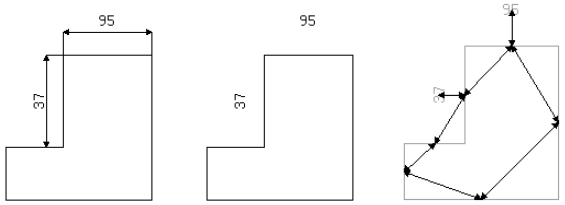


Fig. 8: Five examples of simple technical documents, i.e, rendered projections of engineering parts, used for training. Fig. F.5 shows more examples.



(a) Document (b) Nodes (c) Edges

Fig. 9: Example of a synthetic engineering drawing. Left: The document, where helplines and arrows help to communicate links between dimension annotations and lines. Middle, right: A visualization of the pure record, namely the record nodes (middle) and the relationships (edges; see left) by means of arrows.

novel coarse-to-fine method for pixel-precise prediction: First, we predict the image patch for each coordinate property by calculating a cross-attention probability from the node embedding to the image patches. Then, we predict the pixel inside the 10×10 patch by concatenating the most probable patch embedding to the node embedding and linearly map it to 100 softmax outputs.

Loss functions: Two loss functions are used as outlined above – pairwise loss and remaining-node loss. All loss concerning the prediction of node properties contribute to the overall loss only if the respective node type was predicted correctly; pixel prediction loss only if the respective patch was predicted correctly.

Training: All models are trained to convergence using their respective domain-specific data engine and a batch size of 32. This results in training the sheet music model for 31 250 steps: 25 epochs of 40 000 training samples; the shape drawing model for 506 250 steps: 18 epochs of 900 000 augmented training samples; the engineering drawing model for 62 500 steps: 40 epochs of 50 000 generated training samples. For optimization, we use the AdamW optimizer [90] with a learning rate of $1 \cdot 10^{-4}$, $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1 \cdot 10^{-9}$, a clip norm of 0.1, and a weight decay of 0.004 based on preliminary experiments. To speed up training, we use batch processing and a hardware-friendly structure of array data representation (see Section B).

Evaluation: All models are evaluated using set-aside validation data created by the same domain-specific data engine. Model performance is evaluated using the *transcription accuracy*, defined as the fraction of predictions ($\hat{r}$) that are equal ($\hat{r} = r$) to their target records (r). Records are equal if their nodes, edges, and structure are equal (i.e., a property graph isomorphism, see Section C for a formal definition). For sequences, this means all nodes in the sequence must be pairwise equal. For graphs and sets, this means they must contain the same record nodes, the same relationship nodes, and the same structure. Two nodes are equal if their type is equal and their properties are equal (up to a predefined precision threshold for continuous coordinates). Thus, transcription accuracy renders the entire transcription incorrect for any partial error, e.g., predicting one property wrong. We believe this to be appropriate as any possible error may have detrimental consequences in some downstream task. Note that this conservative metric renders the task particularly hard, similar to the exact match metric in multi-label classification [91], which considers a multi-label classification incorrect if any label is wrong or missing.

4.6 Results

In the main set of experiments, we evaluate the performance when the appropriate inherent bias for each given record structure is used.

Sheet music recognition as document-to-sequence transcription: We use 10 000 randomly generated single-staff music sheets to evaluate the model adapted to the sequential bias (Section 3.5). The model transcribes 9659 sheets of music exactly and has a transcription accuracy of 96.6 %. This is hard to compare exactly to the more forgiving metrics like IoU usually used in OMR [92], but appears on par with the state of the art for this relatively simple task.

Shape drawing recognition as document-to-set transcription: For evaluation, we generate 10 000 2D projections of separate 3D parts from the ABC data set. Our model transcribes 7490 projections correctly up to a coordinate precision threshold of about ± 4 pixels. This results in a transcription accuracy of 74.9 %, indicating that our method successfully learned the document-to-set task end to end.

Engineering drawing recognition as document-to-graph transcription: We use 1000 randomly generated L-shapes using the same synthetic data engine as for training. The model (with graph bias) transcribes 748 L-shapes correctly up to a coordinate precision threshold of about ± 4 pixels. This results in a transcription accuracy of 74.8 %. To the authors' knowledge, this is the first time that the transcription of mechanical engineering drawings has been successfully learned end-to-end, and only recently has successful end-to-end learning of any inherently non-sequential document type been reported for the first time [39, 40].

The above results are proof of principle that our method of inductive bias adaptation works, i.e., it enables a model to successfully learn the document-to-sequence, document-to-set, and document-to-graph tasks.

4.7 Ablation studies: Choosing inappropriate biases

In this section, we conduct an ablation study to determine whether the transcription performance above is actually due to our method of bias adaptation by deliberately using a non-optimal bias for certain cases: The set and graph inductive bias in case of an inherently sequential record structure and the sequential inductive bias in case of an inherently set or graph record structure. This last setup basically replicates what has implicitly been done in much of past document recognition works.

Treating sheet music with a set bias: The model architecture built for the set inductive bias (Section 3.6) is adapted to the sheet music scenario (Section 3.5) with its inherently sequential record structure. As this architecture predicts music symbols in no particular order, we must add a horizontal position property to each record node, representing its position in the sequence, to recover the actual sequential record at inference time. This approach effectively encodes a sequence as a set, as suggested in [54]. Using the set inductive bias leads to a more difficult task, as the model must learn the sequential nature on its own without structural guidance through an appropriately biased model architecture. For example, the prediction of the last record node requires checking all "taken" nodes against all symbols visualized in the document. The model transcribes 1937 sheets of music exactly and thus has a transcription accuracy of 19.4 %, dropping 77.2 percentage points compared to the model with sequential bias. The model performs especially badly on longer sequences with a transcription accuracy of 0.0 % for more than 15 symbols. Similarly, the model architecture employing the general graph inductive bias (Section 3.7) yields 0.0 % accuracy under the baseline experimental setup. The set and graph bias make fewer assumptions about the underlying graph structure than the sequential bias. While this makes the training less data-efficient, more data would enable the model (or a bigger model) to eventually learn the inherent structure and solve such transcription tasks. For example, extending the experimental training setup for the same graph model to 500 epochs (and increasing batch size to 512) improves the transcription accuracy to 10.4 %.

		Inductive bias		
		seq	set	graph
Record	seq	**96.6 %**	19.4 %	0.0 %
	set	12.0 %	**74.9 %**	-
	graph	0.0 %	-	**74.8 %**

Table 1: Modeling appropriate structure-specific inductive biases (bold numbers) expectedly shows a significantly better transcription accuracy than applying inappropriate biases (plain numbers).

Treating shape drawings with a sequence bias: The model architecture built for the sequential inductive bias (Section 3.5) is adapted to the shape drawing scenario (Section 3.6) with its inherent set record structure. Using the sequential bias leads to a more difficult task, as the model must learn the uniform distribution over the remaining nodes rather than a single node, hardened by record nodes carrying a type and properties. The model transcribes 1193 drawings correctly up to a coordinate precision threshold of about ±4 pixels, resulting in a transcription accuracy of 12.0 %, dropping 62.9 percentage points compared to the model with set bias. The model performs especially badly on records with more than four record nodes, making only 23 correct transcriptions and a transcription accuracy of 0.6 % on such records. The sequential bias makes more (here: wrong) assumptions about the underlying graph structure. These assumptions lead to a noisy training signal, as the predicted next node can be any remaining node. We suspect that this noisy signal leads to unstable training, making more data insufficient for learning non-sequential transcription tasks. Similarly, adapting this misaligned bias to the engineering drawings scenario (Section 3.7), which possesses a general graph structure, fails entirely, yielding 0 correct transcriptions (a 0.0 % accuracy). This suggests that, in such cases, a more general bias design is not only beneficial but also necessary, e.g., for a document foundation model supporting various document types.

Table 1 summarises these results. The omitted entries ("-") represent redundant configurations: applying a graph bias to a set record trivially reduces to the set bias, while extending a set bias to model relations yields the graph bias.

4.8 Limitations and scope

Our controlled environments rely on two primary simplifications. First, we base our custom inductive bias design on a unified base architecture, making only slight adaptations to minimize confounding architectural variables. Second, all experiments are conducted on synthetic, simplified data. While these simplifications serve as valid experimental constraints to substantiate our central claim, they highlight the following challenges for scaling document transcription to more realistic settings:

Inductive bias design: The document-to-record task maps a specific record structure to a class of structure-specific inductive biases; however, determining the optimal design remains an open research question, particularly for non-sequential structure prediction. While remaining-node prediction offers conceptual advantages over existing methods (see Section 2.2), these must be shown to translate into empirical gains in complex, real-world settings. Conversely, existing bipartite matching approaches, though successful in scene graph generation, must demonstrate that they can scale to graphs of higher order and size: While we here primarily motivate structure-specific inductive biases for document transcription, the connection we establish is bidirectional; thus, document transcription itself serves as a rich and relevant testbed for improving inductive-bias design in structured-output prediction.

Data acquisition and scale: Our approach assumes access to large-scale datasets containing ground-truth record representations. For convention-bound document types, record-aligned representations often already exist (frequently driven by the conventions themselves), making our approach directly applicable. Examples of such representations include the Kern syntax or MusicXML for OMR, LaTeX for mathematical expressions, the data models generated during CAD workflows for engineering drawings, and BIM workflows for floor plans.

For domains lacking such representations, data acquisition will be costly. However, the document-to-record perspective offers a mitigation strategy: by grouping seemingly unrelated document types according to their shared inherent record structure, it enables cross-domain data pooling. This structural alignment enables cross-domain learning and serves as a foundational step toward developing highly capable, generalized document foundation models.

5 Conclusion

This paper started with the observation that documents stored as images on the one side and pictures taken in the real world on the other side have many principled differences, calling for distinct approaches for their analysis. In addition to evident dissimilarities in pixel density, which later motivated our removal of uninformative patches from model input, we highlighted fundamental distinctions in their content and meaning: Documents are designed to convey a fixed set of information, which led to a distinct perspective: Rather than recognizing selected aspects in a document only, document recognition should be regarded as a transcription task of the full information. An analogous transcription is impossible for complex and information-rich natural images, where no description can capture all relevant information to form a natural interface for downstream tasks.

Taking on this perspective shifts the focus of document recognition away from traditional practices – adapting naturally misaligned computer vision approaches – to what content must be extracted for holistic document understanding, i.e., the transcription or record of that document. That shift to document-to-record transcription revealed several intrinsic, archetypal record structures shared across otherwise disjointed document types, such as documents containing sequential data forming linear graphs, or interlinked data forming unrestricted graphs. These kinds of structures group document domains on a conceptual level, enabling a method of principled inductive bias design to better predict outputs following these structures. This is the main conceptual advance presented in this paper.

For validation, we used this method to design inductive biases for three progressively more complex record structures and learn respective document-to-record transcription models efficiently (specifically: data-efficiently) in an end-to-end framework. In particular, based on this approach we showed a successful implementation of an end-to-end learned document recognition

system for engineering drawings for the first time in the literature. Ablation studies showed that the advance of improved transcription accuracy with less training data can indeed be attributed to the design of suitable inductive biases and not to other components of the pipeline or setup.

By framing document recognition as a domain-agnostic (but record-structure-specific) transcription task, our work opens numerous avenues for future research. First, it enables (end-to-end) learning to understand non-sequential document types, which previously lacked a viable path to achieve similar recognition success than simpler document types. Second, the presented method for structured inductive bias design stimulates inquiry into further structure-imposed graph types present in documents' records, such as unordered tree structures (e.g., hierarchical organizational charts) or multi-type graph compositions (for example, engineering drawing recognition can be decomposed into first recognizing the 3D shape from the 2D outlines – an unrestricted graph, as shape primitives are interlinked –, followed by matching the annotations – an unordered tree structure, as annotations do not annotate each other). Third, our work groups document types conceptually that were disjoint previously, paving the way for document foundation models trained across many document types, potentially enabling emergent capabilities in general document understanding.

Finally, beyond its theoretical implications, this perspective provides a practical rubric for end-to-end document recognition by distilling semantic information from their visual and syntactic representations. We provide a detailed summary on these practical considerations in Section E.

Acknowledgements. This work has been financially supported by the Innosuisse grant 102.983 IP-ICT "Master3D" and the PhD support grant "DDLearn" from the ZHAW School of Engineering.

References

[1] Stadelmann, T., Amirian, M., Arabaci, I., Arnold, M., Duivesteijn, G.F., *et al.*: Deep learning in the wild. In: Proc. of the Artificial Neural Networks in Pattern Recognition 8th IAPR TC3 Workshop, pp. 17–38. Springer, Siena, Italy (2018). DOI: 10.1007/978-3-319-99978-4_2

[2] Chai, J., Zeng, H., Li, A., Ngai, E.W.: Deep learning in computer vision: A critical review of emerging techniques and application scenarios. Machine Learning with Applications **6**, 100134–100147 (2021) DOI: 10.1016/j.mlwa.2021.100134

[3] Subramani, N., Matton, A., Greaves, M., Lam, A.: A survey of deep learning approaches for OCR and document understanding. arXiv preprint (2020) DOI: 10.48550/arXiv.2011.13534

[4] Ríos-Vila, A., Calvo-Zaragoza, J., Paquet, T.: Sheet Music Transformer: End-to-end optical music recognition beyond monophonic transcription. In: Proc. of the 18th Int. Conf. Doc. Anal. Recognit. (ICDAR), pp. 20–37. Springer, Athens, Greece (2024). DOI: 10.1007/978-3-031-70552-6_2

[5] Meier, B., Stadelmann, T., Stampfli, J., Arnold, M., Cieliebak, M.: Fully convolutional neural networks for newspaper article segmentation. In: Proc. of the 14th Int. Conf. Doc. Anal. Recognit. (ICDAR), vol. 1, pp. 414–419 (2017). DOI: 10.1109/ICDAR.2017.75

[6] Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., *et al.*: TrOCR: Transformer-based optical character recognition with pre-trained models. In: Proc. of the 37th AAAI Conf. Artif. Intell., Washington, DC, USA, pp. 13094–13102 (2023). DOI: 10.1609/aaai.v37i11.26538

[7] Schmitt-Koopmann, F.M., Huang, E.M., Hutter, H.-P., Stadelmann, T., Darvishy, A.: MathNet: A data-centric approach for printed mathematical expression recognition. IEEE Access **12**, 76963–76974 (2024) DOI: 10.1109/ACCESS.2024.3404834

[8] Wei, H., Liu, C., Chen, J., Wang, J., Kong, L., *et al.*: General OCR theory: Towards OCR-2.0 via a unified end-to-end model.

arXiv preprint (2024) DOI: 10.48550/arXiv.2409.01704

[9] Sarkar, S., Pandey, P., Kar, S.: Automatic detection and classification of symbols in engineering drawings. arXiv preprint (2022) DOI: 10.48550/arXiv.2204.13277

[10] Rezvanifar, A., Cote, M., Albu, A.B.: Symbol spotting on digital architectural floor plans using a deep learning-based framework. In: Proc. of the 2020 Conf. Comput. Vis. Pattern Recognit. Workshops, Seattle, WA, USA, pp. 568–569 (2020). DOI: 10.1109/CVPRW50498.2020.00292

[11] Uzair, W., Chai, D., Rassau, A.: ElectroNet: An enhanced model for small-scale object detection in electrical schematic diagram. Research Square preprint (2023) DOI: 10.21203/rs.3.rs-3137489/v1

[12] Mardiana, B.D., Hadiningrum, T.R., Siahaan, D.: Comparative analysis of deep learning models for validating use case diagrams. In: Proc. of the 16th Int. Conf. Inf. Technol. Electr. Eng. (ICITEE), pp. 141–146. IEEE, Bali, Indonesia (2024). DOI: 10.1109/ICITEE62483.2024.10808842

[13] Gada, M.: Object detection for P&ID images using various deep learning techniques. In: Proc. of the 2021 Int. Conf. Comput. Commun. Inform. ICCCI, pp. 1–5. IEEE, Coimbatore, India (2021). DOI: 10.1109/ICCCI50826.2021.9402386

[14] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., *et al.*: ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. (IJCV) **115**, 211–252 (2015) DOI: 10.1007/s11263-015-0816-y

[15] Minaee, S., Boykov, Y., Porikli, F., Plaza, A., Kehtarnavaz, N., *et al.*: Image segmentation using deep learning: A survey. IEEE Trans. Pattern Anal. Mach. Intell. (PAMI) **44**, 3523–3542 (2021) DOI: 10.1109/TPAMI.2021.3059968

[16] Zou, Z., Chen, K., Shi, Z., Guo, Y., Ye, J.: Object detection in 20 years: A survey. Proc. of the IEEE **111**(3), 257–276 (2023) DOI: 10.1109/JPROC.2023.3238524

[17] Tuggener, L., Elezi, I., Schmidhuber, J., Stadelmann, T.: Deep watershed detector for music object recognition. arXiv preprint (2018) DOI: 10.48550/arXiv.1805.10548

[18] Yamasaki, T., Zhang, J., Takada, Y.: Apartment structure estimation using fully convolutional networks and graph model. In: Proc. of the 2018 ACM Workshop on Multimedia for Real Estate Tech. RETech'18, pp. 1–6. Association for Computing Machinery, New York, NY, USA (2018). DOI: 10.1145/3210499.32105

[19] Buzzy, M., Thesma, V., Davoodi, M., Mohammadpour Velni, J.: Real-time plant leaf counting using deep object detection networks. Sensors **20**(23), 6896–6910 (2020) DOI: 10.3390/s20236896

[20] Pal, S.K., Pramanik, A., Maiti, J., Mitra, P.: Deep learning in multi-object detection and tracking: state of the art. Applied Intelligence **51**, 6400–6429 (2021) DOI: 10.1007/s10489-021-02293-7

[21] Khor, K.S., Liu, C., Cheah, C.C.: Robotic grasping of unknown objects based on deep learning-based feature detection. Sensors **24**(15), 4861–4882 (2024) DOI: 10.3390/s24154861

[22] Hays, J., Efros, A.A.: IM2GPS: Estimating geographic information from a single image. In: Proc. of the 2008 Conf. Comput. Vis. Pattern Recognit., pp. 1–8. IEEE, Anchorage, AK, USA (2008). DOI: 10.1109/CVPR.2008.4587784

[23] Wilson, R.J.: Introduction to Graph Theory, 4th edn. Addison Wesley, Harlow, UK (1986)

[24] Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Adv. Neural Inf. Process. Syst., vol. 27. Montréal, Kanada (2014). DOI: 10.48550/arXiv.1409.3215

[25] Poznanski, J., Borchardt, J., Dunkelberger,

J., Huff, R., Lin, D., *et al.*: olmOCR: Unlocking trillions of tokens in PDFs with vision language models. arXiv preprint (2025) DOI: 10.48550/arXiv.2502.18443

[26] Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., *et al.*: Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. arXiv preprint (2024) DOI: 10.48550/arXiv.2409.12191

[27] Hamdi, L., Tamasna, A., Boisson, P., Paquet, T.: VISTA-OCR: Towards generative and interactive end to end OCR models. arXiv preprint (2025) DOI: 10.48550/arXiv.2504.03621

[28] Xypolopoulos, C., Shang, G., Fei, X., Nikolentzos, G., Abdine, H., *et al.*: Graph linearization methods for reasoning on graphs with large language models. arXiv preprint (2024) DOI: 10.48550/arXiv.2410.19494

[29] Hajic, J., Dorfer, M., Widmer, G., Pecina, P.: Towards full-pipeline handwritten OMR with musical symbol detection by U-nets. In: Proc. of the 19th Trans. Int. Soc. Music Inf. Retr. (ISMIR), Paris, France, pp. 225–232 (2018). DOI: 10.5281/zenodo.1492388

[30] Tuggener, L., Satyawan, Y.P., Pacha, A., Schmidhuber, J., Stadelmann, T.: The DeepScoresV2 dataset and benchmark for music object detection. In: 2020 25th Int. Conf. on Pat. Recog. (ICPR), pp. 9188–9195. IEEE, Milan, Italy (2021). DOI: 10.1109/ICPR48806.2021.9412290

[31] Schmitt-Koopmann, F.M., Huang, E.M., Hutter, H.-P., Stadelmann, T., Darvishy, A.: FormulaNet: A benchmark dataset for mathematical formula detection. IEEE Access **10**, 91588–91596 (2022) DOI: 10.1109/ACCESS.2022.3202639

[32] Kim, H., Kim, S., Yu, K.: Automatic extraction of indoor spatial information from floor plan image: A patch-based deep learning methodology application on large-scale complex buildings. ISPRS Int. J. Geo-Inf. **10**, 828–843 (2021) DOI: 10.3390/ijgi10120828

[33] Seo, J., Park, H., Choo, S.: Inference of drawing elements and space usage on architectural drawings using semantic segmentation. Applied Sciences **10**, 7347–7362 (2020) DOI: 10.3390/app10207347

[34] Huber, F., Hagel, G.: Towards detection and syntactical analysis in UML class diagrams for software engineering education. In: Proc. of the 2020 IEEE Glob. Eng. Educ. Conf. (EDUCON), pp. 3–7. IEEE, Porto, Portugal (2020). DOI: 10.1109/EDUCON45650.2020.9125244

[35] Rosen, K.H., Krithivasan, K.: Discrete Mathematics and Its Applications, 7th edn. McGraw-Hill, New York, NY (2013)

[36] Lee, K., Joshi, M., Turc, I.R., Hu, H., Liu, F., *et al.*: Pix2Struct: Screenshot parsing as pretraining for visual language understanding. In: Proc. of the 40th Int. Conf. Mach. Learn. (ICML), pp. 18893–18912. PMLR, Honolulu, HI, USA (2023). DOI: 10.48550/arXiv.2210.03347

[37] Cong, Y., Yang, M.Y., Rosenhahn, B.: RelTR: Relation transformer for scene graph generation. IEEE Trans. Pattern Anal. Mach. Intell. (PAMI), 11169–11183 (2023) DOI: 10.48550/arXiv.2201.11460

[38] Shit, S., Koner, R., Wittmann, B., Paetzold, J., Ezhov, I., *et al.*: Relationformer: A unified framework for image-to-graph generation. In: Proc. of the European Conf. Comput. Vis. (ECCV), pp. 422–439. Springer, Tel Aviv, Israel (2022). DOI: 10.48550/arXiv.2203.10202

[39] Stürmer, M., Graumann, M., Koch, T.: From engineering diagrams to graphs: digitizing P&IDs with transformers. In: Proc. of the 12th Int. Conf. Data Sci. Adv. Anal. (DSAA), Birmingham, UK (2025). DOI: 10.1109/DSAA65442.2025.11248012

[40] Hu, S., Wu, W., Su, R., Hou, W., Zheng, L., et al.: Raster-to-graph: floorplan recognition via autoregressive graph prediction with an attention transformer **43**(2), 15007 (2024) DOI: 10.1111/cgf.15007

[41] Zhang, Q., Huang, V.S.-J., Wang, B., Zhang, J., Wang, Z., *et al.*: Document parsing unveiled: Techniques, challenges, and prospects for structured information extraction. arXiv preprint (2024) DOI: 10.48550/arXiv.2410.21169

[42] Blecher, L., Cucurull, G., Scialom, T., Stojnic, R.: Nougat: Neural optical understanding for academic documents. arXiv preprint (2023) DOI: 10.48550/arXiv.2308.13418

[43] Hu, A., Xu, H., Ye, J., Yan, M., Zhang, L., *et al.*: mPLUG-DocOwl 1.5: Unified structure learning for OCR-free document understanding. arXiv preprint (2024) DOI: 10.48550/arXiv.2403.12895

[44] Stadelmann, T., Klamt, T., Merkt, P.H.: Data centrism and the core of data science as a scientific discipline. Archives of Data Science, Series A **8**, 1–16 (2022) DOI: 10.5445/IR/1000143637

[45] Luley, P.-P., Deriu, J.M., Yan, P., Schatte, G.A., Stadelmann, T.: From concept to implementation: The data-centric development process for AI in industry. In: Proc. of the 10th Swiss Conf. Data Sci. (SDS), pp. 73–76. IEEE, Zurich, Switzerland (2023). DOI: 10.1109/SDS57534.2023.00017

[46] Tuggener, L., Sager, P., Taoudi-Benchekroun, Y., Grewe, B.F., Stadelmann, T.: So you want your private LLM at home? A survey and benchmark of methods for efficient GPTs. In: Proc. of the 11th Swiss Conf. Data Sci. (SDS), pp. 205–212. IEEE, Zurich, Switzerland (2024). DOI: 10.1109/SDS60720.2024.00036

[47] Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., *et al.*: Relational inductive biases, deep learning, and graph networks. arXiv preprint (2018) DOI: 10.48550/arXiv.1806.01261

[48] Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric Deep Learning: Going beyond euclidean data. IEEE Signal Process. Mag. **34**(4), 18–42 (2017) DOI: 10.1109/MSP.2017.2693418

[49] Gavranović, B., Lessard, P., Dudzik, A., Von Glehn, T., Araújo, J.G., et al.: Position: Categorical Deep Learning is an algebraic theory of all architectures (2024) DOI: 10.48550/arXiv.2402.15332

[50] Shiebler, D., Gavranović, B., Wilson, P.: Category theory in machine learning. arXiv preprint (2021) DOI: 10.48550/arXiv.2106.07032

[51] You, J., Ying, R., Ren, X., Hamilton, W., Leskovec, J.: GraphRNN: Generating realistic graphs with deep auto-regressive models. In: Proc. of the 35th Int. Conf. Mach. Learn. (ICML), vol. 80, pp. 5708–5717. PMLR, Stockholm, Sweden (2018). DOI: 10.48550/arXiv.1802.08773

[52] Chen, X., Wang, Y., He, J., Du, Y., Hassoun, S., *et al.*: Graph generative pre-trained transformer. In: Proc. of the 42nd Int. Conf. Mach. Learn. (ICML), Vancouver, Canada (2025). DOI: 10.48550/arXiv.2501.01073

[53] Wang, Z., Shi, J., Heess, N., Gretton, A., Titsias, M.K.: Learning-order autoregressive models with application to molecular graph generation. In: Proc. of the 42th Int. Conf. Mach. Learn. (ICML) (2025). DOI: 10.48550/arXiv.2503.05979

[54] Vinyals, O., Bengio, S., Kudlur, M.: Order matters: sequence to sequence for sets. In: Proc. of the 4th Int. Conf. Learn. Represent. (ICLR), San Juan, Puerto Rico (2016). DOI: 10.48550/arXiv.1511.06391

[55] Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., *et al.*: DiGress: Discrete denoising diffusion for graph generation. In: Proc. of the 11th Int. Conf. Learn. Represent. (ICLR), Kigali, Rwanda (2023). DOI: 10.48550/arXiv.2209.14734

[56] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., et al.: Language models are unsupervised multitask learners. OpenAI Blog (2019)

[57] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., *et al.*: Language models

are few-shot learners. In: Adv. Neural Inf. Process. Syst., vol. 33, pp. 1877–1901 (2020). DOI: 10.48550/arXiv.2005.14165

[58] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., *et al.*: End-to-end object detection with transformers. In: Proc. of the European Conf. Comput. Vis. (ECCV), pp. 213–229 (2020). DOI: 10.1007/978-3-030-58452-8_13

[59] Zhu, X., Su, W., Lu, L., Li, B., Wang, X., *et al.*: Deformable DETR: Deformable transformers for end-to-end object detection. In: Proc. of the 9th Int. Conf. Learn. Represent. (ICLR) (2021). DOI: 10.48550/arXiv.2010.04159

[60] Li, F., Zhang, H., Liu, S., Guo, J., Ni, L.M., *et al.*: DN-DETR: Accelerate detr training by introducing query denoising. In: Proc. of the IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), New Orleans, LA, USA, pp. 13619–13627 (2022). DOI: 10.1109/CVPR52688.2022.01325

[61] Zhang, H., Li, F., Liu, S., Zhang, L., Su, H., *et al.*: DINO: DETR with improved denoising anchor boxes for end-to-end object detection. In: Proc. of the 11th Int. Conf. Learn. Represent. (ICLR), Kigali, Rwanda (2023). DOI: 10.48550/arXiv.2203.03605

[62] Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., *et al.*: Object-centric learning with Slot Attention. In: Adv. Neural Inf. Process. Syst., vol. 33, pp. 11525–11538 (2020). DOI: 10.48550/arXiv.2006.15055

[63] Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., *et al.*: Set Transformer: A framework for attention-based permutation-invariant neural networks. In: Proc. of the 36th Int. Conf. Mach. Learn. (ICML), Long Beach, CA, USA (2019). DOI: 10.48550/arXiv.1810.00825

[64] Hong, S.-J., Choi, C.-Y., Lee, S.-W.: Layer-wise query selection to eliminate redundant queries in DETR. Applied Sciences **15**(14), 7686 (2025) DOI: 10.3390/app15147686

[65] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., *et al.*: XLNet: generalized autoregressive pretraining for language understanding. In: Adv. Neural Inf. Process. Syst., vol. 32. Vancouver, Canada (2019). DOI: 10.48550/arXiv.1906.08237

[66] Lu, Y., Rai, H., Chang, J., Knyazev, B., Yu, G., *et al.*: Context-aware scene graph generation with seq2seq transformers. In: Proc. of the IEEE/CVF Int. Conf. Comput. Vis. (ICCV), pp. 15931–15941 (2021). DOI: 10.1109/ICCV48922.2021.01563

[67] Kim, B., Lee, J., Kang, J., Kim, E.-S., Kim, H.J.: HOTR: End-to-end human-object interaction detection with transformers. In: Proc. of the IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 74–83 (2021). DOI: 10.1109/CVPR46437.2021.00014

[68] Li, R., Zhang, S., He, X.: SGTR: End-to-end scene graph generation with transformer. In: Proc. of the IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), New Orleans, LA, USA, pp. 19486–19496 (2022). DOI: 10.1109/CVPR52688.2022.01888

[69] Im, J., Nam, J., Park, N., Lee, H., Park, S.: EGTR: Extracting graph from transformer for scene graph generation. In: Proc. of the IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Seattle, WA, USA, pp. 24229–24238 (2024). DOI: 10.1109/CVPR52733.2024.02287

[70] Nienhuys, H.-W., Nieuwenhuizen, J.: LilyPond – Essay on automated music engraving. (2003)

[71] Angles, R.: The property graph database model. In: AMW (2018)

[72] Bishop, C.M.: Mixture density networks. (unpublished) technical report at Aston University (1994)

[73] LeCun, Y.: A path towards autonomous machine intelligence. Version 0.9.2, 2022-06-27 (2022)

[74] Schmidhuber, J., Prelinger, D.: Discovering

predictable classifications. Neural Comput. **5**(4), 625–635 (1993) DOI: 10.1162/neco.1993.5.4.625

[75] Cho, K., Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., *et al.*: Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proc. of the 2014 Conf. Empir. Methods Nat. Lang. Process. (EMNLP), pp. 1724–1734. Association for Computational Linguistics, Doha, Qatar (2014). DOI: 10.3115/v1/D14-1179

[76] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., *et al.*: Attention is all you need. In: Adv. Neural Inf. Process. Syst., vol. 30. Curran Associates, Inc., Long Beach, CA, USA (2017). DOI: 10.48550/arXiv.1706.03762

[77] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., *et al.*: An Image is worth 16x16 words: Transformers for image recognition at scale. In: Proc. of the 9th Int. Conf. Learn. Represent. (ICLR) (2021). DOI: 10.48550/arXiv.2010.11929

[78] He, K., Chen, X., Xie, S., Li, Y., Dollár, P., *et al.*: Masked autoencoders are scalable vision learners. In: Proc. of the 2022 Conf. Comput. Vis. Pattern Recognit. (CVPR), pp. 16000–16009. IEEE, New Orleans, LA, USA (2022). DOI: 10.1109/CVPR52688.2022.01553

[79] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., *et al.*: Graph attention networks. arXiv preprint (2017) DOI: 10.48550/arXiv.1710.10903

[80] Veličković, P.: Everything is connected: Graph neural networks. Curr. Opin. Struct. Biol. (COSB) **79**, 102538 (2023) DOI: 10.1016/j.sbi.2023.102538

[81] Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Comput. **1**, 270–280 (1989)

[82] He, H., Su, W.J.: A law of next-token prediction in large language models. Phys. Rev. E **112**(3), 035317 (2025) DOI: 10.48550/arXiv.2408.13442

[83] Pannatier, A., Courdier, E., Fleuret, F.: σ-GPTs: A new approach to autoregressive models. In: Proc. of the Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases (ECMLPKDD), Vilnius, Lithuania, pp. 143–159 (2024). DOI: 10.48550/arXiv.2404.09562

[84] Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM **13**, 377–387 (1970)

[85] Boudaoud, A., Mahfoud, H., Chikh, A.: Towards a complete direct mapping from relational databases to property graphs. In: Proc. of the 38th Int. Conf. Data Eng. (ICDE), pp. 222–235. Springer, Kuala Lumpur, Malaysia (2022). DOI: 10.1007/978-3-031-21595-7_16

[86] Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., *et al.*: ABC: A big CAD model dataset for geometric deep learning. In: Proc. of the 2019 Conf. Comput. Vis. Pattern Recognit. (CVPR), Long Beach, CA, USA, pp. 9601–9611 (2019). DOI: 10.1109/CVPR.2019.00983

[87] Galimberti, R.: An algorithm for hidden line elimination. Commun. ACM **12**, 206–211 (1969)

[88] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., *et al.*: Image transformer. In: Proc. of the 35th Int. Conf. Mach. Learn. (ICML), vol. 80, pp. 4055–4064. PMLR, Stockholm, Sweden (2018). DOI: 10.48550/arXiv.1802.05751

[89] Hendrycks, D., Gimpel, K.: Gaussian error linear units (GELUs). arXiv preprint (2016) DOI: 10.48550/arXiv.1606.08415

[90] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: Proc. of the 8th Int. Conf. Learn. Represent. (ICLR) (2018). DOI: 10.48550/arXiv.1711.05101

[91] Ganda, D., Buch, R.: A survey on multi label classification. Recent Trends in Programming

Languages **5**, 19–23 (2018)

[92] Tuggener, L., Emberger, R., Ghosh, A., Sager, P., Satyawan, Y.P., *et al.*: Real world music object recognition. Trans. Int. Soc. Music Inf. Retr. **7**, 1–14 (2024) DOI: 10.5334/tismir.157

[93] Su, H., Liu, X., Niu, J., Cui, J., Wan, J., *et al.*: MARVEL: Raster gray-level manga vectorization via primitive-wise deep reinforcement learning. IEEE Transactions on Circuits and Systems for Video Technology **34**, 2677–2693 (2024) DOI: 10.1109/TCSVT.2023.3309786

[94] Lanfant, B., Lack, S., Meyer, B., Abdulkadir, A., Stadelmann, T., *et al.*: 3D Master-based method for optimizing the cost calculation of PBF-LB/M manufactured parts. BHM Berg-und Hüttenmännische Monatshefte **170**(3), 158–171 (2025) DOI: 10.1007/s00501-025-01563-y

Appendix

A Proof for read function space reduction

Here, we provide the mathematical proof that it follows from Equation (1) that the read function space reduces to a single read function: $\mathcal{G} = \{g^*\}$. First, we restate Equation (1) with the function composition definition inserted:

$$\forall r \in \mathcal{R}, \forall f \in \mathcal{F}, \forall g \in \mathcal{G} : g(f(r)) = r \qquad (5)$$

Now, we assume that there exist two functions in $\mathcal{G}$, $g_1 \in \mathcal{G}$ and $g_2 \in \mathcal{G}$ and show that they must be the same function. Given an arbitrary write function $f_n \in \mathcal{F}$, it follows that g_1 and g_2 must be the same function for all records written by f_n:

$$\forall r \in \mathcal{R} : \; g_1(f_n(r)) = r = g_2(f_n(r)) \qquad (6)$$

As we chose f_n arbitrarily, this holds for any $f \in \mathcal{F}$. Thus, g_1 and g_2 are the same function for all valid documents of our domain, where valid documents are $\mathcal{D} = \{f(r) \mid \forall r \in \mathcal{R}, \forall f \in \mathcal{F}\}$. So, g_1 and g_2 are the same function across all domain-relevant inputs; we call that function g^*, and, therefore: $\mathcal{G} = \{g^*\}$.

B Training details

Next to the setup described in Section 4.5, we use a hardware-friendly tensor representation and batch processing to speed up training.

Tensor representation: Current hardware is optimized for tensor operations. Thus, for performance considerations, we use a structure of arrays (SoA) layout: We split the nodes of the input sequence into separate vectors: a vector of node types, a vector of discrete properties, and a vector of continuous properties. To ensure efficient processing and easy masking, we pad fields with varying numbers of properties so that each node occupies the same amount of space within these vectors. With this representation, tensor operations, such as linear maps, can be run over those vectors in parallel. For the output prediction, we still follow the SoA layout: We produce a vector for the logits of all node types and a matrix for the logits of all discrete properties, padded to the maximum number of discrete properties per node type to fit a matrix structure.

Batch processing: During training, we use a batch size of 32. Batch elements can have different sequence lengths, as records have a varying number of nodes and images a varying number of patches (we filter empty patches). Therefore, when running the model in batch mode, we pad the shorter sequences to fit into a tensor representation. With masking, we ensure that these padded elements do not affect the model's functionality. Specifically, we remove them for the attention mechanism and loss calculation.

C Record Equality

Node equality: A node n consists of a type $\text{type}(n)$ and properties $\text{props}(n)$. Two nodes, n_1 and n_2, are equal $n_1 \overset{\epsilon}{=} n_2$ if their types match and their properties are equal (up to a precision ϵ for continuous properties):

$$\text{type}(n_1) = \text{type}(n_2) \wedge \text{props}(n_1) \overset{\epsilon}{=} \text{props}(n_2)$$

Edge equality: An edge consists of a type $\text{type}(e)$ and properties $\text{props}(e)$. Two edges, e_1 and e_2, are equal $e_1 \overset{\epsilon}{=} e_2$ if their types match and their properties are equal (up to a precision ϵ for continuous properties):

$$\text{type}(e_1) = \text{type}(e_2) \wedge \text{props}(e_1) \overset{\epsilon}{=} \text{props}(e_2)$$

Record equality: A record $r = (N, E)$ is a property graph consisting of a set of nodes N and a set of edges E, where an edge connects two nodes, a source node $n_1 = \alpha(e)$ and target node $n_2 = \beta(e)$. Two records, $r_1 = (N_1, E_1)$ and $r_2 = (N_2, E_2)$, are equal $r_1 \overset{\epsilon}{=} r_2$ if nodes, edges and the structure are preserved. This is formalized by requiring the existence of bijective functions ensuring these properties:

$$\begin{aligned}
&\exists \text{ a bijection } f : N_1 \to N_2, \\
&\exists \text{ a bijection } g : E_1 \to E_2 \text{ s.t.} \\
&\underbrace{(\forall n \in N_1, n \overset{\epsilon}{=} f(n))}_{\text{nodes preservation}} \wedge \underbrace{(\forall e \in E_1, e \overset{\epsilon}{=} g(e))}_{\text{edges preservation}} \wedge \\
&\underbrace{(\forall e \in E_1, f(\alpha(e)) \overset{\epsilon}{=} \alpha(g(e)) \wedge f(\beta(e)) \overset{\epsilon}{=} \beta(g(e))}_{\text{structure preservation}}
\end{aligned}$$

D Node dissimilarity

A record node n has a discrete type $type(n) \in \mathcal{T}$ of a finite set of types $\mathcal{T}$ and type-specific properties $props(n) = (p_1, \ldots, p_{D+C})$. The properties consist of discrete and continuous properties: $(p_1, ..., p_D)$ are D discrete properties, where each $p_k \in \mathcal{V}_k$ for a finite set of values $\mathcal{V}_k$; $(p_{D+1}, ..., p_{D+C})$ are C continuous properties, where each $p_k \in \mathbb{R}$. A node prediction $\hat{n}$ for a node n provides a predicted distribution for the type $type(\hat{n})$ and type-specific property predictions $props(\hat{n}) = (\hat{p}_1, ...\hat{p}_{D+C})$: $(\hat{p}_1, ..., \hat{p}_D)$ are D discrete property predictions, where prediction $\hat{p}_k$ is a probability distribution over the set of possible values $\mathcal{V}_k$; $(\hat{p}_{D+1}, ..., \hat{p}_{D+C})$ are the continuous property predictions, where prediction $\hat{p}_k \in \mathbb{R}$ is a point estimate. The dissimilarity $l(n, \hat{n})$ between a node n and a node prediction $\hat{n}$ is defined as the sum of the losses for the type and all properties. We use the cross-entropy loss, $\mathcal{L}_{CE}$, for categorical predictions and the squared error for continuous predictions. A weighting parameter λ indicates that these discrete and continuous loss terms must be balanced in some capacity when used together:

$$\begin{aligned}
l(n, \hat{n}) = \; & \mathcal{L}_{CE}(type(n), type(\hat{n})) \\
& + \sum_{k=1}^{D} \mathcal{L}_{CE}(p_k, \hat{p}_k) \\
& + \lambda \sum_{k=D+1}^{D+C} (p_k - \hat{p}_k)^2
\end{aligned} \tag{7}$$

E Practical considerations

In this section, we take a more applied viewpoint on document-as-transcription and discuss how our insights directly affect practical considerations for building document recognition systems.

Architectural implications. The document-as-transcription perspective highlights that document recognition is a many-to-one, document-to-record task. Therefore, we can model the task as deterministic function approximation. This avoids the target ambiguity and complex training dynamics typical of ill-posed inverse problems, where a single observation often maps to multiple valid interpretations. Furthermore, as contrasted with graph generation (see Section 2.2), this deterministic mapping frees the architecture from the heavy probabilistic modeling required to learn distributions over arbitrary graphs, allowing it to optimize purely for recovering a unique target structure.

As a practical side-benefit of our document-centric focus, we exploit the inherent sparsity of convention-bound documents. By employing a vision transformer for image encoding, we can safely drop empty background patches, significantly reducing computational overhead.

Record schema design. For any specific document type, the precise record schema must be defined, detailing the exact semantic entities, relations, and associated properties. The transcription lens guides this schema design by strictly separating semantic content from extraneous visual artifacts. For existing record-aligned representations (e.g., Kern syntax, LaTeX), our perspective motivates normalizing these representations to better align with the record's uniqueness and removing rendering-only information from the ground truth, as it is part of the synthesis process, not the record (cf. [7]).

Crucially, not all structured formats constitute a record. For instance, Scalable Vector Graphics (SVGs) describe visual geometry rather than semantic topology. A single drawn line in an SVG might represent multiple distinct semantic edges (or vice versa); thus, image vectorization [93] is fundamentally distinct from document transcription. Vectorization recovers visual paths, whereas transcription requires extracting domain-specific semantics.

Visual diversity and robustness. A practical document recognition system must handle a wide array of visual styles, including pristine digital renderings, noisy scans, different fonts, or hand-drawn variations. Within our proposed framework (see Fig. 4), such stylistic diversity is formalized as the application of distinct, valid synthesis functions. Consequently, this diversity increases the variance of the input distribution without fundamentally altering the underlying transcription task. So, achieving robustness becomes primarily an engineering challenge of implementing approximate synthesis pipelines for data augmentation and adequately scaling the model's capacity (and compute).

Document and record interpretation. Our perspective strictly delineates document-to-record transcription from downstream interpretation. Crucially, the rasterized image and the symbolic record are informationally equivalent representations of the same underlying content. Therefore, transcription is purely a modality translation task; it does not involve interpreting this information for downstream applications, such as cost prediction for engineered parts (cf. [94]). Meaningful document interpretation occurs subsequent to transcription, typically by applying algorithmic reasoning or graph neural networks directly to the extracted symbolic record, or potentially to the learned image embeddings.

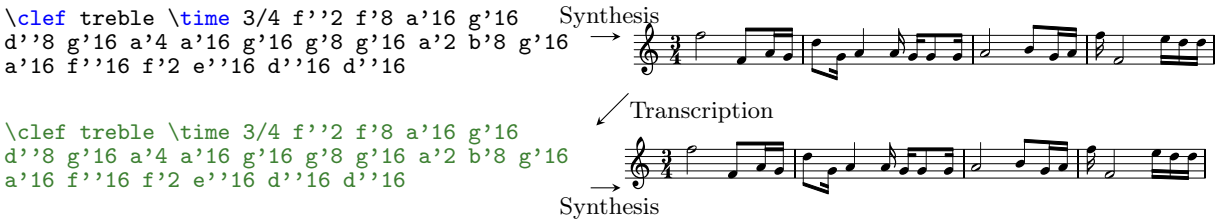
However, we expect that semantically meaningful information is much more explicitly available in the transcribed record than the original image or any manually created partial information extraction, and that some of that semantic information will also be available in the learned internal representations of encoded document images, making them potentially useful for semantic retrieval etc.

Non-convention-bound document transcription. For non-convention-bound documents, Equation (1) no longer holds, as multiple interpretations about the semantics may be valid for a document. One could still try to apply the document-as-transcription framework; however, it loses the architectural implications above, meaning that training a deterministic input-output mapping could be conceptually impossible. Furthermore, a record schema definition may become impossible, as the document loses its intent to convey full information precisely.

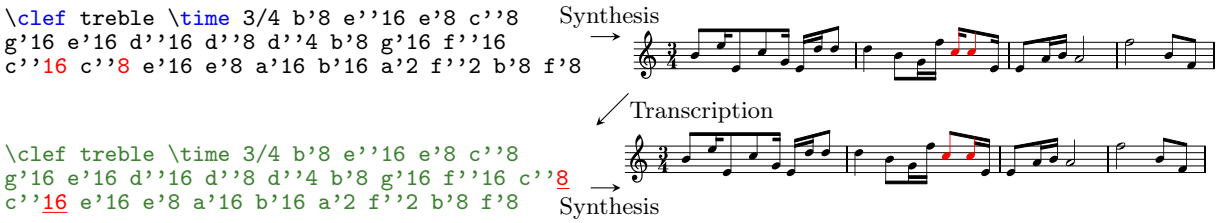
While one could argue that even strict conventions might occasionally contain ambiguities that violate Equation (1), we consider these instances practical anomalies rather than systemic flaws. Such edge cases represent failures in the convention, as any ambiguity that confounds a deterministic transcription model would identically hinder human-to-human communication.

F Supplementary examples

Fig. F.1 shows the synthesis and transcription process for sheet music. Fig. F.2 shows examples of successfully transcribed sheet music. Fig. F.3 shows examples of incorrectly transcribed sheet music. Fig. F.4 shows the synthesis and transcription process for shape drawings. Fig. F.5 shows examples of shape drawings. Fig. F.6 shows examples of correctly transcribed shape drawings. Fig. F.7 shows examples of incorrectly transcribed shape drawings. Fig. F.8 shows examples of simplified engineering drawings.



(a) Correct transcription.



(b) Incorrect transcription (errors highlighted with red underlined text).

Fig. F.1: Example of a correct (a) and an incorrect transcription (b). Both start with the true record (top left, written in LilyPond syntax) and rendering (top right). Then, the model predicts a record (bottom left, written in LilyPond syntax), which is rendered for visual comparison (bottom right).

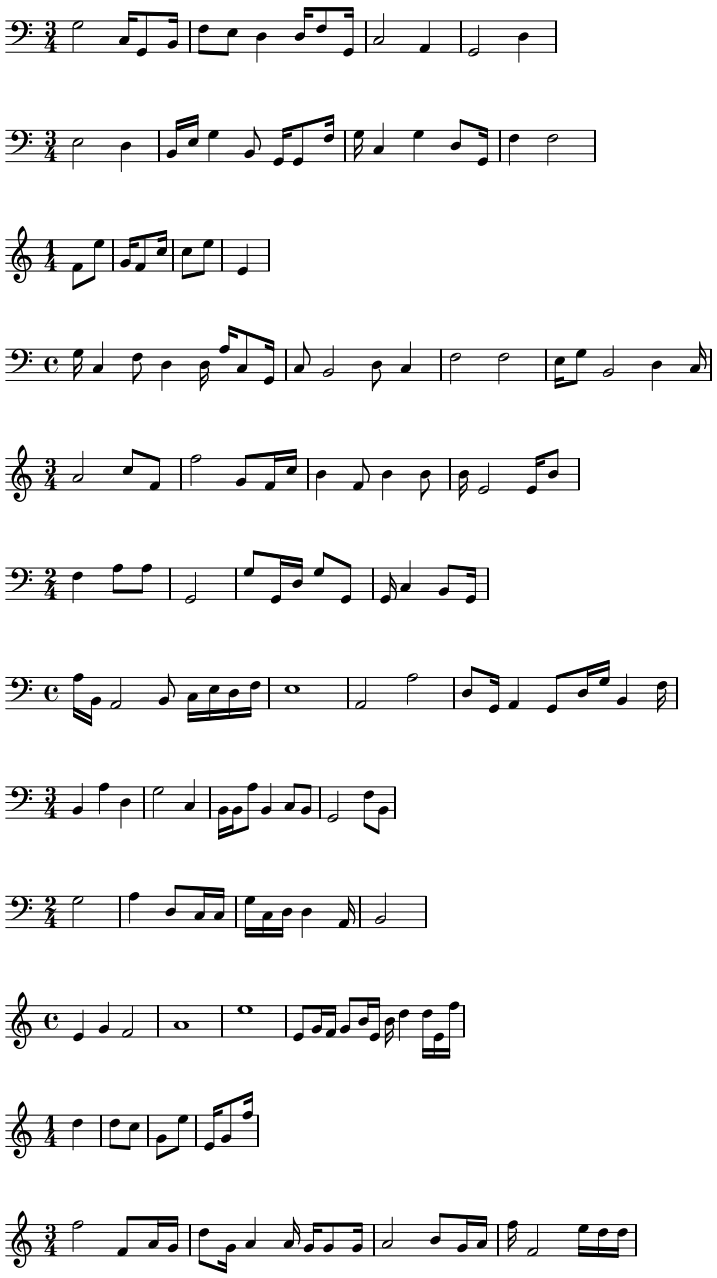


Fig. F.2: First 12 “sheets” of music of the randomly generated validation set. The model exactly transcribes all of them.

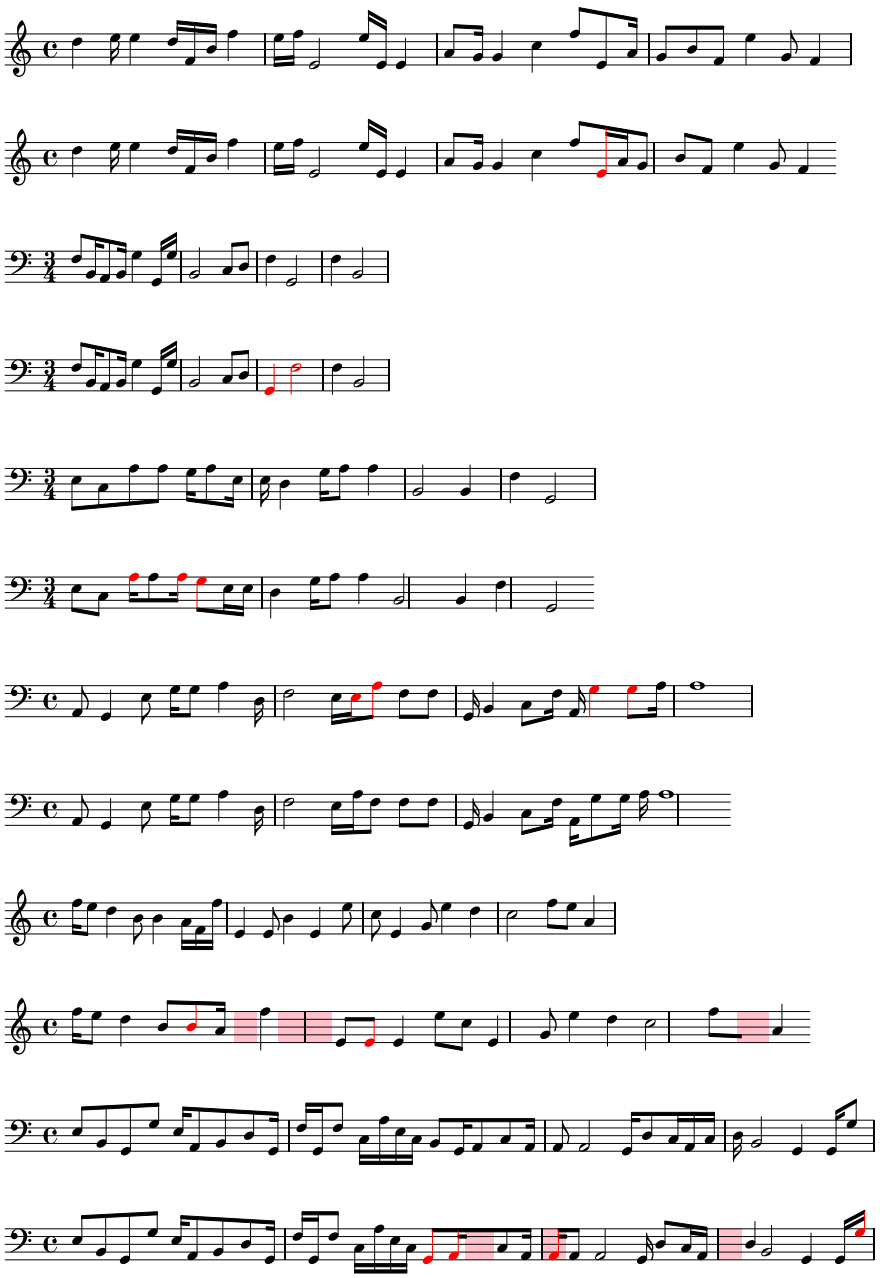
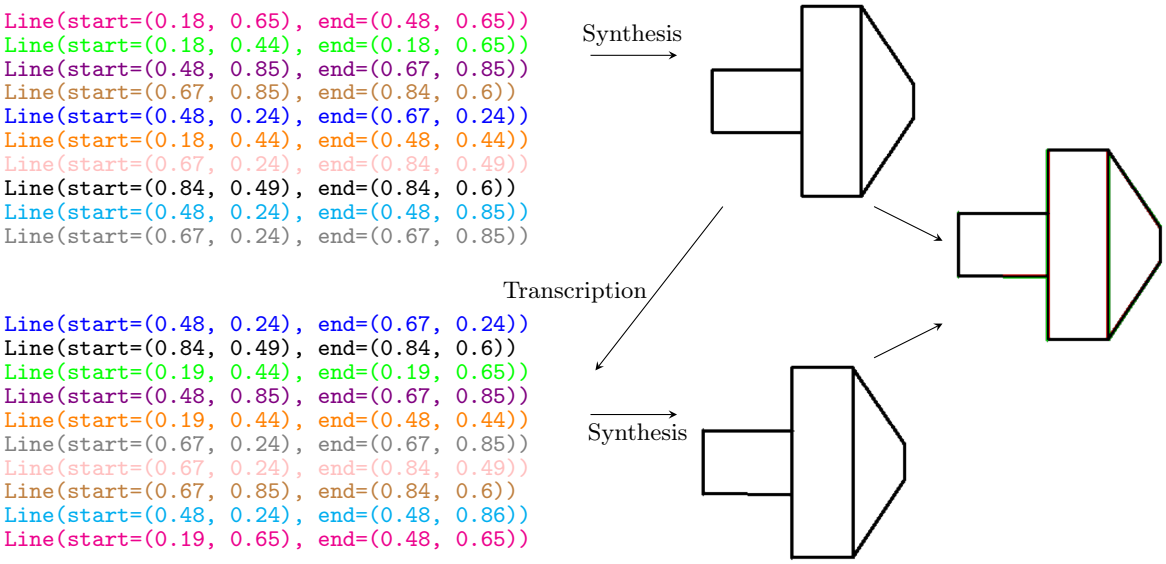
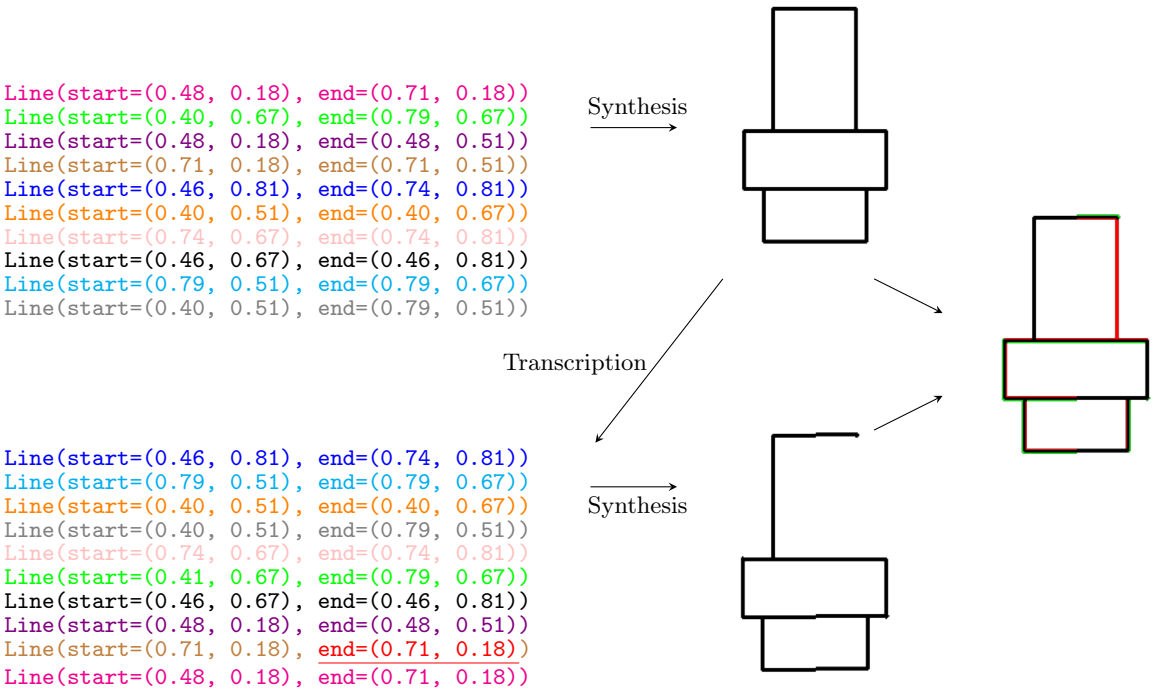


Fig. F.3: Six examples of incorrect transcription increasing in Levenshtein distance. First, the true sheet music is shown, followed by the corresponding prediction. In the predictions, errors are marked like this: a red note without a highlight means at least one music note property was wrong (replace), an empty red highlight means the prediction had a music note missing (delete), a red note in red highlight means the prediction had an extra music note (insert).



(a) Correct transcription.



(b) Incorrect transcription (errors highlighted with red underlined text).

Fig. F.4: Example of a correct (a) and an incorrect (b) transcription of shape drawings. In both cases, we start with the true record (top left) and render it (top right). Then, the model predicts a record (bottom left), which we render for easier comparison (bottom right). Finally, we render the prediction in green on top of the true shape in red, and perfect overlapping matches in black (middle right).

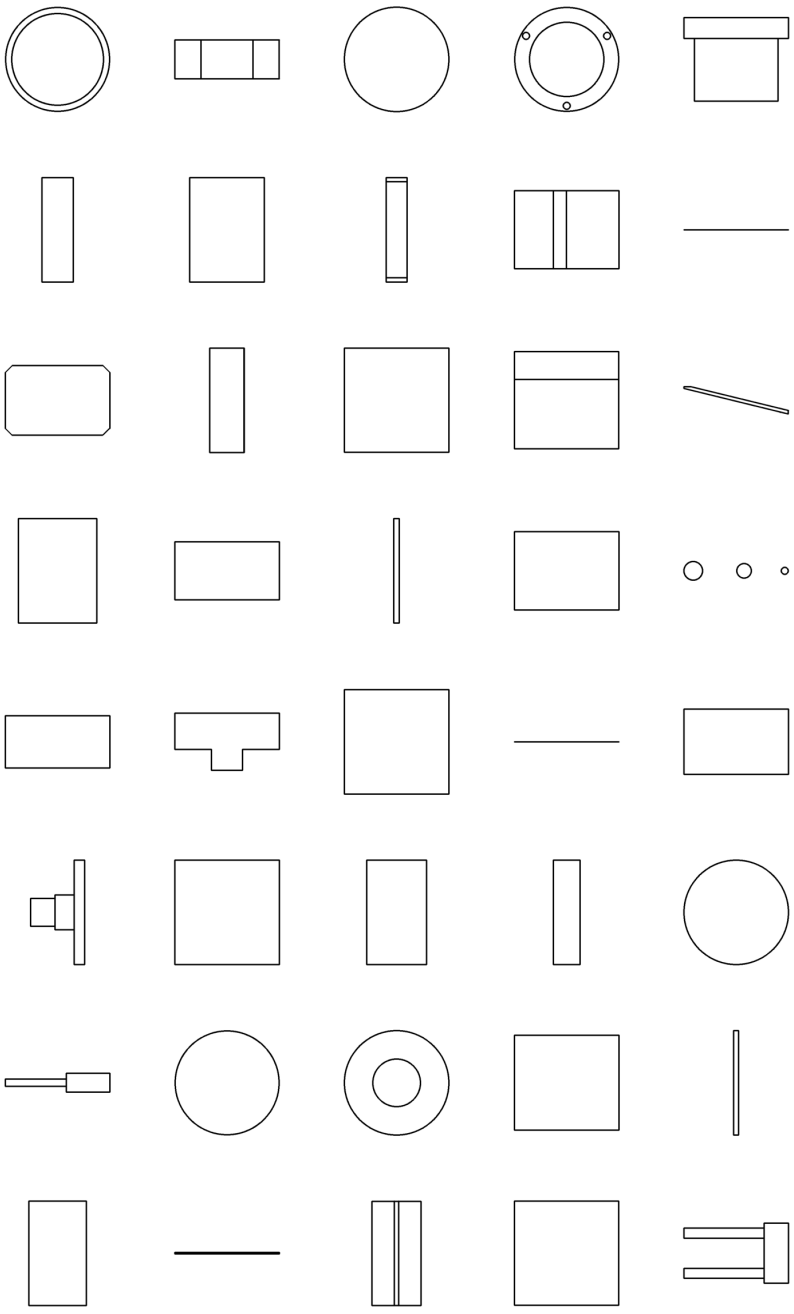


Fig. F.5: Examples of 40 engineering drawings. One cell represents one example.

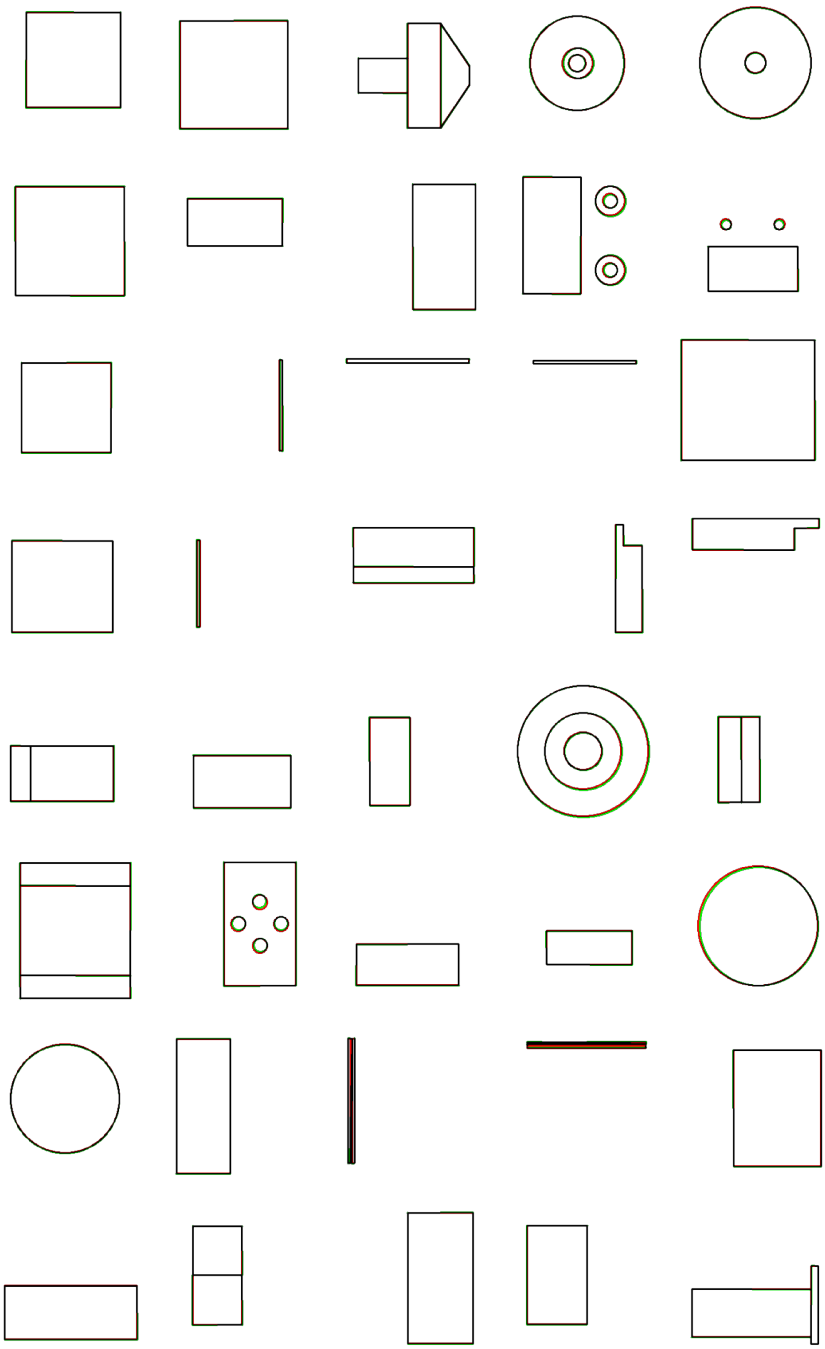


Fig. F.6: First 40 examples of correct transcriptions in the validation set. The ground truth is drawn in red, the prediction in green. If overlapping exactly, the line is drawn in black.

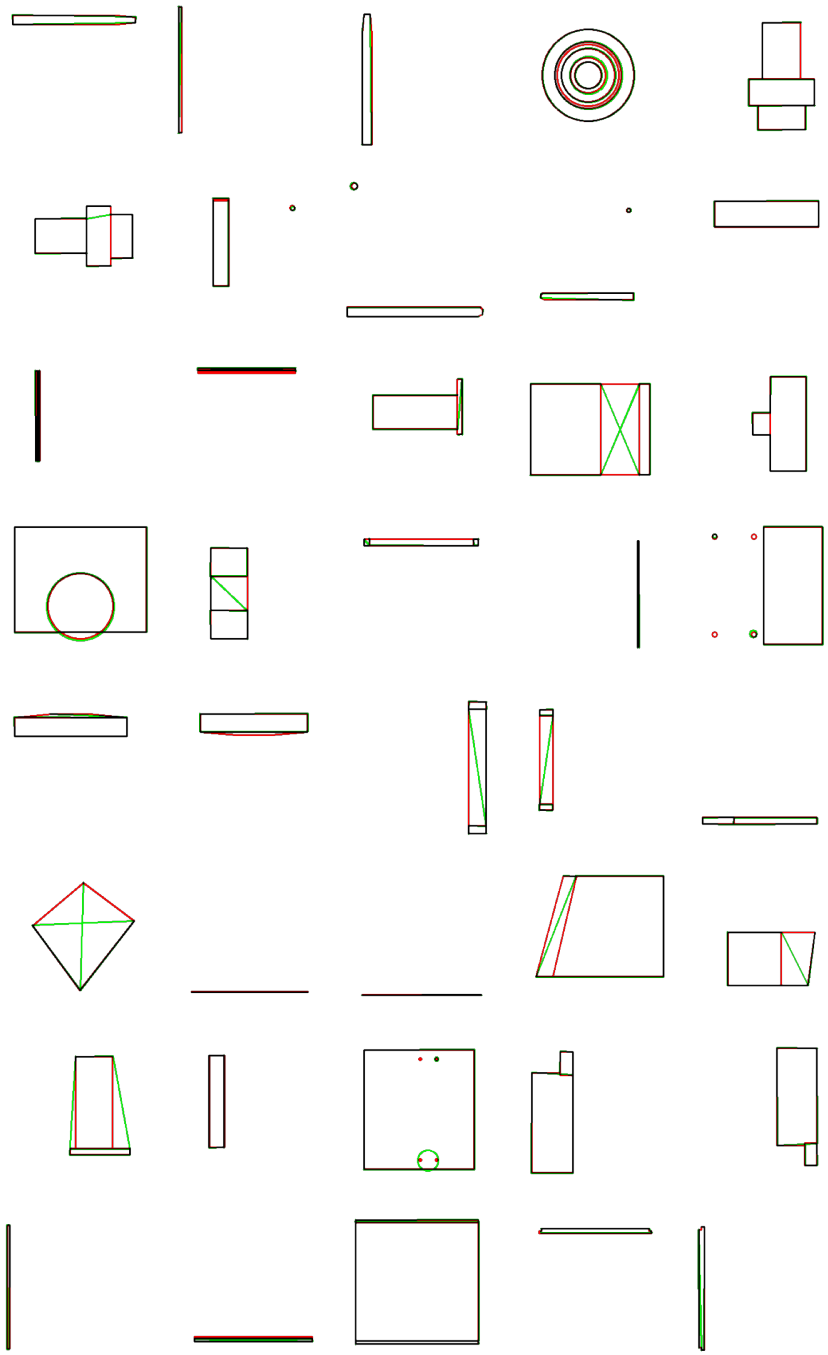


Fig. F.7: First 40 examples of failed transcriptions in the validation set. The ground truth is drawn in red, the prediction in green. If overlapping exactly, the line is drawn in black.

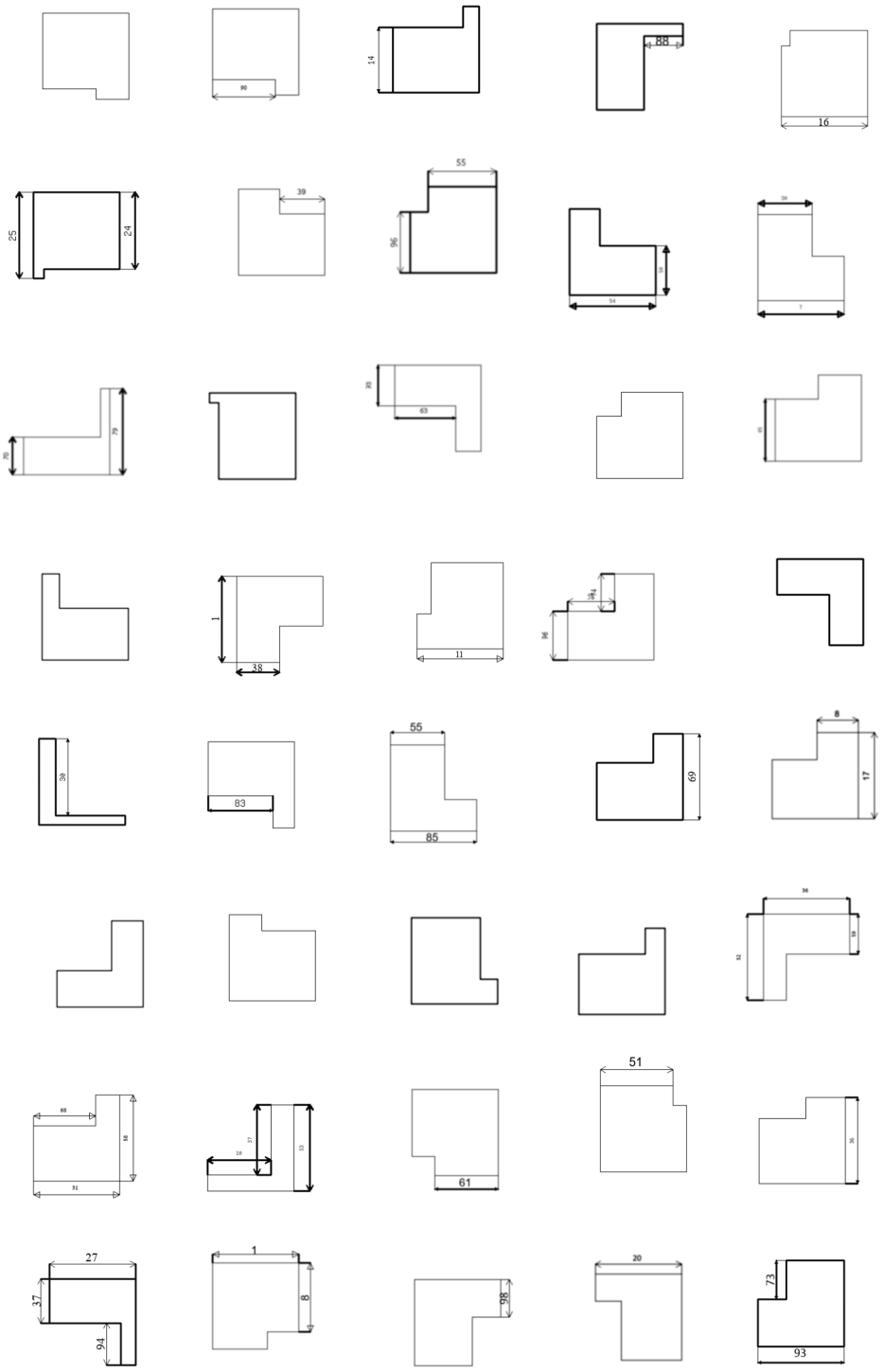


Fig. F.8: Examples of synthetically generated engineering drawings.