Data Engineering Specification: Proprietary CDR to Latent-Ready Tensors

RUSHABH LODHA

November 30, 2025

Abstract

This report defines the strict ETL (Extract, Transform, Load) procedure required to convert unstructured CorelDRAW (`.cdr`) archives into normalized, fixed-dimensional tensors suitable for the *Architecture C* Encoder. The process involves headless vector extraction, topological filtering, mathematical canonicalization of primitives, and final tensor serialization. All operations utilize open-source libraries.

1 Phase 1: Headless Artifact Decoupling

Objective: Conversion of proprietary binary schemas into accessible XML-based SVG standards without GUI intervention.

1.1 1.1 The Inkscape CLI Bridge

We utilize `Inkscape` (v1.2+) as a headless backend. Unlike reverse-engineered parsers (e.g., `libcdr`), Inkscape ensures accurate rendering of affine transformations present in the source file.

Execution Protocol:

```
# Inkscape CLI Command for Layer-Preserving Extraction
inkscape --export-type="svg" \
         --export-plain-svg \
         --export-filename="output_buffer.svg" \
         "input_archive.cdr"
```

2 Phase 2: Semantic Filtering & Extraction

Objective: Isolate structural geometry (Cut/Crease) from decorative elements (Graphics/Text).

2.1 2.1 Heuristic Filtering Logic

Packaging dielines typically encode structural semantics via spot colors. We parse the SVG XML tree to extract paths based on stroke properties.

Mathematical Set Definition: Let $S_{raw} = \{p_1, p_2, \ldots, p_n\}$ be the set of all paths in the SVG. We define filters Φ_{cut} and Φ_{crease}:

$$P_{structure} = \{p \in S_{raw} \mid \text{color}(p) \in \mathcal{C}_{target} \lor \text{id}(p) \in \mathcal{I}_{target}\}$$

Where $\mathcal{C}_{target}$ includes standard dieline colors (e.g., Cyan $\approx (0, 255, 255)$, Magenta $\approx (255, 0, 255)$).

3 Phase 3: Mathematical Canonicalization

Objective: Architecture C requires a homogeneous input space. All geometric primitives (Arcs, Lines, Polygons) must be converted into a unified **Cubic Bézier** representation.

3.1 3.1 Primitive Homogenization

A Cubic Bézier curve $B(t)$ is defined by 4 control points P_0, P_1, P_2, P_3.

Case A: Line to Bézier A linear segment from L_0 to L_1 is represented as:

$$P_0 = L_0, \quad P_1 = \frac{2L_0 + L_1}{3}, \quad P_2 = \frac{L_0 + 2L_1}{3}, \quad P_3 = L_1$$

Case B: Quadratic to Cubic Given a Quadratic Bézier with control points Q_0, Q_1, Q_2, the equivalent Cubic points are:

$$P_0 = Q_0, \quad P_1 = Q_0 + \frac{2}{3}(Q_1 - Q_0), \quad P_2 = Q_2 + \frac{2}{3}(Q_1 - Q_2), \quad P_3 = Q_2$$

Case C: Elliptical Arc to Bézier Arcs must be approximated. We use the split strategy: if the sweep angle $\theta > 90°$, split the arc. For $\theta \leq 90°$, the approximation error is minimized using $\kappa = \frac{4}{3}\tan(\theta/4)$.

3.2 3.2 Spatial Normalization

To ensure the latent space is scale-invariant, all coordinates must be mapped to the unit square $[0, 1]^2$.

Let $V \in \mathbb{R}^{N \times 2}$ be the set of all vertices in a file. 1. **Translation:** Center the geometry.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} v_i, \quad V' = V - \mu$$

2. **Scaling:** Scale by the maximum dimension.

$$s = \max_{v \in V'}(||v||_\infty), \quad V_{norm} = \frac{V'}{2s} + 0.5$$

4 Phase 4: Tensor Serialization (The VAE Input)

Objective: Construct the final tensor X for the model.

4.1 4.1 Path Reordering (Spatial Sorting)

To aid the autoregressive nature of the model, paths should be sorted deterministically. We minimize the Hamiltonian path distance or use a simple lexicographical sort based on the bounding box centroid c_i:

$$\text{sort}(P) \text{ such that } c_i^y < c_{i+1}^y \lor (c_i^y = c_{i+1}^y \land c_i^x < c_{i+1}^x)$$

4.2 4.2 Feature Matrix Construction

Each curve segment i is encoded as a vector $x_i \in \mathbb{R}^{14}$.

Feature Definition:

$$x_i = [\underbrace{sx, sy}_{\text{Start}}, \underbrace{cx1, cy1}_{\text{Control 1}}, \underbrace{cx2, cy2}_{\text{Control 2}}, \underbrace{ex, ey}_{\text{End}}, \underbrace{v_{vis}}_{\text{Visibility}}, \underbrace{v_{cmd}}_{\text{Command Type One-Hot (3)}}]$$

- v_{vis}: Binary flag (1 if pen is down, 0 if pen is up/move).
- v_{cmd}: One-hot vector for [Line, Curve, Move].

Final Tensor Shape:

$$\mathbf{X} \in \mathbb{R}^{N_{max} \times 14}$$

Where N_{max} is the fixed sequence length (e.g., 128 or 256). Sequences shorter than N_{max} are zero-padded; longer sequences are truncated or split.

5 Implementation Tools (Open Source)

- **SVG Parsers:** `svgpathtools` (for math operations), `xml.etree` (for structure).
- **Matrix Ops:** `NumPy`.
- **Geometry:** `Shapely` (for bounding boxes and intersections).
- **Deep Learning specific:** `DeepSVG` (Data loader utilities).