Annotations and Domain Randomisation

Almost every published segmentation result starts from a labelled dataset. Raster well-log digitisation does not get to: the archive is decades of scanned paper at every scale and condition, and nobody ever sat down to annotate which pixels belong to which curve. There is no corpus to learn from, and hand-labelling one at the scale a segmentation network needs is the exact bottleneck the project exists to remove. Our answer was to manufacture the dataset with a procedural log generator, and this whitepaper is the design account of that engine treated as a system in its own right. We describe the geometry layer that synthesises curve traces directly, so that a pixel-exact mask is produced by construction rather than by annotation; the annotation layer that stamps the grid lines, depth ticks, track separators and printed labels a real scanned log carries, so the network learns to ignore them; and the domain-randomisation layer whose knobs - image width from 3,200 to 12,800 pixels, height from 480 to 640 pixels, the number of curves per log, a single grayscale channel, and a tunable noise and realism level - sweep every render across the same distribution the real archive spans. The design discipline is that breadth of randomisation, not volume of hand labels, is what makes the dataset cover the problem: a generator dialled wide manufactured a 20,000-log two-curve multiclass corpus, and the final version-two training set drew 15,000 curves from it, none of them touched by a human annotator. We cover why synthesising the mask beats annotating one for a one-pixel-wide target, how the annotation layer is chosen to be adversarial to the model rather than decorative, which randomisation knobs matter and why each one is bounded by a real archive statistic, and the discipline that keeps a synthetic corpus from teaching the network the generator's own habits instead of the geoscience.

Narendra Patwardhan
Research Collaborator

Tannistha Maiti
Senior AI Researcher

Quamer Nasim
ML Research Engineer

VEERNET / PROCEDURAL-GENERATION / SYNTHETIC-DATA / DOMAIN-RANDOMISATION / DATA-SYNTHESIS / RASTER-LOG-DIGITISATION

CONTENTS

Most machine-learning projects begin with a dataset and design a model for it. Raster well-log digitisation begins with the opposite problem: there is no dataset, there is no realistic prospect of one, and the absence of one is the entire reason the project exists. The raw material is decades of scanned paper logs sitting in operator archives, every one of them an image with all the information of the original log and none of its queryability, and not one of them carries a label that says which pixels are the gamma-ray trace and which are the grid line behind it. A segmentation network needs that label for thousands of images. Nobody ever produced it, and producing it by hand at the required scale is precisely the manual bottleneck the whole effort is meant to remove. You cannot annotate your way out of an annotation problem.

So we did not annotate. We built an engine that manufactures the dataset, and this whitepaper is the design of that engine taken seriously as a system rather than as a preprocessing step. The idea is old and well-founded: if you generate the image, you already know everything about it, so the label is free, and if you randomise the generator's nuisance parameters widely enough the real world becomes, to the model, just one more sample from the synthetic distribution [1]. The contribution here is not the idea but the design of the specific generator that makes it work for printed curves on scanned paper, and the discipline that keeps a manufactured corpus honest. The downstream architecture, an encoder-decoder segmentation network in the U-Net lineage [3], is upstream context for this document and not its subject. The subject is the data factory.

Why synthesise the mask instead of annotating one

The reflexive plan is to collect real scans, pay annotators to trace the curves, and train on that. For this target it fails on two counts before cost even enters the conversation.

The first is resolution. The thing the network has to find is a curve trace that, on the source raster, is frequently a single pixel wide. A human annotating that trace clicks a polyline through it, and the polyline is an approximation of a one-pixel feature drawn by a hand that cannot reliably resolve one pixel. The label you buy is systematically fuzzier than the signal it is supposed to describe, and a sparse, hairline target punishes that fuzz harder than any natural-image task would. The second is consistency. Two annotators tracing the same degraded log produce visibly different curves, and a network trained on the union learns the disagreement as much as the curve.

A procedural generator inverts both problems. Because the geometry layer draws the curve, the mask is not an annotation of the image, it is the same parametric object the image was rendered from, recorded at exactly the pixels the renderer set. The label is pixel-exact by construction and perfectly consistent across the entire corpus, because the same code drew every one. You are not approximating a one-pixel feature after the fact, you are the authority that placed it. For a one-pixel-wide target that distinction is the difference between a usable mask and a smeared one, and it is the first reason the engine is worth building rather than buying.

The whitepaper's first decision is to synthesise the mask rather than annotate one, and this is that decision made physical. The target a label must describe is a curve trace that on the source raster is often a single pixel wide. Switch the toggle between two labellers of that same one-pixel trace. The human annotator clicks a polyline an unsteady hand cannot snap to one pixel, so the mask is fuzzy and a second annotator disagrees on several cells; a sparse hairline target punishes exactly that fuzz. The procedural generator is the authority that placed the pixel, so its mask is the same parametric object the image was rendered from, pixel-exact by construction and identical across every render. The right-hand ledger is what the decision shipped: a 20,000-log two-curve multiclass corpus, a 15,000-curve final v2 training set, and the figure that carries the whole argument, zero human-annotated logs anywhere in the pipeline. Sourced: the one-pixel trace target, the 20,000-log corpus, the 15,000 v2 curves, and the zero hand labels. The pixel-grid render, the hand-polyline wobble, and the two-annotator disagreement overlay are illustrative schematics of the labelling failure mode, flagged on the canvas.

The geometry layer: curves as parametric objects

The geometry layer is where a synthetic log starts. Each curve is a parametric trace laid down across a depth axis, with controlled smoothness, amplitude and crossing behaviour, drawn at the native resolution of the target image rather than resized into it. Drawing at native resolution is not an aesthetic choice. A well log encodes value as horizontal deflection against depth, so the shape of the trace is the signal, and any resampling step that smooths a one-pixel line into its neighbours destroys the very thing the network is trained to recover. The generator therefore renders the curve and rasterises its mask in the same pass at the same resolution, and the two are guaranteed to agree pixel for pixel because they come from one parametric source.

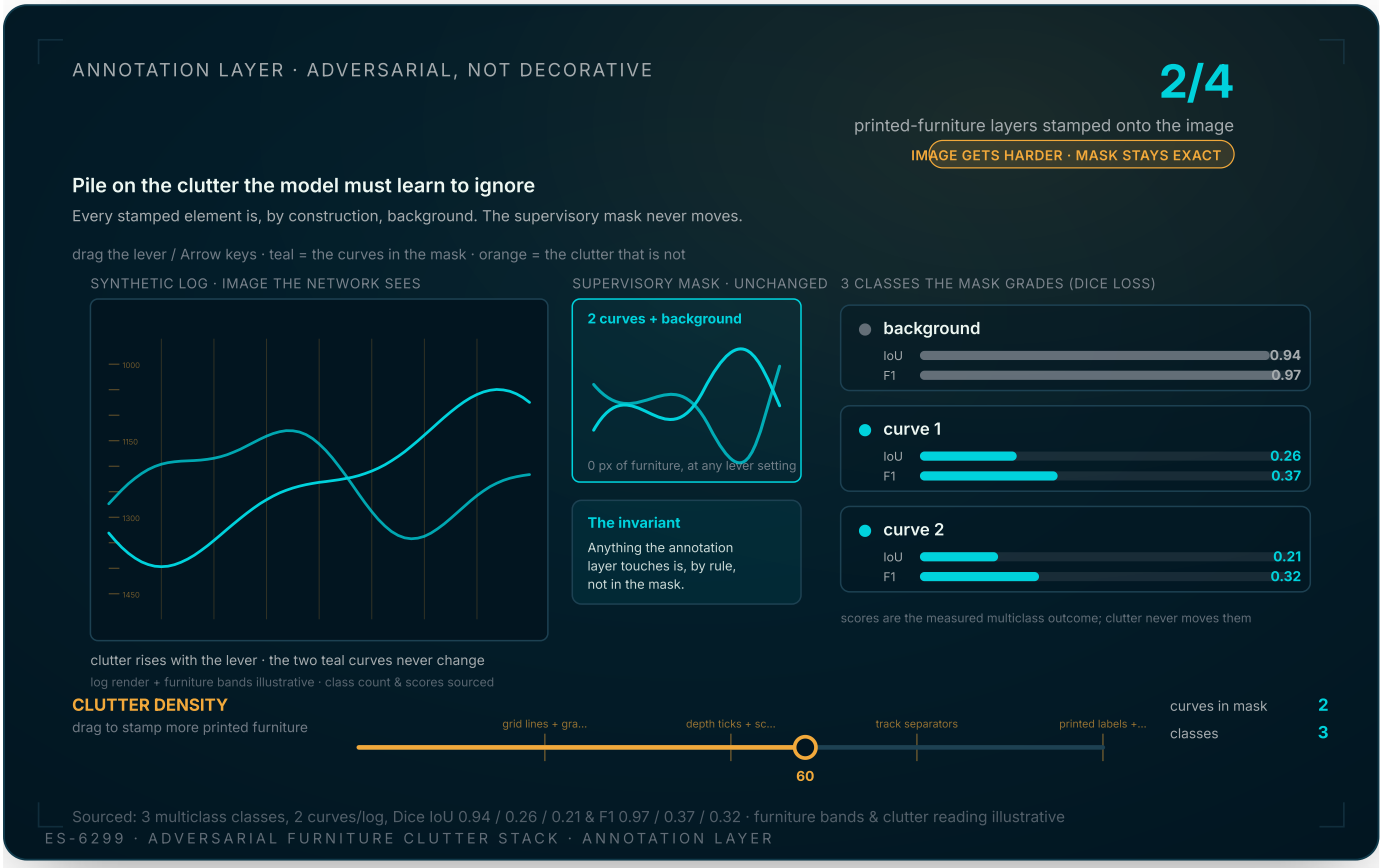
The number of curves on a log is itself a parameter, and it is the parameter that separates the two training regimes. The binary stage drew a single foreground trace against background. The final multiclass stage fixed **two curves per log** against background, three classes in total, because two crossing curves is where the genuinely hard problem lives: the network must keep two thin traces distinct through the regions where they overlap, and a generator that can place two curves with controlled crossing geometry can manufacture exactly the ambiguous cases a real archive would only supply by luck. The input is a **single grayscale channel**, matching the scanned-paper source, so the generator never invents colour information the real logs do not have.

The annotation layer: furniture the model must learn to ignore

A clean curve on an empty field is not a scanned log, and a network trained on clean curves learns nothing about the clutter that actually surrounds a real trace. The annotation layer exists to put that clutter back, deliberately. It stamps the printed furniture every real log carries: the grid lines and graticule, the depth ticks and scale numbers, the track separators that divide the log into columns, and the printed curve labels and header text. None of it is decoration. Every element is chosen because it is something the model would otherwise mistake for signal, and the only way to teach a network to ignore a grid line is to show it ten thousand grid lines that are never part of the answer.

The design rule for this layer is that it should be adversarial to the model rather than flattering to the eye. A grid line that runs parallel to a curve, a depth tick that clips the trace, a printed label that sits exactly where a curve passes behind it, these are the cases that

break a naive model, so they are exactly the cases the annotation layer is tuned to produce often. Crucially, because the geometry layer already fixed the mask, every annotation the layer adds is by definition background. The grid line is drawn into the image and not into the mask, so the supervisory signal stays clean no matter how busy the render becomes. The image gets harder; the label stays exact. That invariant, that anything the annotation layer touches is guaranteed not to be in the mask, is what lets us make the synthetic images arbitrarily cluttered without ever corrupting the ground truth.

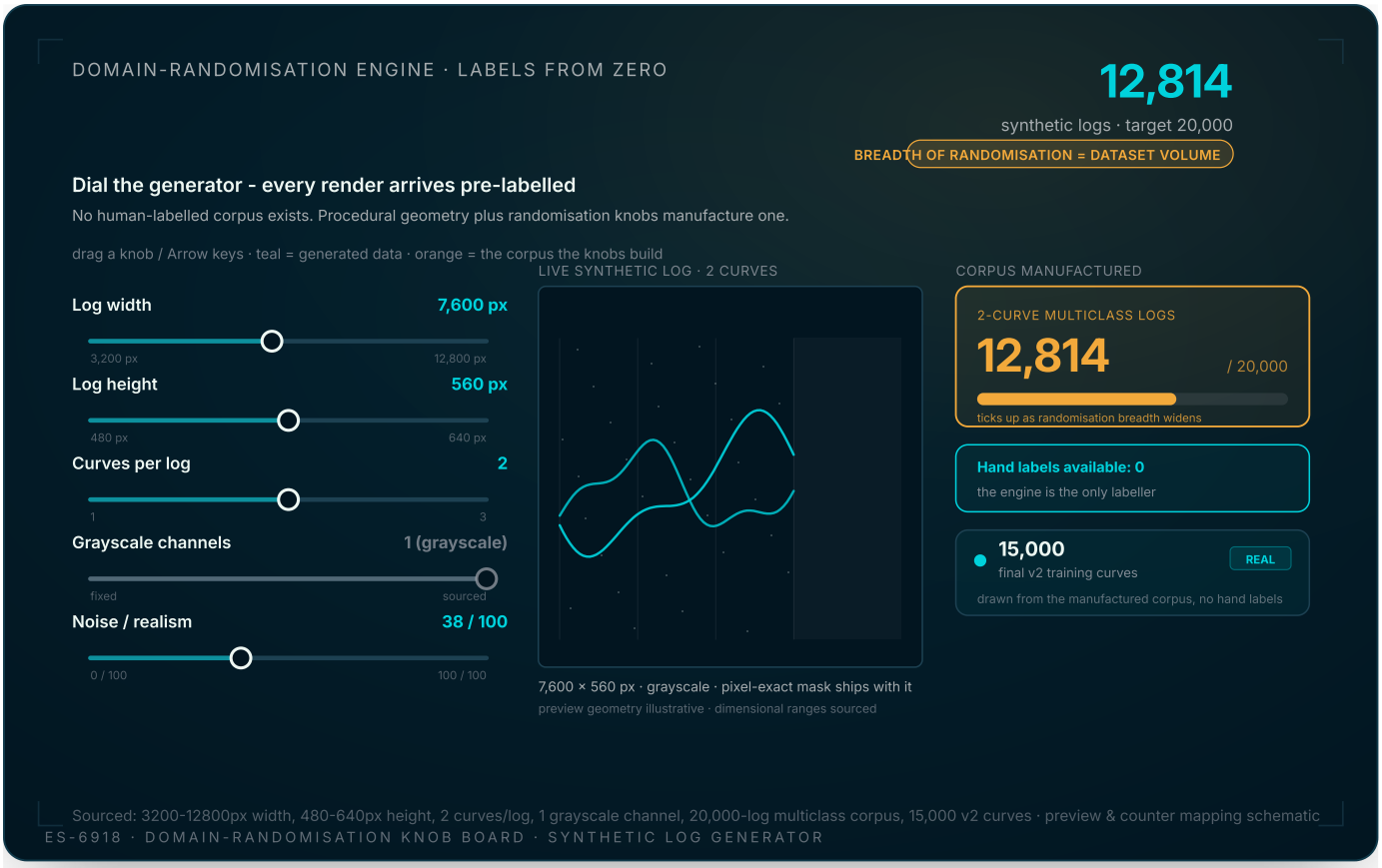


The annotation layer is the whitepaper's least intuitive claim: the clutter on a synthetic log is added on purpose, tuned to be adversarial to the model rather than flattering to the eye, and it is governed by one invariant. Drag the lever and four printed-furniture bands stack onto the log in the order the text lists them, grid lines and graticule, depth ticks and scale numbers, track separators, then printed labels and header text. The image the network sees gets visibly harder, yet the supervisory mask, the two curves against background, does not change a single pixel, because every stamped element is drawn into the image and never into the mask. The right column reports the three multiclass classes the network keeps apart under that clutter (background, curve 1, curve 2) with their measured Dice-loss per-class scores, which stay fixed because clutter cannot corrupt a mask it never touches. Sourced: the three classes, the two curves per log, and the per-class IoU 0.94 / 0.26 / 0.21 and F1 0.97 / 0.37 / 0.32. The four furniture bands, the clutter difficulty reading, and the rendered log tile are illustrative schematics of the annotation layer, flagged on the canvas.

The domain-randomisation layer: knobs, not realism

The geometry and annotation layers can produce one convincing log. They cannot, on their own, produce a dataset that covers the real archive, because a real archive is not one log, it is a distribution of logs printed at different scales by different service companies across decades and scanned on different equipment. Covering that distribution is the job of the domain-randomisation layer, and the design insight that governs it is the one the synthetic-data literature arrived at the hard way: the goal is not to make any single synthetic image photorealistic, it is to make the *distribution* of synthetic images wide enough that the real distribution falls inside it [1][2]. Deliberately varied, even deliberately unrealistic, synthetic data that spans the nuisance parameters widely beats carefully photorealistic data that spans them narrowly, because breadth is what closes the gap between synthetic and real, not fidelity.

Concretely, the layer exposes a set of knobs, and the discipline is that every knob's range is pinned to a statistic of the real archive rather than chosen for convenience. Image **width is randomised from 3,200 to 12,800 pixels** and **height from 480 to 640 pixels**, because that is the dimensional span the real scanned logs actually occupy; the generator is not allowed to draw a log narrower or wider than something the archive contains. The number of curves, the single grayscale channel, and a tunable noise and realism level that controls scan speckle, ink density and degradation are the other knobs. The knob board below is that layer made interactive: dial each parameter and the synthetic log regenerates live, while a counter ticks toward the corpus the breadth manufactures.



The synthetic-data engine is itself a designed system. With no human-labelled corpus to learn from, the generator manufactures its own ground truth: procedural geometry draws each curve and a set of domain-randomisation knobs spreads every render across the real archive's dimensional range. Dial each knob - image width 3200-12800px, height 480-640px, curves per log, grayscale channel depth, and noise / realism level - and the live synthetic-log preview regenerates while the corpus counter ticks toward the 20,000-log two-curve multiclass dataset and marks the 15,000 final v2 training curves. That is the argument: the breadth of randomisation is what manufactures dataset volume, standing in for the zero hand labels that never existed. Sourced: the 3200-12800px width range, the 480-640px height range, 2 curves per log in the final multiclass dataset, the 1 grayscale input channel, the 20,000-log two-curve multiclass corpus, and the 15,000 final v2 training curves. The live preview render and the way a single knob maps to a counter increment are schematic depictions of the generator's behaviour, both flagged on the canvas.

The point the knob board makes physically is the point of the whole design. Widen the knobs and the generator covers more of the real distribution per render, which is why breadth, not a labelling budget, is the lever that sets how much of the problem the dataset represents. A generator dialled narrow produces a large pile of nearly-identical logs that teach the network one corner of the archive. A generator dialled wide produces a corpus that spans it. The volume follows almost for free, because every render is pre-labelled, so the only cost of another sample is the compute to draw it.

From knobs to corpus: 20,000 logs, zero hand labels

Run the wide-dialled generator and the dataset materialises at whatever scale the compute budget allows, every instance arriving with its pixel-exact mask attached. The engine manufactured a **20,000-log two-curve multiclass corpus** this way, and the final version-two training set drew **15,000 curves** from it. The number that matters most is the one that is not in those figures: the count of human-annotated logs underneath them is **zero**. There was no corpus to start from, and there is none now either, in the sense that no person ever labelled a single training image. The labels were manufactured alongside the pixels, by the same code, at the same resolution, across the same dimensional range the real archive spans.

That is the design payoff stated plainly. A segmentation network that could not have been trained at all, for want of a labelled dataset, is trained on a dataset that did not exist until the generator drew it. The sparse-foreground imbalance the two-curve logs create, two thin traces against a vast background, is the kind of imbalance a Tversky-style objective is built to handle [4], but the imbalance only becomes a tractable training problem because the generator can produce as many hard, crossing-curve examples as the optimiser needs. The data factory does not merely fill a gap left by missing labels; it gives the training process a controllable supply of exactly the cases that are hardest to learn.

Keeping a manufactured corpus honest

A generated dataset has a failure mode that a collected one does not: the network can learn the generator instead of the geoscience. If every synthetic curve is a little too smooth, if the noise is always the same flavour of speckle, if the grid lines always sit on the same pitch, the model will quietly overfit to those tells and then meet a real scan that has none of them. The design discipline that guards against this is the reason the randomisation knobs are bounded by archive statistics rather than by taste, and the reason the annotation layer is tuned to be adversarial rather than tidy. The defence against learning the generator is to make the generator's output as varied, within the real distribution's support, as the real distribution itself.

Three habits hold the line in practice. The first is that every knob range is justified by a measured property of the real archive, so the synthetic distribution cannot drift outside the real one's support, and the width and height bounds are the clearest example. The second

is that realism is spent on the things the model can exploit, the clutter and degradation the annotation and noise layers introduce, rather than on cosmetic detail the model never sees. The third is validation against real scans, the held-out reality check that tells you whether the breadth you dialled actually covers the target or merely looks like it should. A synthetic corpus is only as good as its coverage of the real distribution, and coverage is something you verify, not something you assume because the renders look convincing.

What the engine is, and what comes after it

Treated as a system, the procedural log generator is three cooperating layers with one invariant. The geometry layer draws curves as parametric objects and hands back a pixel-exact mask for free. The annotation layer stamps the printed furniture a real scan carries and is guaranteed, by the invariant, never to touch the mask. The domain-randomisation layer sweeps every render across the real archive's dimensional range through knobs bounded by real statistics, and it is the breadth of that sweep, not any labelling budget, that decides how much of the problem the dataset covers. Out of those three layers came a 20,000-log multiclass corpus and a 15,000-curve final training set with no human annotation anywhere in the pipeline.

The levers still open are the obvious extensions of the same design. A richer geometry layer can synthesise more curve types and more deliberately pathological crossings, expanding the catalogue of hard cases the optimiser can be fed on demand. The annotation layer can grow to cover rarer furniture, the hand-written marginalia and stamps and stains that the long tail of the archive carries. And the randomisation knobs can be widened further as new archive statistics are measured, always inside the discipline that a knob's range is a measured property of reality and not a guess. The model is downstream of all of it, and it was only ever trainable because the engine underneath it manufactured, rather than waited for, the labelled dataset the field assumes you already have.

Get the full whitepaper

This page is the long-form summary. The complete whitepaper includes the full layer-by-layer generator specification, the parameter ranges and their archive justifications, the annotation catalogue, the synthetic-versus-real coverage validation, and the corpus-generation throughput numbers.

References

[1] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P. (2017). Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. IROS. The principle that randomising a generator's nuisance parameters widely enough makes the real world look like just another sample of the synthetic distribution. https://arxiv.org/abs/1703.06907

[2] Tremblay, J., et al. (2018). Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization. CVPR Workshops. Evidence that deliberately unrealistic, widely-randomised synthetic data can outperform carefully photorealistic synthetic data on the real target. https://arxiv.org/abs/1804.06516

[3] Ronneberger, O., Fischer, P., Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI. The encoder-decoder shape that consumes the synthetic image and emits the per-pixel mask the generator manufactures. https://arxiv.org/abs/1505.04597

[4] Salehi, S. S. M., Erdogmus, D., Gholipour, A. (2017). Tversky Loss Function for Image Segmentation Using 3D Fully Convolutional Deep Networks. MLMI. The loss that handles the sparse-foreground imbalance the two-curve synthetic logs create. https://link.springer.com/chapter/10.1007/978-3-319-67389-9_44

EARTHSCAN

Designing a Procedural Log Generator: Geometry, Annotations and Domain Randomisation

Published September 2022. Generated 2026-07-22.
https://www.earthscan.io/whitepapers/procedural-log-generator-domain-randomisation