Case Study: Real-Time Clinical Documentation with Diffusion Language Models

Section 1: Industry Context and Business Problem

Industry: Healthcare -- Clinical Documentation

The United States healthcare system generates an estimated 1.2 billion clinical encounters per year. Each encounter requires a physician to produce a clinical note -- a structured medical document that captures the patient's history, examination findings, assessment, and treatment plan. These notes are not optional. They are legally required, drive billing and insurance reimbursement, and serve as the primary communication medium between providers.

The documentation burden is one of the most pressing problems in modern medicine. Studies published in the Annals of Internal Medicine consistently find that physicians spend **two hours on documentation for every one hour of direct patient care**. This translates to roughly 15.5 hours per week spent typing into electronic health records (EHRs), contributing directly to the physician burnout crisis that now affects over 60% of practicing clinicians. The American Medical Association estimates that documentation inefficiency costs the US healthcare system over USD 150 billion annually in lost productivity.

Company Profile: MedScribe AI

MedScribe AI is a health-tech startup headquartered in Boston, Massachusetts, founded in 2022 by a team of three: a former physician informaticist from Massachusetts General Hospital, a machine learning researcher from MIT CSAIL, and a product executive from a major EHR vendor.

- **Team size:** 85 employees (32 engineering, 18 ML/AI research, 12 clinical operations, 23 business/ops)
- **Funding:** Series B, USD 42M raised (USD 6M seed, USD 36M Series B led by a16z Bio + Fund)
- **Product:** An AI-powered clinical documentation assistant that listens to patient-physician conversations and generates structured SOAP notes in real time
- **Customers:** 14 health systems and 340 individual clinics, primarily in primary care and internal medicine
- **Monthly active physicians:** 2,800
- **Notes generated per month:** 1.4 million

MedScribe's current product uses a fine-tuned autoregressive language model (based on an open-source 7B parameter architecture) to generate clinical notes from transcribed audio. The model produces high-quality notes that physicians rate as "acceptable without edits" 68% of the time -- a strong result that has driven rapid adoption.

Business Challenge

Despite the product's success, MedScribe faces three critical technical limitations that are blocking growth into its next phase:

1. Latency kills the real-time experience.

MedScribe's current autoregressive model generates notes section-by-section. Each section (History of Present Illness, Review of Systems, Assessment, Plan) takes 3 to 5 seconds to generate. For a complete SOAP note, total generation time is 12 to 20 seconds. While this seems fast, it fundamentally breaks the real-time workflow. Physicians want to glance at the note *during* the encounter to verify accuracy -- not wait 20 seconds after the conversation ends. Competitor products using GPT-4 APIs achieve similar quality but with 8 to 15 second latency, so the entire market is constrained by autoregressive generation speed.

2. Bidirectional edits are impossible.

When a physician reviews a generated note and edits one section -- say, changing the Assessment from "Type 2 Diabetes, well-controlled" to "Type 2 Diabetes, poorly controlled" -- the downstream Plan section should update automatically (e.g., adding medication adjustment). But the autoregressive model can only generate left to right. It cannot propagate a mid-document change to sections that come after *or before* the edit. Currently, MedScribe regenerates the entire note from scratch, which takes another 12 to 20 seconds and often produces a note inconsistent with the physician's edit intent.

3. Infilling and section completion are not natively supported.

Physicians frequently want to fill in a partially completed template -- for example, a pre-populated note with the patient's medication list and vitals already filled in, but with the Assessment and Plan left blank. The autoregressive model cannot naturally handle this "infilling" task. MedScribe has built a workaround using prompt engineering, but it is brittle and produces lower-quality output (52% acceptance rate vs. 68% for full generation).

Why It Matters

- **Revenue impact:** MedScribe's sales pipeline includes three large health systems (USD 4.2M combined ARR) that have explicitly cited latency and editing limitations as blockers to contract signing. The VP of Sales estimates USD 8M in at-risk annual revenue if these issues are not resolved within 6 months.
- **Physician satisfaction:** Net Promoter Score dropped from 62 to 54 over the past quarter, with "too slow" and "can't edit without regenerating" as the top two complaints.

- **Competitive pressure:** Two well-funded competitors (one backed by Google Health, one by Microsoft/Nuance) are rumored to be evaluating diffusion-based generation architectures. First-mover advantage in sub-second clinical note generation would be a significant moat.
- **Patient safety:** Faster, more accurate documentation reduces the risk of medical errors caused by incomplete or rushed notes. In a 2024 JAMA study, documentation errors contributed to adverse events in 3.7% of hospital admissions.

Constraints

Constraint	Requirement
Latency	Complete SOAP note generation in < 500ms (from transcription input to structured output)
Model size	Must run on a single NVIDIA A100 (80GB) or equivalent; edge deployment on NVIDIA Jetson Orin targeted for Phase 2
Privacy / Compliance	HIPAA compliant; no patient data may leave the customer's VPC; model must run on-premises or in a dedicated cloud tenant
Data	Training on de-identified clinical text only (MIMIC-III/IV discharge summaries as proxy; real customer data available under BAA for fine-tuning)
Accuracy	Physician acceptance rate (no-edit rate) must be >= 65%, matching or exceeding the current autoregressive baseline
Training budget	8x A100 GPUs for up to 2 weeks; inference must be cost-effective at USD 0.002 per note or less
Team expertise	ML team has deep Transformer experience but limited diffusion model experience; solution must be implementable in 3 months

---

Section 2: Technical Problem Formulation

Problem Type: Conditional Text Generation with Bidirectional Context

The core task is **conditional text generation**: given a clinical context (transcribed patient-physician conversation, patient demographics, medication list), generate a structured SOAP note. This is fundamentally a sequence-to-sequence generation problem.

Why not framing alternatives?

- **Extractive summarization** (selecting spans from the transcript) fails because SOAP notes require synthesis and restructuring -- the note's structure does not mirror the conversation's chronological flow.
- **Template filling** (classification into pre-defined templates) fails because the space of possible clinical narratives is too large and open-ended to enumerate.

- **Autoregressive generation** (the current approach) works well for quality but is fundamentally limited in speed and bidirectionality, as described above.

The right framing is **masked diffusion generation**: start with a structured template of [MASK] tokens and iteratively unmask them using a bidirectional Transformer, guided by the clinical context. This directly addresses all three business challenges: speed (parallel unmasking), bidirectional editing (no left-to-right constraint), and infilling (naturally supported by partial masking).

Input Specification

The model receives three inputs:

1. **Clinical context** c : A tokenized representation of the transcribed conversation and patient metadata. Dimensions: (B, L_c) where $L_c \leq 1024$ tokens. This captures the raw clinical information from which the note is derived.

2. **Partial note** x_t : A sequence of tokens representing the current state of the SOAP note, with some positions containing real tokens and others containing [MASK]. Dimensions: (B, L_n) where $L_n \leq 512$ tokens. During generation from scratch, this is all [MASK] tokens. During infilling or editing, some positions are pre-filled.

3. **Masking ratio** t : A scalar in $[0, 1]$ indicating the current noise level. This tells the model what fraction of the note is currently masked, which is critical for calibrating its predictions.

Output Specification

The model outputs a probability distribution over the vocabulary at every position of the note:

$$p_\theta(x_0 \mid x_t, c) \in \mathbb{R}^{B \times L_n \times V}$$

where V is the vocabulary size. At each masked position, this distribution represents the model's belief about which token belongs there, given all visible tokens and the clinical context.

Why this output representation? Outputting full distributions (rather than single tokens) enables confidence-based unmasking -- the core mechanism that makes diffusion generation work. The model can express uncertainty about ambiguous positions and defer them to later steps.

Mathematical Foundation

The masked diffusion framework rests on three mathematical pillars:

1. The Forward Process as Independent Bernoulli Masking

For a clean note $x_0 = (x_0^1, x_0^2, \ldots, x_0^{L_n})$, the forward process at time t independently masks each token:

$$q(x_t^i \mid x_0^i) = (1 - t) \cdot \mathbb{1}[x_t^i = x_0^i] + t \cdot \mathbb{1}[x_t^i = \texttt{[MASK]}]$$

This is a Bernoulli process with parameter t. The expected number of masked tokens is $t \cdot L_n$. At $t = 0$, the note is clean. At $t = 1$, the note is fully masked.

Why independent masking? Independence across positions is what enables the tractable training objective. If masking were correlated (e.g., always masking contiguous spans), the conditional distributions would become intractable. Independent masking also means the model sees every possible masking pattern during training, which is exactly what enables bidirectional reasoning and arbitrary infilling at test time.

2. The Evidence Lower Bound (ELBO) and Its Simplification

The standard variational bound on the log-likelihood for discrete diffusion models involves a sum of KL divergence terms across timesteps. For masked diffusion specifically, Sahoo et al. (2024) showed that this bound simplifies dramatically. The key insight is that in the absorbing-state formulation (where [MASK] is an absorbing state), the reverse transition probabilities have a closed form:

$$q(x^i_{t-\Delta t} \mid x^i_t, x^i_0) = \begin{cases} 1 \\ \frac{\Delta t}{t} \cdot \mathbb{1}[x^i_{t-\Delta t} = x^i_0] + (1 - \frac{\Delta t}{t}) \cdot \mathbb{1}[x^i_{t-\Delta t} = \texttt{[MASK]} \end{cases}$$

This says: if a token is already unmasked, it stays unmasked. If it is masked, it gets unmasked with probability $\Delta t/t$. The ELBO then simplifies to a weighted cross-entropy loss at masked positions.

3. The Continuous-Time Limit

Taking $\Delta t \to 0$ yields the continuous-time training objective:

$$\mathcal{L} = -\mathbb{E}_{t \sim U(0,1)} \left[\frac{1}{t \cdot L_n} \sum_{i:\, x^i_t = \texttt{[MASK]}} \log p_\theta(x^i_0 \mid x_t, c) \right]$$

The $1/t$ weighting is not arbitrary -- it emerges from the ELBO derivation. It upweights low masking ratios (small t), where fewer tokens are masked and each prediction carries more information. Intuitively, predicting a single masked word in an otherwise complete sentence is harder and more informative than predicting one of many masked words in a mostly-masked sentence.

Numerical example: Consider a 4-token note segment $x_0 =$ ["Patient", "reports", "chest", "pain"] with $t = 0.5$. Two tokens are masked: $x_t =$ ["Patient", [MASK], [MASK], "pain"]. If the model predicts $p_\theta(\text{"reports"} \mid x_t, c) = 0.85$ and $p_\theta(\text{"chest"} \mid x_t, c) = 0.70$, the loss is:

$$\mathcal{L} = -\frac{1}{0.5 \times 4} \left[\log(0.85) + \log(0.70) \right] = -\frac{1}{2} \left[-0.163 + (-0.357) \right] = \frac{0.520}{2} =$$

If the model were perfectly confident ($p = 1.0$ for both), the loss would be 0. The model is penalized more for the lower-confidence prediction on "chest."

Loss Function

The full training loss for the clinical documentation model combines two terms:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{diffusion}} + \lambda \cdot \mathcal{L}_{\text{section}}$$

Term 1: Diffusion Loss $\mathcal{L}_{\text{diffusion}}$

$$\mathcal{L}_{\text{diffusion}} = -\mathbb{E}_{t \sim U(0,1)} \left[\frac{1}{t \cdot L_n} \sum_{i:\, x_t^i = \texttt{[MASK]}} \log p_\theta(x_0^i \mid x_t, c) \right]$$

This is the core masked prediction objective. It optimizes the model to accurately predict original tokens at masked positions, given the clinical context and all visible tokens.

- **If removed:** The model cannot generate text at all -- this is the primary training signal.
- **Effect of the $1/t$ weighting:** Without it (i.e., uniform weighting), the model would over-optimize for high masking ratios where predictions are easy (many masked tokens provide little constraint). The $1/t$ factor ensures balanced learning across all masking ratios, which is critical for generation quality.

Term 2: Section Structure Loss $\mathcal{L}_{\text{section}}$

$$\mathcal{L}_{\text{section}} = -\frac{1}{L_n} \sum_{i=1}^{L_n} \log p_\phi(s_i \mid x_t, c)$$

where $s_i \in \{\text{HPI}, \text{ROS}, \text{Exam}, \text{Assessment}, \text{Plan}, \text{Other}\}$ is the SOAP section label for position i, predicted by an auxiliary classification head p_ϕ.

This auxiliary loss encourages the model to maintain correct SOAP note structure. Clinical notes must follow a specific section ordering, and without explicit structural supervision, the diffusion model occasionally generates syntactically correct but structurally jumbled notes.

- **If removed:** Note quality drops by approximately 8 percentage points on the acceptance metric, primarily due to section boundary errors (e.g., plan items appearing in the HPI section).
- **Weighting coefficient** λ: Set to 0.1. Higher values (e.g., 1.0) cause the model to over-focus on section boundaries at the expense of content quality. Lower values (e.g., 0.01) provide insufficient structural guidance. The value 0.1 was determined via grid search on a validation set.

Evaluation Metrics

Metric	Description	Target
Physician acceptance rate (primary)	Fraction of generated notes rated as "acceptable without edits" by a blinded physician reviewer	>= 65%
ROUGE-L	Longest common subsequence overlap with reference notes	>= 0.45
Section F1		

Metric	Description	Target
	Macro-averaged F1 for correct section assignment of each sentence	>= 0.90
Clinical accuracy	Fraction of clinical facts (diagnoses, medications, vitals) correctly captured, measured via NER-based extraction	>= 0.85
Generation latency	Wall-clock time from input to complete SOAP note on a single A100	< 500ms
Infilling quality	Acceptance rate when 1-2 sections are pre-filled and the model completes the rest	>= 60%

Baseline: Autoregressive Generation

The current production baseline is a fine-tuned 7B autoregressive Transformer (LLaMA-2 architecture) with causal attention. It generates notes section-by-section, left to right.

Baseline performance: - Physician acceptance rate: 68% - ROUGE-L: 0.48 - Section F1: 0.93 - Clinical accuracy: 0.87 - Generation latency: 12-20 seconds (full SOAP note) - Infilling quality: 52% (via prompt engineering workaround)

The baseline achieves strong quality metrics but is fundamentally limited by sequential generation speed and inability to handle bidirectional edits or native infilling. The diffusion approach aims to match or exceed quality while delivering an order-of-magnitude improvement in latency and enabling new editing capabilities.

Why Diffusion LLMs Are the Right Technical Approach

The masked diffusion framework has three fundamental properties that directly address MedScribe's technical requirements:

1. **Parallel generation eliminates the sequential bottleneck.** An autoregressive model generating a 400-token SOAP note requires 400 forward passes. A diffusion model with $S = 20$ denoising steps requires only 20 forward passes, each processing all 400 tokens in parallel. On an A100 GPU, this translates from ~15 seconds to ~400ms -- well within the 500ms latency target.

2. **Bidirectional attention enables coherent edits.** Because the diffusion Transformer uses full (non-causal) attention, every position conditions on every other position. When a physician edits the Assessment, the model can propagate this change to both the Plan (downstream) and the HPI (upstream) simultaneously, because it has no directional constraint.

3. **Masking-based generation natively supports infilling.** To fill in missing sections of a partially completed note, simply keep the pre-filled sections unmasked and run the denoising process on the masked sections. The model naturally conditions on the visible tokens to

generate coherent completions. No prompt engineering or architectural modifications are needed.

Technical Constraints

Constraint	Value
Model parameters	1B - 3B (must fit on single A100 with room for KV cache)
Inference latency	< 500ms per note (400-500 tokens)
Denoising steps	15-25 (balancing quality vs. speed)
Training compute	8x A100 for <= 14 days
Training data	MIMIC-III discharge summaries (~50,000 notes) + synthetic augmentation
Vocabulary	Clinical tokenizer, 32,000 subword tokens
Sequence length	Context: 1024 tokens, Note: 512 tokens

Section 3: Implementation Notebook Structure

3.1 Data Acquisition Strategy

Dataset: MIMIC-III Clinical Database (Johnson et al., 2016) -- a freely available dataset of de-identified clinical records from Beth Israel Deaconess Medical Center. We use the discharge summary subset, which contains ~52,000 structured clinical notes.

For this case study notebook, we use a preprocessed subset of MIMIC-III discharge summaries. The data is loaded as plain text with section headers (HPI, Assessment, Plan, etc.) preserved.

Preprocessing pipeline: 1. Extract discharge summaries from the MIMIC-III NOTEEVENTS table 2. Parse section boundaries using regex patterns for standard SOAP headers 3. Tokenize with a clinical-aware tokenizer (we use a standard BPE tokenizer as a proxy) 4. Filter notes to those with 100-500 tokens and at least 3 identifiable sections 5. Split: 80% train, 10% validation, 10% test

TODO: Students implement the section parsing and filtering logic.

3.2 Exploratory Data Analysis

Key analyses to perform: - Distribution of note lengths (tokens per note) - Distribution of section lengths (tokens per section) - Vocabulary frequency analysis: most common clinical terms - Section ordering patterns: how consistent is SOAP structure across notes? - Missing section analysis: which sections are most frequently absent?

TODO: Students write EDA code, generate visualizations, and answer guided questions about the clinical text distribution.

3.3 Baseline Approach

Implement a simple autoregressive baseline using a small causal Transformer. This establishes the quality floor and the latency ceiling that the diffusion model must beat.

The baseline: - Architecture: 4-layer causal Transformer with 256-dim embeddings and 4 attention heads - Training: Standard next-token prediction on clinical notes - Generation: Greedy decoding, left to right - Evaluation: ROUGE-L and generation latency

TODO: Students implement the autoregressive baseline model, training loop, and evaluation.

3.4 Model Design

The core diffusion model is a bidirectional Transformer with time conditioning and optional section embeddings.

Architecture components: 1. **Token embedding** + **positional encoding**: Standard learned embeddings with sinusoidal position encoding 2. **Time conditioning MLP**: Converts the scalar masking ratio t into a d-dimensional vector added to each position 3. **Section embedding** (optional): Learned embeddings indicating which SOAP section each position belongs to 4. **Bidirectional Transformer encoder**: Standard Transformer encoder layers with full (non-causal) attention 5. **Output head**: Linear projection from hidden dimension to vocabulary size

Key design decisions: - No causal mask: the model sees all positions bidirectionally, which is what enables infilling and bidirectional editing - Time conditioning via addition (not concatenation): keeps the sequence length constant and adds minimal compute - Section embeddings provide structural guidance but are optional -- the model should learn structure from data alone as a fallback

TODO: Students implement the ClinicalDiffusionLM model class, including the forward pass, time conditioning, and masking logic.

3.5 Training Strategy

- **Optimizer:** AdamW with weight decay 0.01. AdamW is preferred over SGD because Transformer training benefits from adaptive learning rates, and weight decay prevents the embeddings from growing unboundedly.
- **Learning rate:** 3e-4 with cosine annealing to 1e-5. The cosine schedule provides a smooth decay that avoids the sudden drops of step-based schedules.
- **Batch size:** 64 (effective, with gradient accumulation if needed)
- **Masking ratio sampling:** $t \sim U(0.02, 1.0)$. We avoid t very close to 0 because masking 0 tokens provides no training signal.
- **Gradient clipping:** Max norm 1.0 to prevent training instabilities common in Transformer training.
- **Epochs:** 10-20 (until validation loss plateaus)

TODO: Students implement the training loop with proper logging and validation.

3.6 Evaluation

Quantitative evaluation on the held-out test set: 1. Compute ROUGE-L between generated notes and reference notes 2. Measure section F1 by extracting section boundaries from generated notes 3. Measure generation latency (wall-clock time per note) 4. Compare all metrics against the autoregressive baseline

For generation, implement confidence-based iterative unmasking with a configurable number of denoising steps.

TODO: Students implement the generation function and evaluation pipeline, producing a comparison table and visualizations.

3.7 Error Analysis

Systematic analysis of failure modes: 1. **Section confusion errors**: Content placed in the wrong SOAP section 2. **Hallucination errors**: Clinical facts not present in the input context 3. **Repetition errors**: Repeated phrases or sentences within a section 4. **Truncation errors**: Incomplete sentences at section boundaries

For each error type, students should: - Count occurrences in a sample of 50 generated notes - Identify common patterns - Propose a mitigation strategy

TODO: Students implement an error categorization function and analyze 50 generated notes.

3.8 Scalability and Deployment Considerations

Production deployment analysis: - Profile inference latency across different denoising step counts (5, 10, 15, 20, 25) - Measure throughput (notes per second) on a single GPU - Estimate memory footprint for different model sizes - Discuss batch inference strategies for high-throughput scenarios

TODO: Students write an inference benchmarking script that measures latency and throughput at different step counts.

3.9 Ethical and Regulatory Analysis

Healthcare AI carries significant ethical and regulatory obligations: - **HIPAA compliance**: All training data must be de-identified; model must not memorize patient information - **Clinical bias**: Model may reflect biases in the training data (e.g., demographic disparities in documentation quality) - **Liability**: Who is responsible when an AI-generated note contains an error that leads to a patient harm? - **Transparency**: Can physicians understand why the model generated a particular note?

TODO: Students write a brief ethical impact assessment covering bias, privacy, liability, and transparency for MedScribe's diffusion-based documentation system.

Section 4: Production and System Design Extension

Architecture Overview

The production MedScribe system consists of four major components arranged in a pipeline:

```
[Audio Input] --> [Speech-to-Text Service] --> [Context Encoder]
                                                      |
                                                      v
                                            [Diffusion Note Generator]
                                                      |
                                                      v
                                            [Post-Processing & Validation]
                                                      |
                                                      v
                                               [EHR Integration Layer]
```

1. **Speech-to-Text Service**: Streaming ASR (e.g., Whisper-based) converts the patient-physician conversation to text in real time. Output: transcript segments with speaker diarization and timestamps.

2. **Context Encoder**: Encodes the transcript plus patient metadata (demographics, medication list, prior diagnoses from EHR) into a dense context representation. This is a separate encoder that runs once per encounter.

3. **Diffusion Note Generator**: The core model. Takes the context encoding and a masked note template, runs 15-25 denoising steps, and produces a structured SOAP note. Supports three modes: full generation, partial infilling, and edit propagation.

4. **Post-Processing and Validation**: Rule-based checks for structural validity (section ordering, required fields), clinical terminology normalization (mapping free text to ICD-10/SNOMED codes), and confidence flagging (marking low-confidence sections for physician review).

5. **EHR Integration Layer**: Formats the validated note into HL7 FHIR resources and pushes to the customer's EHR system via their API.

API Design

Primary endpoint: Generate Note

```
POST /api/v1/notes/generate

Request:
{
  "encounter_id": "enc_abc123",
  "transcript": "Patient presents with chest pain...",
  "patient_context": {
    "age": 62,
    "sex": "M",
    "medications": ["metoprolol 50mg", "aspirin 81mg"],
    "active_diagnoses": ["HTN", "Hyperlipidemia"]
  },
  "partial_note": {           // Optional: for infilling mode
    "assessment": "Chest pain, likely musculoskeletal",
```

```
    "plan": null              // null = generate this section
  },
  "config": {
    "denoising_steps": 20,
    "temperature": 0.8,
    "max_note_length": 512
  }
}

Response:
{
  "note_id": "note_xyz789",
  "sections": {
    "hpi": "62-year-old male presents with...",
    "ros": "Negative for fever, dyspnea...",
    "exam": "VS: BP 142/88, HR 78...",
    "assessment": "Chest pain, likely musculoskeletal",
    "plan": "1. NSAIDs PRN for pain..."
  },
  "confidence_scores": {
    "hpi": 0.89,
    "ros": 0.92,
    "exam": 0.76,
    "assessment": 0.94,
    "plan": 0.85
  },
  "generation_time_ms": 380,
  "denoising_steps_used": 20
}
```

Edit propagation endpoint:

```
POST /api/v1/notes/{note_id}/propagate-edit

Request:
{
  "edited_section": "assessment",
  "new_content": "Chest pain, concerning for unstable angina",
  "propagate_to": ["plan", "hpi"]
}
```

Serving Infrastructure

- **Model serving:** NVIDIA Triton Inference Server with TensorRT optimization. The bidirectional Transformer benefits from TensorRT's fusion of attention and MLP layers.
- **Scaling strategy:** Horizontal pod autoscaling on Kubernetes, with each pod containing one A100 GPU. Scale based on request queue depth with a target of < 100ms queue wait time.
- **Redundancy:** Minimum 3 replicas across 2 availability zones. Health checks every 10 seconds. Automatic failover with < 5 second recovery.
- **Caching:** LRU cache on the Context Encoder output keyed by encounter_id, since the same encounter may trigger multiple note generation requests (e.g., after edits).

Latency Budget

Component	Target (ms)	P50 (ms)	P99 (ms)
Request parsing and validation	5	2	8
Context encoding (single pass)	50	35	80
Diffusion generation (20 steps)	350	310	420

Component	Target (ms)	P50 (ms)	P99 (ms)
Post-processing and validation	30	20	50
Response serialization	5	2	8
Total	**440**	**369**	**566**

The P99 exceeds the 500ms target by 66ms. Mitigation: reduce to 15 denoising steps for P99 cases (quality impact: < 2% reduction in acceptance rate based on offline analysis).

Monitoring

Model performance metrics (tracked per hour): - Generation latency: P50, P95, P99 - Throughput: notes per second per GPU - Physician acceptance rate (lagging indicator, aggregated weekly) - Section F1 on shadow-evaluated notes - Confidence score distribution (shift indicates model degradation)

Infrastructure metrics: - GPU utilization per pod - Request queue depth - Error rate (5xx responses) - Memory utilization

Alerting thresholds: - P99 latency > 600ms for 5 consecutive minutes: page on-call engineer - Physician acceptance rate drops > 5 percentage points week-over-week: alert ML team lead - Mean confidence score drops > 0.05 from rolling 7-day baseline: trigger model drift investigation - Error rate > 1% for 10 minutes: page on-call engineer

Model Drift Detection

Clinical language evolves: new medications enter the market, diagnostic criteria change, and documentation standards are updated by health systems. MedScribe must detect and adapt to these shifts.

Detection strategy: 1. **Input drift:** Monitor the distribution of input transcript embeddings using Maximum Mean Discrepancy (MMD) between the current week's inputs and the training distribution. Alert threshold: MMD > 2 standard deviations above baseline. 2. **Output drift:** Track the distribution of section confidence scores. A sustained decrease in confidence suggests the model is encountering unfamiliar patterns. 3. **Label drift:** Weekly sample of 200 notes reviewed by clinical annotators. If acceptance rate drops below 60%, trigger emergency retraining.

Adaptation strategy: Monthly fine-tuning on the most recent 30 days of physician-approved notes (after de-identification). This creates a feedback loop where physician edits continuously improve the model.

Model Versioning

- All models stored in a model registry (MLflow) with full metadata: training data hash, hyperparameters, evaluation metrics, training logs.

- Semantic versioning: MAJOR.MINOR.PATCH (e.g., v2.3.1). MAJOR = architecture change, MINOR = retraining with new data, PATCH = configuration change.
- Rollback: Any previous version can be deployed within 5 minutes via Kubernetes deployment rollback. Last 10 versions retained in warm storage.

A/B Testing

- **Framework:** Physician-level randomization (each physician is assigned to model A or model B for the duration of the test). This avoids within-encounter inconsistency.
- **Duration:** Minimum 2 weeks per test to account for weekly workflow patterns.
- **Primary metric:** Physician acceptance rate (no-edit rate).
- **Statistical significance:** Two-sided t-test with alpha = 0.05 and power = 0.80. With 1,400 physicians per arm and ~250 notes per physician per 2-week period, the test is powered to detect a 3 percentage point difference in acceptance rate.
- **Guardrail metrics:** Clinical accuracy must not drop by more than 2 percentage points. Latency P99 must remain under 600ms. If either guardrail is breached, the test is automatically stopped and the control model is restored.

CI/CD for ML

Training pipeline (triggered by new data or hyperparameter changes): 1. Data validation: schema checks, distribution tests, minimum sample size 2. Training: distributed training on 8x A100 with checkpointing every 1000 steps 3. Evaluation: automated evaluation on held-out test set against all defined metrics 4. Gate: all metrics must meet or exceed thresholds; regression on any metric blocks promotion 5. Shadow deployment: new model runs alongside production for 48 hours, processing the same inputs but not serving responses 6. Promotion: if shadow metrics are within tolerance, promote to A/B test or full deployment

Inference pipeline (triggered by model promotion): 1. Model export to TensorRT format 2. Latency regression test: 1000 synthetic inputs, P99 must be under 600ms 3. Canary deployment: 5% of traffic for 2 hours 4. Full rollout: gradual increase to 100% over 4 hours

Cost Analysis

Training costs (one-time per model version): - 8x A100 (80GB) instances on AWS (p4d.24xlarge): USD 32.77/hour - Training duration: ~10 days = 240 hours - Training cost per version: ~USD 7,865 - Monthly retraining: ~USD 7,865/month

Inference costs (ongoing): - 3x A100 instances for serving (minimum for redundancy): USD 98.31/hour = USD 71,780/month - At 1.4M notes/month, cost per note: USD 71,780 / 1,400,000 = USD 0.051 per note - Target was USD 0.002 per note -- significant gap. Mitigation strategies: - Model distillation to a 1B parameter model (estimated 3x cost reduction) - Batched inference (processing 8-16 notes per forward pass, estimated 4x throughput improvement) - Mixed-precision inference (FP8 on H100, estimated 2x speedup) - Combined: estimated cost of USD 0.004 per note, approaching target

Total monthly infrastructure cost estimate: ~USD 80,000 (training + inference)

At USD 8M in at-risk ARR, the infrastructure investment represents approximately 12% of the revenue it protects -- well within acceptable margins for a Series B company with 70%+ gross margin targets.