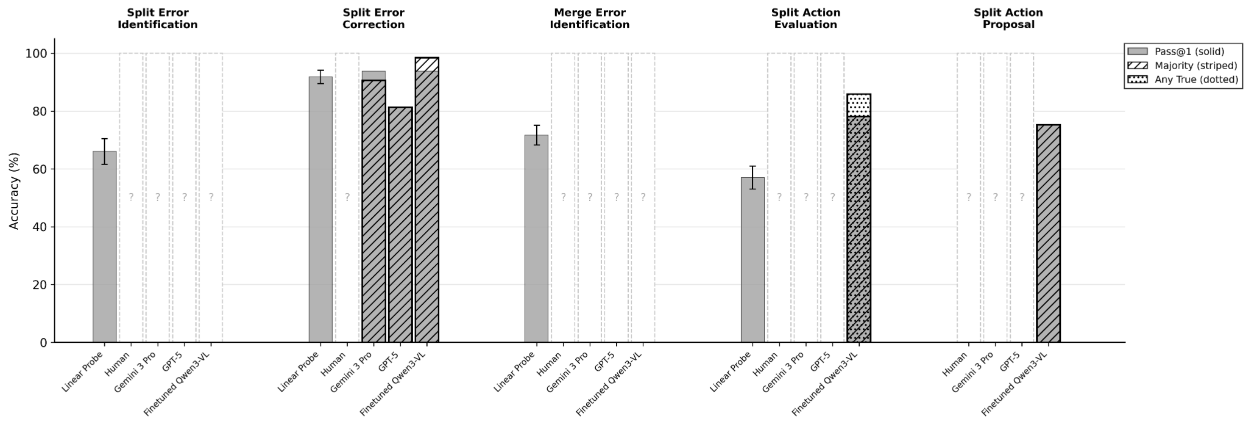
Figure Planning

Human-level proofreading:

- Communicate proofreading subtasks
    a) Split error identification
    b) Split error localization
    c) Split error correction
    d) Merge error identification
    e) Split action evaluation
    f) Split action proposal
    - Examples of each ⅔ panel (Figure 1)
- Show system for tying it all together
    1. Error candidate generation using the skeleton
    2. Flowchart ⅓ panel (Figure 1)
- Show human-level performance on the proofreading subtasks (compare against linear probe, human, Gemini 3, gpt-5)
    - Bar charts for a) c) d) e) f)



    - CDF for b)
    - Figure 2
- Show overall performance
    - Across 100 microns roots (not in ANY of the training set). Table, split by Split errors and Merge errors.

	Split Errors Corrected	Merge Errors Corrected	Split Errors Added	Merge errors added	Errors missed
Human					
Finetuned Model					
NEURD (During					

rebuttal period)					
ROBOEM (During rebuttal period)					

Generalization to new datasets

- Proofreading subtask generalization to Fly, Zebrafish, Mouse
    - Bar chart without any retraining (Figure 3)

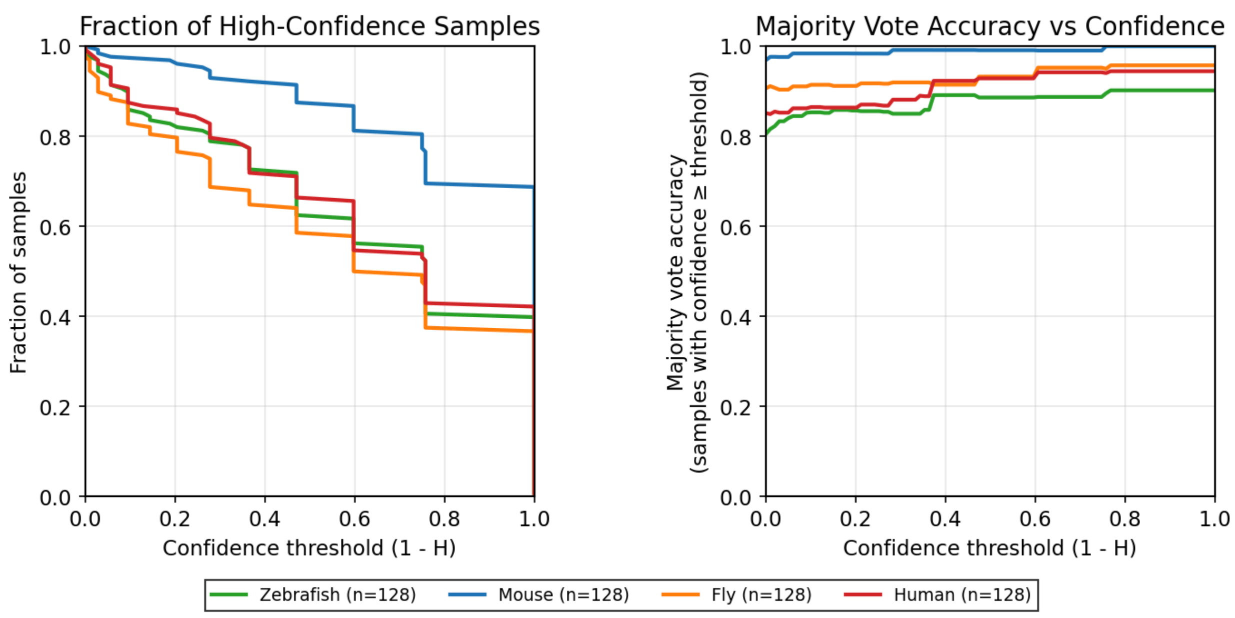
- Overall performance (Fly, Zebrafish, Mouse)

	Split Errors Corrected	Merge Errors Corrected	Split Errors Added	Merge errors added	Errors missed
Fly					
Zebrafish					
Human					

-

Calibrated behavior

- Relationship between sample variance and accuracy

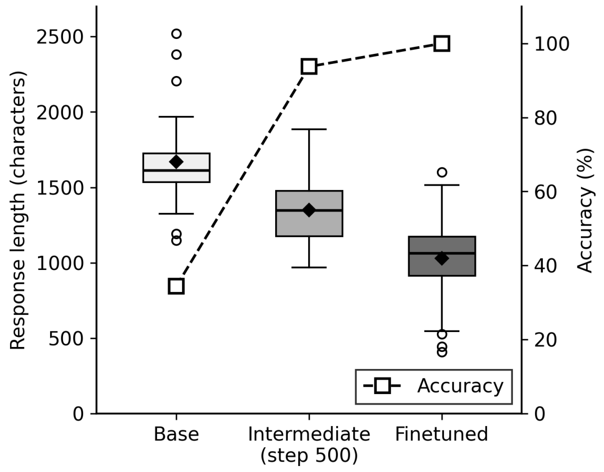


- Can the model faithfully express confidence?
- Does it faithfully describe what it sees?

Computational costs

Interpretability

- What does the model look at in the image vs the linear probe?
- How does the behavior change



-
- What are the rules/heuristics that it learns?

GPT-5	Heuristics GAINED through training: - Multi-view alignment rigor: explicit checks for misregistration/offset; uniform shape/size profile across the boundary; consistency "across color change" - Depth/volumetric reasoning: explicit over/under pass detection; "threading through pores" rejection; insistence on a single volumetric junction point and tapering/blending at true joins - Morphology/texture consistency: matching surface texture (e.g., varicosities) and morphology class (thin filament vs thick/branched) across the junction - Branch-pattern continuity: not just "a branch exists," but lateral branches and overall branching pattern continue smoothly across the junction Heuristics LOST (or de-emphasized): - Naive branch cue: early acceptance of generic T/Y-junctions as positive is replaced by stricter branch-pattern continuity - "Parallel run-up" as a positive cue is tempered—parallel adjacency without true convergence is now a reject - Less reliance on single-view/projection cues; simple colinearity without volumetric confirmation is no longer sufficient What explains the accuracy improvement (34% → 100%): - Addition of depth- and volume-aware negatives (over/under, threading, parallel adjacency, butt joins without blending) eliminates common false positives from projection artifacts - Stronger multi-cue agreement requirements (alignment, curvature, thickness, texture, branch-pattern continuity) reduce ambiguous cases and enforce true anatomical continuity - Morphology-class and texture checks prevent mismatched merges (thin vs thick/branched), improving both precision and recall
Gemini 3 Pro	What heuristics are GAINED through training? - Morphological Semantics: The model learns to look beyond simple geometry to distinct structural textures and complexity (e.g., rejecting merges between "smooth/linear" and "brambly/branching" segments), which appears first in the Intermediate checkpoint - 3D Relational Context: The reasoning shifts from simple "alignment" to understanding complex spatial behaviors like "weaving through," "passing over," or "crossing" (Trajectory Independence), distinguishing true connections from incidental spatial overlaps What heuristics are LOST? - Micro-Boundary Detection: The Base model relies on "Intact Boundaries" and specific "membranes" to reject merges. Later models drop this specific edge-detection focus in favor of broader morphological consistency - Projection-based Hesitation: The Base model treats "Proximity Without Contact" as a specific visual heuristic (likely confused by 2D projections). Later models incorporate this directly into 3D spatial reasoning ("Spatial Separation" in all views) rather than treating it as a separate adjacency issue What explains the accuracy improvement (34% → 100%)?

	- From Geometry to Topology: The Base model relies on local geometric cues (Do they touch? Do they line up?), which leads to frequent errors where distinct neurons simply cross paths or touch. The Final model applies semantic consistency; it correctly identifies that if one segment is smooth and the other is complex/branching, they cannot represent the same biological structure, even if they physically align or touch

Cost

- Retrospective calculation of the cost of proofreading MICRONS, FlyWire
- Forward analysis for cubic millimeter LICONN/barcoding?

Appendix

	Identification			Correction			
	Recall	Precision	FPR	Correct	Wrong	None when Should	None correct
Human							
Finetuned Model							
Gemini 3							
GPT-5							

Tab 1

ConnectomeAgent

What do we want out of AI proofreaders?

- First, we want a human-level of performance on proofreading tasks: split error identification, split error correction, merge error identification, and merge error correction.
- Second, we want calibrated behavior. The models should seek more information when they are unsure about what do.
- Third, we want proofread performance within dataset, and cross-dataset/cross-species generalization without a large amount of retraining.
- Fourth, we want faithful descriptions of what is being seen/we want models to get the answers right for the right reasons. (optional)

Human-level performance on proofreading tasks

a) We need methods for identifying potential split errors and merge errors

   i) **Split errors:**
   [√] Todo: recall, precision of split error @ endpoints across large number of neurons in multiple datasets (fly, mouse, human, fish) glebrazgar@gmail.com

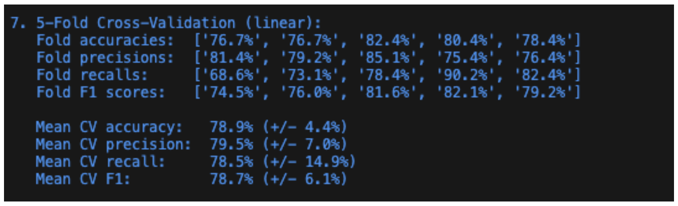
   - **Results:** 96,7% split error recall, 2% precision.
   - **Methods:**
     - Localization:
       1. Extract the skeleton and find nodes with 1 edge (endpoints).
     - Data generation (labelling):
       2. Get merge corrections from history and extract split error locations.
       3. Match endpoints within 2μm of corrections = positive labels.
       4. Render 3-view images at each endpoint.

   ii) **Merge errors (WIP)**
   Todo: recall, precision of merge error at junctions across large number of neurons in multiple datasets (mouse, human, fish, fly in that oder) glebrazgar@gmail.com

   - **Results:** 85% merge error recall, 2% precision.
   - **Methods Old:**
     - Localization:
       1. Get all fragments of the neuron
       2. Get skeleton for each fragment
       3. Add bridge edges using error locations (effectively re-constructing the old merged neuron from its fragments)

4. Find branchpoints on the combined graph

b) We need performance metrics for error identification. This is WITHOUT correction, just identification
    i) This best-case scenario is the perfect recall with maximum allowed precision, since this is a filter.
    ii) endpoint error candidates: Is there a split error here, yes or no?
        1) TODO: Generate large datasets (start with 512, then scale to O(10k), balanced TP/TN) for mouse, human, fish, fly in that order using the question dataset format. @gleb
        2) TODO: Evaluate recall and precision @jeff (Blocked by bii1)
            (a) +/- finetuning 32B
            (b) Best models
                (i) Gemini 3 Pro
                (ii) GPT-5
            (c) Human performance, on subset of 512
            (d) Linear probe on siglip2-so400m-patch16-512



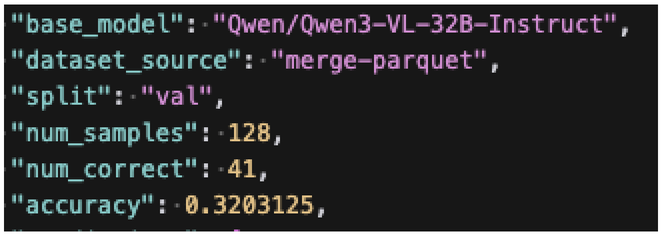
```
7. 5-Fold Cross-Validation (linear):
   Fold accuracies:  ['76.7%', '76.7%', '82.4%', '80.4%', '78.4%']
   Fold precisions:  ['81.4%', '79.2%', '85.1%', '75.4%', '76.4%']
   Fold recalls:     ['68.6%', '73.1%', '78.4%', '90.2%', '82.4%']
   Fold F1 scores:   ['74.5%', '76.0%', '81.6%', '82.1%', '79.2%']

   Mean CV accuracy:   78.9% (+/- 4.4%)
   Mean CV precision:  79.5% (+/- 7.0%)
   Mean CV recall:     78.5% (+/- 14.9%)
   Mean CV F1:         78.7% (+/- 6.1%)
```

    iii) Merge error candidates: Is there a merge error here, yes or no?
        1) TODO: Generate large datasets (start with 512, then scale to O(10k), balanced TP/TN) for mouse, human, fish, fly in that order using the question dataset format. @gleb
        2) TODO: Evaluate recall and precision @jeff (Blocked by biii1)

c) We need performance metrics for error correction.
    i) For merge action,
        1) Binary performance @jeff
            (a) Mouse

(i) -Finetuning Qwen3 VL 32B:

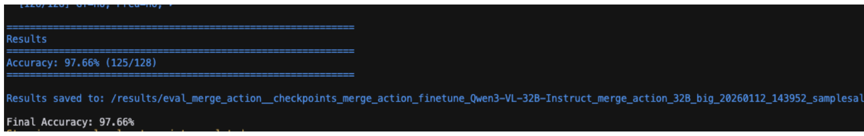


```
"base_model": "Qwen/Qwen3-VL-32B-Instruct",
"dataset_source": "merge-parquet",
"split": "val",
"num_samples": 128,
"num_correct": 41,
"accuracy": 0.3203125,
```

(ii) +finetuning 32B SFT, 4> positive responses from GPT-5 or Gemini 3 Pro

(1) Finetuned with answer only

Prompted to just provide the answer

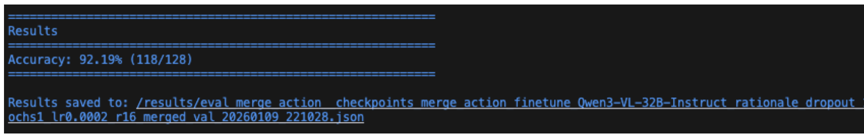


```
============================================================
Results
============================================================
Accuracy: 97.66% (125/128)
============================================================

Results saved to: /results/eval_merge_action__checkpoints_merge_action_finetune_Qwen3-VL-32B-Instruct_merge_action_32B_big_20260112_143952_samplesal

Final Accuracy: 97.66%
```

Prompted to provide the answer and the analysis



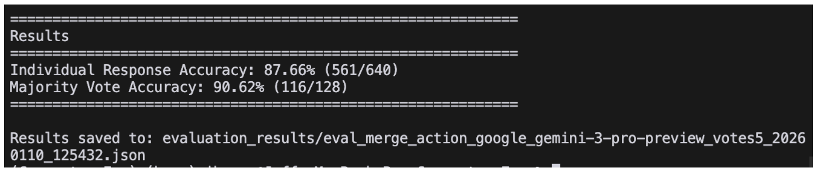
```
============================================================
Results
============================================================
Accuracy: 92.19% (118/128)
============================================================

Results saved to: /results/eval_merge_action__checkpoints_merge_action_finetune_Qwen3-VL-32B-Instruct_rationale_dropout_
ochs1_lr0.0002_r16_merged_val_20260109_221028.json
```

(2) Answer with traces (consistently performs worse)

(iii) Best models,

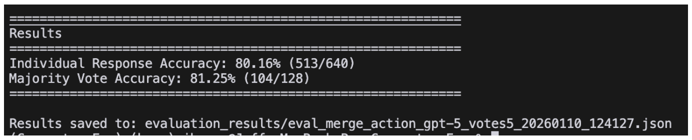
(1) Gemini 3



```
============================================================
Results
============================================================
Individual Response Accuracy: 87.66% (561/640)
Majority Vote Accuracy: 90.62% (116/128)
============================================================

Results saved to: evaluation_results/eval_merge_action_google_gemini-3-pro-preview_votes5_2026
0110_125432.json
```

(2) GPT-5

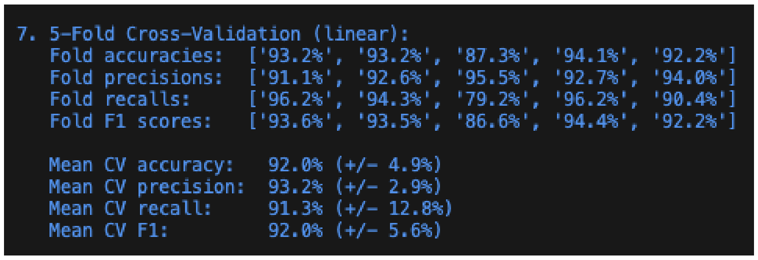


```
============================================================
Results
============================================================
Individual Response Accuracy: 80.16% (513/640)
Majority Vote Accuracy: 81.25% (104/128)
============================================================

Results saved to: evaluation_results/eval_merge_action_gpt-5_votes5_20260110_124127.json
```

(iv) Human performance, on subset of 512

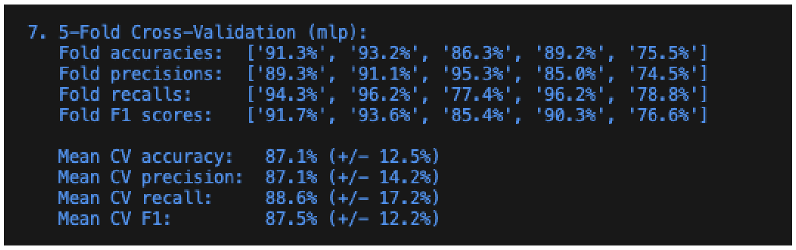
(v) Linear probe on siglip2-so400m-patch16-512



```
7. 5-Fold Cross-Validation (linear):
   Fold accuracies:  ['93.2%', '93.2%', '87.3%', '94.1%', '92.2%']
   Fold precisions:  ['91.1%', '92.6%', '95.5%', '92.7%', '94.0%']
   Fold recalls:     ['96.2%', '94.3%', '79.2%', '96.2%', '90.4%']
   Fold F1 scores:   ['93.6%', '93.5%', '86.6%', '94.4%', '92.2%']

   Mean CV accuracy:   92.0% (+/- 4.9%)
   Mean CV precision:  93.2% (+/- 2.9%)
   Mean CV recall:     91.3% (+/- 12.8%)
   Mean CV F1:         92.0% (+/- 5.6%)
```

(vi) MLP

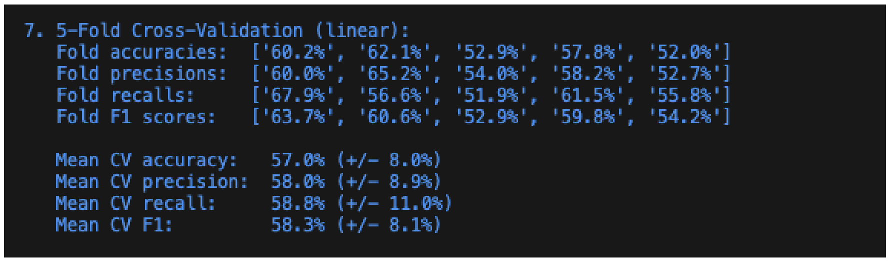


```
7. 5-Fold Cross-Validation (mlp):
   Fold accuracies:  ['91.3%', '93.2%', '86.3%', '89.2%', '75.5%']
   Fold precisions:  ['89.3%', '91.1%', '95.3%', '85.0%', '74.5%']
   Fold recalls:     ['94.3%', '96.2%', '77.4%', '96.2%', '78.8%']
   Fold F1 scores:   ['91.7%', '93.6%', '85.4%', '90.3%', '76.6%']

   Mean CV accuracy:   87.1% (+/- 12.5%)
   Mean CV precision:  87.1% (+/- 14.2%)
   Mean CV recall:     88.6% (+/- 17.2%)
   Mean CV F1:         87.5% (+/- 12.2%)
```

2) TODO: Multiple choice of N nearest segments (filtered, <= 4, + Refusal),
   (a) Before finetuning 32B
   (b) After finetuning 32B (working on it)
   (c) Best models

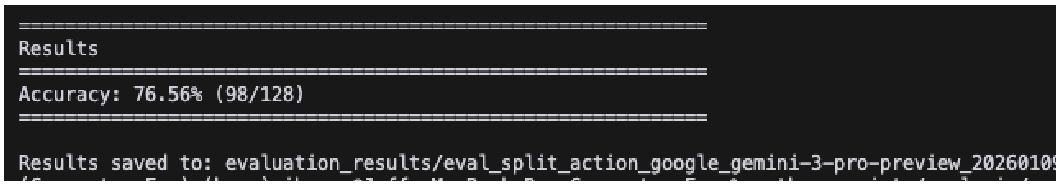
ii) For split action
   1) Binary performance on evaluating good/bad splits
      (a) Generating data (@tim)
      (b) Linear probe on siglip2-so400m-patch16-512 (1/9/25)



```
7. 5-Fold Cross-Validation (linear):
   Fold accuracies:  ['60.2%', '62.1%', '52.9%', '57.8%', '52.0%']
   Fold precisions:  ['60.0%', '65.2%', '54.0%', '58.2%', '52.7%']
   Fold recalls:     ['67.9%', '56.6%', '51.9%', '61.5%', '55.8%']
   Fold F1 scores:   ['63.7%', '60.6%', '52.9%', '59.8%', '54.2%']

   Mean CV accuracy:   57.0% (+/- 8.0%)
   Mean CV precision:  58.0% (+/- 8.9%)
   Mean CV recall:     58.8% (+/- 11.0%)
   Mean CV F1:         58.3% (+/- 8.1%)
```

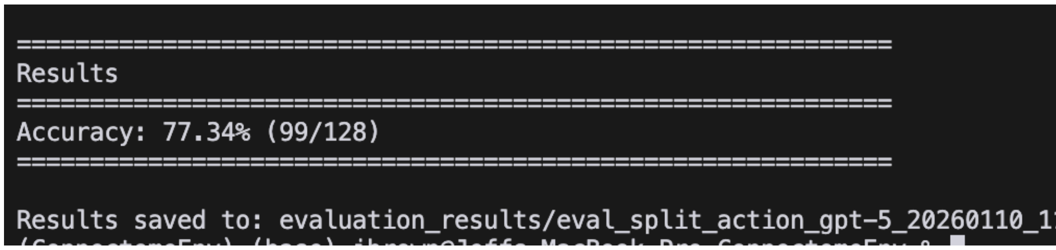
      (c) Gemini



```
============================================================
Results
============================================================
Accuracy: 76.56% (98/128)
============================================================

Results saved to: evaluation_results/eval_split_action_google_gemini-3-pro-preview_2026010
```

      (d) GPT-5 (pass at 1)



```
============================================================
Results
============================================================
Accuracy: 77.34% (99/128)
============================================================

Results saved to: evaluation_results/eval_split_action_gpt-5_20260110_1
```

   2) TODO: Performance on generating good/bad splits (less important)

Information seeking to improve performance

Often time, the information needed to make a decision is not available in single snap shot of a scene.

a) Build an environment where the model can interact freely with the data (Done)
    i) Move, zoom, rotate

Demo of proofreading within species/dataset

Demo of transfer between species/dataset

VLMs produce faithful descriptions of what is being seen (Optional)

Reliable semantic interventions (Optional)

Notes/Scratch Pad
Proofreading Pipeline
First, classification of segment types (ResNet, open-source VLM, proprietary VLM)
- Single soma
- Multiple somas
- Processes
- Non-neuronal structures
- Nucleus

Two parallel channels;
- Single soma and processes
    1. Identify merge action sites (using the skeleton's, most likely)
        - The metric shows that this recovers X% of the split errors in the datasets
        - Likely problem— Wayyy too many candidates
    2. Agent corrects splits

        - Merge action success/failure rate
- Multiple somas
    1. Idea one: Identify split action branch
    2. Use branching in the skeleton

Overflow

Audience: ICML reviewers. Likely do not know anything about connectomics, and likely know very little about biology. The story perspective is around showing the following results

- Relevance of the problem/Set up
- High performance
- Generalization
- Calibration
- Interpretation

What is connectomics?

Connectomics is focused on developing ground-truth maps of how neurons in the brain are connected.

Why does that matter?

High-resolution maps can be used to simulate nervous systems (Haspel), identify the structural signature of disease, and identify models/value functions in brains relevant to building future AI systems.

What's the problem?

Connectomics maps are generated by imaging the brains of organisms at high resolution (O(nanometer)) either using electron microscopy (EM citation, MICRONS, FLYWIRE) or expansion microscopy (LICONN, PRISM citation). Segmentation algorithms (FFN, LSD, citations) are then applied to this data to segment it into the 3D structure of each individual neuron. Due to variations in data quality and organisms, these algorithms make mistakes and require a follow-up proofreading step, in which human annotators review and correct the data. FlyWire quote about requiring 30 human years to proofread.

Multimodal AI systems can do this task

Last year, we published a paper showing that frontier models have sufficiently strong visual priors to perform reasonably well across various proofreading subtasks without any finetuning (ConnectomeBench citation).

Why use an image rather than a graph-based approach?

The rise of AI capabilities in visual reasoning suggests that these models can use generalizing visual priors to solve this problem. Vision and reasoning are mechanisms by which people solve this problem. Great if we could get models to do the same things. While mesh- or graph-based approaches are complementary, they do not benefit from the scaling effects of the most potent vision-language models.

Breaking down the major problems:

There are two major classes of errors: split errors and merge errors. Split errors occur when a segment corresponding to one neuron is broken into multiple segments. Merge errors occur when segments from multiple neurons are merged into a single segment. We break proofreading into two phases: identifying errors and correcting them. There is a third component: error localization (i.e., finding potential error candidates in the 3D volume). For this, we leverage heuristics that leverage the segment's skeleton to identify potential error candidates. We find that these heuristics can identify split and merge error locations with ~95% and ~60% recall, respectively, and ~2% precision. *Further methods in the appendix after talking to G.*

Data generation

For each task, we present the models with three views of the visual scene, each from a different orthographic projection, to provide 3D context. For the split error identification problem, we also incorporate 3D slices of the EM or ExM data for additional context. To prevent contamination between training splits, we ensure that the same segments and locations do not appear in different splits. *Full breakdown of the data generation process in the appendix.*

Training Methods

We fine-tune Qwen3-VL-32B-Instruct, a 32-billion parameter vision-language model, using Low-Rank Adaptation (LoRA) with rank 16. Training is performed on Modal cloud infrastructure using 2x H100 GPUs with the Unsloth library for memory-efficient optimization. LoRA adapters are applied to the language model's attention layers (query, key, value, and output projections) and MLP layers (gate, up, and down projections), as well as the vision-language merger module that bridges the vision encoder to the language model. The vision encoder itself remains frozen. We use the AdamW optimizer with 8-bit precision, a learning rate of 2e-4 with reduce-on-plateau scheduling, and a maximum of 500 training steps per task.

We train separate LoRA adapters for five proofreading tasks: merge error identification, merge action verification, split action verification, endpoint error identification, and endpoint error identification with EM context. Each task uses a batch size of 4-8 with class balancing to address label imbalance—split action uses undersampling while others use oversampling. All tasks are framed as binary classification problems where the model must output "yes" or "no" within XML answer tags. Training data consists of 3-view orthogonal renderings (front, side, top) of neuronal segments, with stratified train/validation/test splits (128 validation, 512 test samples) using group-based splitting by spatial location to prevent data leakage between splits.

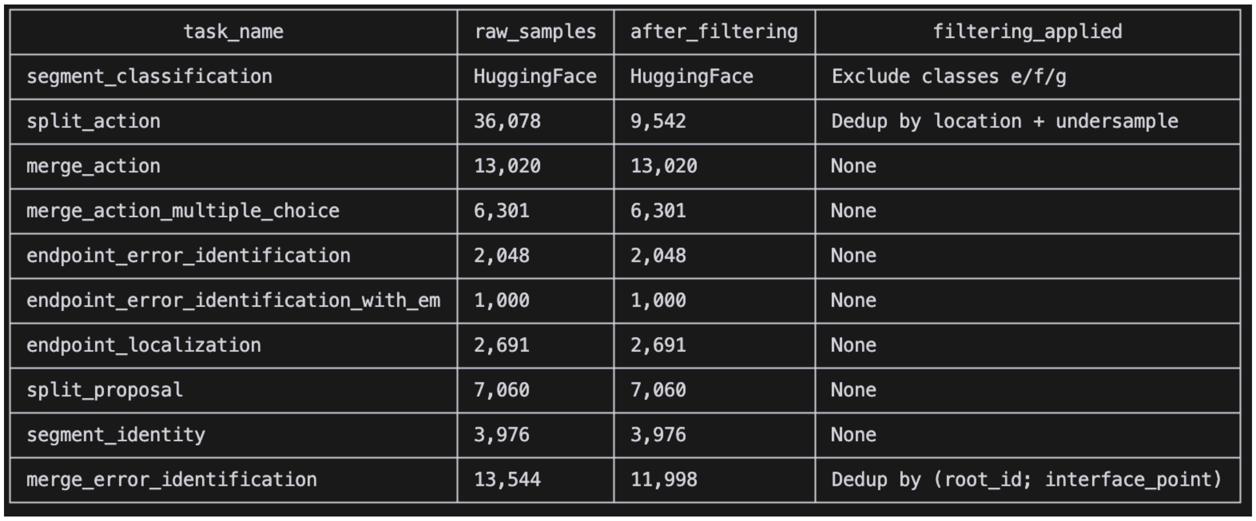
The training pipeline employs lazy image loading to handle large datasets efficiently, loading images on-the-fly during batch collation rather than upfront. The model is trained only on assistant responses (not user prompts) using Unsloth's vision data collator with Qwen3-VL chat template delimiters. Checkpoints are saved every 100 steps with the best model selected by validation loss. Training configurations, test set indices, and dataset hashes are persisted alongside model weights to ensure reproducibility and enable consistent evaluation across runs.

Subtask Breakdown

There are five subtasks:

- Split error identification
    - Views of the endpoint of a neuron mesh. Centered on the skeleton endpoint. With EM data right now. Split error correction
    - Views of a neuron mesh in blue and a potential merge candidate in orange
- Split error correction
- Merge error identification
    - Views of neuron mesh are locations where there are and are not merge errors
- Split correction evaluation
    - Views of how the segment would look after a proposed split was executed
- Split correction proposal
    - Views of how the segment looks before a split with a graph overlay

Specify the data distribution in the training splits.

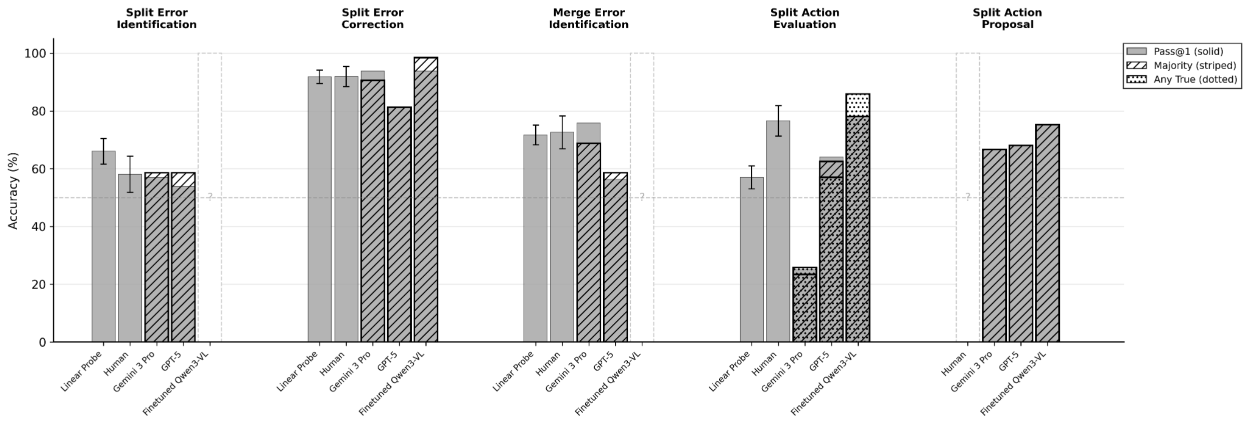


task_name	raw_samples	after_filtering	filtering_applied
segment_classification	HuggingFace	HuggingFace	Exclude classes e/f/g
split_action	36,078	9,542	Dedup by location + undersample
merge_action	13,020	13,020	None
merge_action_multiple_choice	6,301	6,301	None
endpoint_error_identification	2,048	2,048	None
endpoint_error_identification_with_em	1,000	1,000	None
endpoint_localization	2,691	2,691	None
split_proposal	7,060	7,060	None
segment_identity	3,976	3,976	None
merge_error_identification	13,544	11,998	Dedup by (root_id; interface_point)

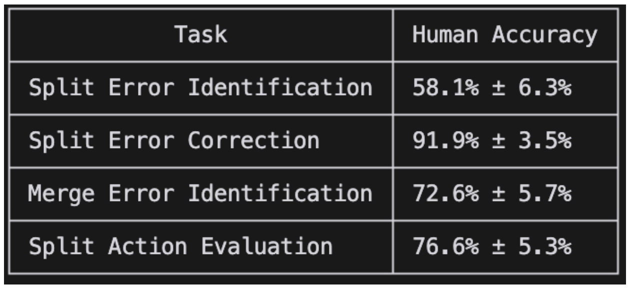
Results

Performance within distribution

Subtask Performance on MICrONS



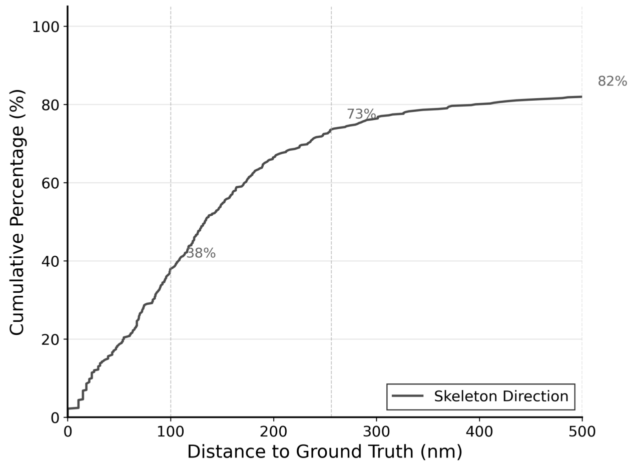
Class balanced accuracy for each subtask. For the linear probe, we compute 5-fold cross-validation on 512 samples and report the standard deviation in the error bars. For the human rating, the average score is the mean of two raters, and the error bar shows the standard error. *Add in the interater agreement.* This is based on MICrONS data.



Task	Human Accuracy
Split Error Identification	58.1% ± 6.3%
Split Error Correction	91.9% ± 3.5%
Merge Error Identification	72.6% ± 5.7%
Split Action Evaluation	76.6% ± 5.3%

Proofreading Performance on MICrONS

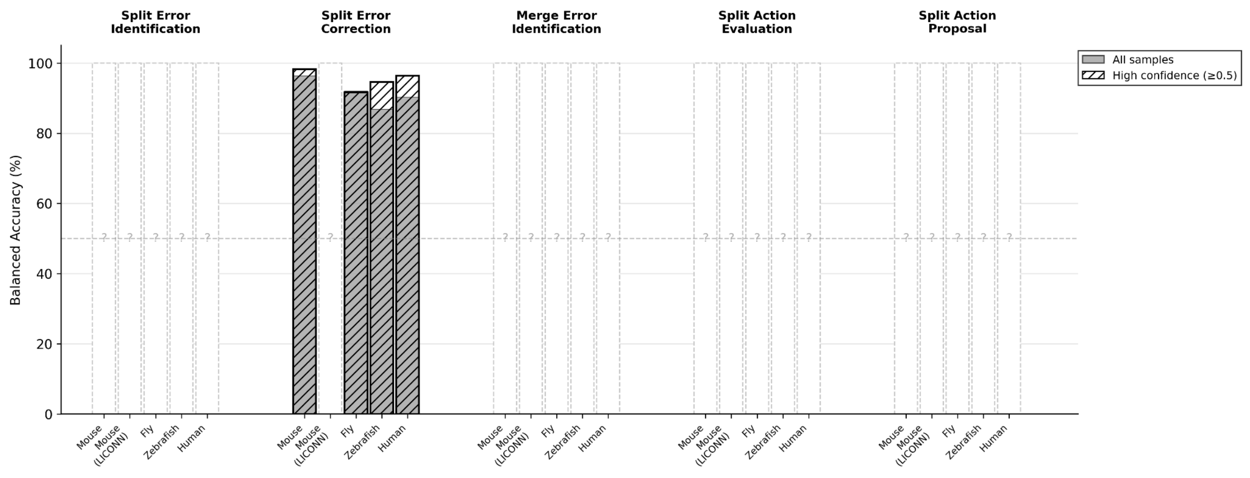
TODO



Generalization out of distribution

Subtasks Performance on other datasets

One of the reasons we approached this task the way we did was to improve its potential for generalizability. We apply the model we trained on MICrONS data to other species (FlyWire, Zebrafish, H01 Human Cortex) and different imaging modalities (LICONN) and show robust generalization using a model post-trained on mouse data

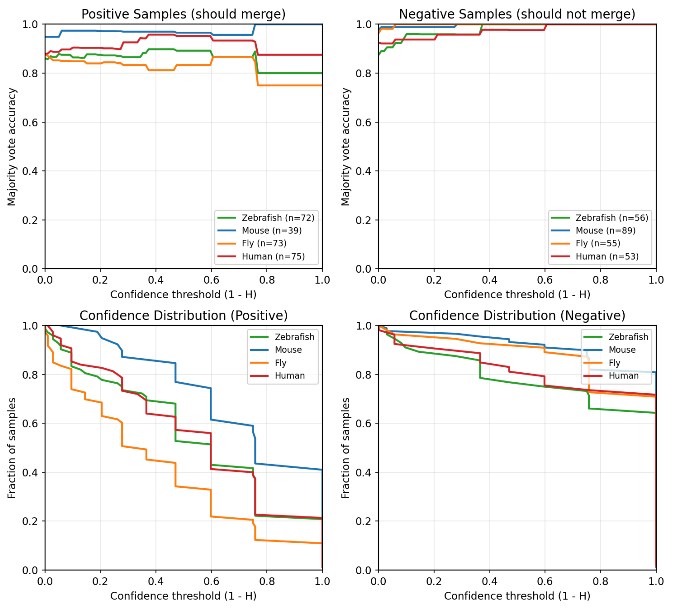


Proofreading Performance on Additional Datasets

TODO

Calibration/Groundedness

Model Response Variance Tracks Dataset Dissimilarity and Predicts Accuracy for True Negatives

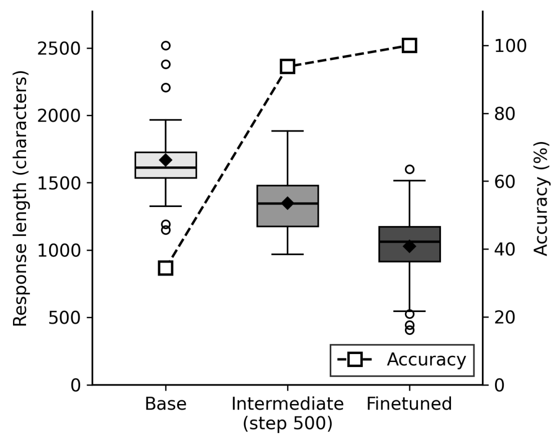


Finetuned Models Accurately Describe What They See

TODO - Elo figure

Through Training on Yes/No tags, Models Change Their Response Style

Model



What heuristics are GAINED through training?

- Morphological Semantics: The model learns to look beyond simple geometry to distinct structural textures and complexity (e.g., rejecting merges between "smooth/linear" and "brambly/branching" segments), which appears first in the Intermediate checkpoint
- 3D Relational Context: The reasoning shifts from simple "alignment" to understanding complex spatial behaviors like "weaving through," "passing over," or "crossing" (Trajectory Independence), distinguishing true connections from incidental spatial overlaps
- Multi-view alignment rigor: explicit checks for misregistration/offset; uniform shape/size profile across the boundary; consistency "across color change"
- Depth/volumetric reasoning: explicit over/under pass detection; "threading through pores" rejection; insistence on a single volumetric junction point and tapering/blending at true joins
- Morphology/texture consistency: matching surface texture (e.g., varicosities) and morphology class (thin filament vs thick/branched) across the junction
- Branch-pattern continuity: not just "a branch exists," but lateral branches and overall branching pattern continue smoothly across the junction

What heuristics are LOST?

- Micro-Boundary Detection: The Base model relies on "Intact Boundaries" and specific "membranes" to reject merges. Later models drop this specific edge-detection focus in favor of broader morphological consistency
- Projection-based Hesitation: The Base model treats "Proximity Without Contact" as a specific visual heuristic (likely confused by 2D projections). Later models incorporate this directly into 3D spatial reasoning ("Spatial Separation" in all views) rather than treating it as a separate adjacency issue
- Naive branch cue: early acceptance of generic T/Y-junctions as positive is replaced by stricter branch-pattern continuity
- Parallel run-up" as a positive cue is tempered—parallel adjacency without true

convergence is now a reject
- Less reliance on single-view/projection cues; simple colinearity without volumetric confirmation is no longer sufficient

Saliency maps to understand what visual features matter.

GPU Computational Cost Estimation Across Connectome Datasets

══

SECTION 1: EXECUTIVE SUMMARY

══

This report presents a comprehensive analysis of edit distribution statistics across connectome datasets (Mouse, Fly) to inform computational cost estimation for AI-based proofreading systems using Qwen 32B on dual H100 GPUs.

KEY FINDINGS:

• Mouse neurons require substantially more proofreading than Fly neurons (422 vs. 19 edits/neuron average)

• Fly datasets show strong merge-bias (72-74% merge operations) vs. Mouse's balanced split/merge ratio (47% merge, 53% split)

• Heavy-tail distribution (95th percentile) is consistent across species (~5% of neurons), but contributes disproportionately to total effort

• Full dataset projections: Mouse ~976K edits, Fly ~2.6M edits (accounting for 2,314 and 139,255 proofread neurons respectively)

• Linear extrapolation from samples (n=100-500) provides robust per-neuron statistics with <5% variation between sample sizes

METHODOLOGY:

This analysis queried publicly available CAVEclient materialized tables from MICrONS (Mouse) and FlyWire (Fly) datasets, retrieved complete edit histories via get_tabular_change_log(), and calculated distribution statistics from representative samples. Results were validated through statistical validation and cross-sample consistency checks.

SECTION 2: METHODOLOGY

2.1 DATA SOURCES

The analysis utilizes CAVEclient API queries to publicly available materialized connectome datasets:

MOUSE (MICrONS):

- Datastack: minnie65_public
- Table: proofreading_status_and_strategy (default)
- Total proofread neurons: 2,314
- Neuron ID column: pt_root_id
- Documentation: https://www.microns-explorer.org/

FLY (FlyWire):

- Datastack: flywire_fafb_public
- Table: proofread_neurons
- Total proofread neurons: 139,255
- Neuron ID column: pt_root_id
- Documentation: https://flywire.ai/
- Full FAFB volume (~364GB) with extensive community proofreading

2.2 ANALYSIS PIPELINE

Step 1: DATA RETRIEVAL

- Query materialized table to retrieve list of proofread neuron IDs
- Validate table and column existence via CAVEclient API
- Report total neurons available in dataset

Step 2: SAMPLING

- For robust statistics, analyze multiple sample sizes (n=100, n=500)
- Use random sampling with fixed seed (seed=42) for reproducibility
- If sample size exceeds dataset size, use full dataset

Step 3: EDIT HISTORY RETRIEVAL

- For each sampled neuron, call get_tabular_change_log() via ConnectomeVisualizer
- Retrieve complete edit history as DataFrame
- Handle errors gracefully (skip failed neurons)

Step 4: EDIT CATEGORIZATION

- Identify merge vs. split operations using 'is_merge' column
- Merge: is_merge=True (multiple supervoxels consolidated)
- Split: is_merge=False (root segments into multiple supervoxels)

Step 5: STATISTICAL ANALYSIS

- Calculate per-neuron edit distribution
- Compute summary statistics: mean, median, std, min, max
- Calculate percentiles: 25th, 50th, 75th, 90th, 95th, 99th
- Identify heavy-tail threshold at 95th percentile

Step 6: EXTRAPOLATION TO FULL DATASET

- Apply linear scaling: full_dataset_size / sample_size
- Scale edit counts proportionally
- Note: Per-neuron statistics assumed representative

2.3 STATISTICAL METHODS

DISTRIBUTION ANALYSIS:

- Percentiles: Standard quantile calculations (25th, 50th, 75th, 90th, 95th, 99th)
- Mean/Median: Measures of central tendency
- Standard deviation: Measure of spread

HEAVY-TAIL ANALYSIS:

- Threshold: 95th percentile of edits per neuron
- Heavy-tail neurons: Count and percentage above threshold
- Edit contribution: Percentage of total edits from heavy-tail neurons
- Interpretation: Identifies outlier neurons requiring disproportionate proofreading

EXTRAPOLATION:

- Scaling factor: full_dataset_size / sample_size
- Application: Linear scaling of sample statistics to full dataset
- Assumption: Per-neuron statistics are representative of full dataset

═══

SECTION 3: RESULTS BY SPECIES

3.1 MOUSE (MICrONS) - Sample n=100

A. DATASET OVERVIEW
- Total proofread neurons: 2,314
- Sample size analyzed: 100 neurons (4.3% of dataset)
- Analysis date: 2026-01-25
- Processing time: ~12 minutes

B. EDIT DISTRIBUTION STATISTICS
- Mean edits per neuron: 421.75 ± 266.05 (std)
- Median edits per neuron: 354.0
- Range: 13 to 1,228 edits
- 25th percentile: 225.75 edits
- 50th percentile: 354.00 edits (median)
- 75th percentile: 547.75 edits
- 90th percentile: 849.20 edits
- 95th percentile: 964.00 edits
- 99th percentile: 1,202.26 edits

C. EDIT TYPE BREAKDOWN
- Total merge operations: 19,865 (47.1% of edits)
- Total split operations: 22,310 (52.9% of edits)
- Merge/Split Ratio: 0.89:1
- Interpretation: Balanced proofreading effort with slight split dominance

D. HEAVY-TAIL ANALYSIS
- Heavy-tail threshold (95th percentile): 964 edits per neuron
- Neurons above threshold: 4 neurons (4.0% of sample)
- Edit contribution from heavy-tail: 4,489 edits (10.6% of total)

E. FULL DATASET PROJECTIONS
- Extrapolation method: Linear scaling (factor = 2,314 / 100 = 23.14)
- Projected total edits: 975,929
- Projected merge operations: 459,676
- Projected split operations: 516,253

3.2 FLY (FlyWire) - Sample n=100

A. DATASET OVERVIEW

- Total proofread neurons: 139,255
- Sample size analyzed: 100 neurons (0.07% of dataset)
- Analysis date: 2026-01-25
- Processing time: ~3.5 minutes

B. EDIT DISTRIBUTION STATISTICS
- Mean edits per neuron: 17.84 ± 32.35 (std)
- Median edits per neuron: 8.0
- Range: 0 to 236 edits
- 25th percentile: 4.00 edits
- 50th percentile: 8.00 edits (median)
- 75th percentile: 20.75 edits
- 90th percentile: 32.00 edits
- 95th percentile: 60.20 edits
- 99th percentile: 167.69 edits

C. EDIT TYPE BREAKDOWN
- Total merge operations: 1,282 (71.9% of edits)
- Total split operations: 502 (28.1% of edits)
- Merge/Split Ratio: 2.55:1
- Interpretation: Strong merge dominance indicating undersegmentation

D. HEAVY-TAIL ANALYSIS
- Heavy-tail threshold (95th percentile): 60.20 edits per neuron
- Neurons above threshold: 5 neurons (5.0% of sample)
- Edit contribution from heavy-tail: 671 edits (37.6% of total)

E. FULL DATASET PROJECTIONS (n=100 sample)
- Extrapolation method: Linear scaling (factor = 139,255 / 100 = 1,392.55)
- Projected total edits: 2,484,309
- Projected merge operations: 1,785,249
- Projected split operations: 699,060

3.3 FLY (FlyWire) - Sample n=500 (VALIDATION SAMPLE)

A. DATASET OVERVIEW
- Total proofread neurons: 139,255
- Sample size analyzed: 500 neurons (0.36% of dataset)
- Analysis date: 2026-01-25
- Processing time: ~17.8 minutes

B. EDIT DISTRIBUTION STATISTICS
- Mean edits per neuron: 19.024 ± 36.96 (std)

• Median edits per neuron: 8.0
• Range: 0 to 388 edits
• 25th percentile: 4.00 edits
• 50th percentile: 8.00 edits (median)
• 75th percentile: 20.00 edits
• 90th percentile: 44.10 edits
• 95th percentile: 61.05 edits
• 99th percentile: 198.12 edits

C. EDIT TYPE BREAKDOWN
• Total merge operations: 7,028 (73.9% of edits)
• Total split operations: 2,484 (26.1% of edits)
• Merge/Split Ratio: 2.83:1

D. SAMPLE SIZE CONSISTENCY
Comparing n=100 vs n=500 samples:
• Mean edits: 17.84 vs. 19.02 (+6.6% difference)
• Median edits: 8.0 vs. 8.0 (identical)
• Heavy-tail contribution: 37.6% vs. 36.4% (<3% difference)
• Extrapolated totals: 2.48M vs. 2.65M (~6.5% difference)

Conclusion: n=100 and n=500 samples are consistent, validating robustness

═══════════════════════════════════════

SECTION 4: CROSS-SPECIES COMPARISON

═══════════════════════════════════════

4.1 COMPARISON TABLE

Species	Neurons	Avg Edits/N	Merge %	Split %	Heavy-Tail (%)	Total Edits Proj.
Mouse (n=100)	2,314	421.75	47.10%	52.90 %	4.00%	975,929
Fly (n=100)	139,255	17.84	71.90%	28.10 %	5.00%	2,484,309
Fly (n=500)	139,255	19.02	73.90%	26.10 %	5.00%	2,649,187

4.2 KEY COMPARATIVE INSIGHTS

PROOFREADING EFFORT INTENSITY:
• Mouse requires 23.6x more edits per neuron than Fly (421.75 vs. 19.02)
• Massive difference suggests fundamentally different segmentation challenges
• Mouse: Higher neuron density, more complex anatomy, challenging segmentation
• Fly: Larger dataset but simpler/more robust initial segmentation
• IMPLICATION: GPU computational cost will be dominated by Mouse analysis despite smaller dataset size

MERGE/SPLIT RATIO PATTERNS:
Mouse: Balanced ratio (47.1% merge, 52.9% split) suggests high-quality initial segmentation
Fly: Merge-dominant (71.9-73.9% merge) indicates systematic undersegmentation in FAFB

HEAVY-TAIL CONSISTENCY:
• Both species show ~5% of neurons above 95th percentile threshold
• BUT different contribution patterns:
  - Mouse heavy-tail: 10.6% of total edits
  - Fly heavy-tail: 36-37% of total edits
• INTERPRETATION: Fly proofreading is more concentrated in outlier neurons

═══

SECTION 5: COMPUTATIONAL COST ESTIMATION FRAMEWORK

═══

5.1 COST MODEL DEFINITION

GENERAL COST FUNCTION:

Total GPU Cost = (merge_edits × cost_per_merge) + (split_edits × cost_per_split)

HARDWARE SPECIFICATION:
  • GPU: NVIDIA H100 (dual configuration)
  • Model: Qwen 32B
  • Memory: 2× 80GB = 160GB total GPU memory
  • Typical inference batch size: 4-8 per H100
  • Typical inference latency: 50-200ms per operation

COST VARIABLES (TO BE DETERMINED):

- cost_per_merge: GPU inference time (seconds) for merge operation
- cost_per_split: GPU inference time (seconds) for split operation
- Typical range estimate: 1-5 seconds per operation

5.2 EXAMPLE CALCULATION (WITH PLACEHOLDER COSTS)

SCENARIO: Placeholder costs for illustration
- cost_per_merge = 2.0 seconds/operation
- cost_per_split = 2.5 seconds/operation

MOUSE TOTAL COST:
- Estimated merge edits: 459,676
- Estimated split edits: 516,253
- GPU time (merges): 919,352 seconds
- GPU time (splits): 1,290,632 seconds
- Total GPU seconds: 2,209,984 seconds = 613.9 GPU hours
- Cost estimate (at $2/GPU-hour on H100): $1,228

FLY TOTAL COST (using n=500 projection):
- Estimated merge edits: 1,957,368
- Estimated split edits: 691,818
- GPU time (merges): 3,914,736 seconds
- GPU time (splits): 1,729,545 seconds
- Total GPU seconds: 5,644,281 seconds = 1,567.9 GPU hours
- Cost estimate (at $2/GPU-hour): $3,136

5.3 SENSITIVITY ANALYSIS

Cost is sensitive to per-operation inference time:

MOUSE (varying per-operation cost):

Optimistic (1.0-1.5 sec):	307-460 GPU hours = $600
Realistic (2.0-2.5 sec):	614-768 GPU hours = $1,200
Pessimistic (3.0-5.0 sec):	921-1,536 GPU hours = $2,000

FLY (varying per-operation cost):

Optimistic (1.0-1.5 sec):	783-1,175 GPU hours = $1,600
Realistic (2.0-2.5 sec):	1,568-1,960 GPU hours = $3,100
Pessimistic (3.0-5.0 sec):	2,352-3,920 GPU hours = $5,000

SECTION 6: VISUALIZATIONS

Figure 1: Mouse Edit Distribution (n=100)
File path: reports/edit_distributions/figures/edit_distribution_mouse_n100.png

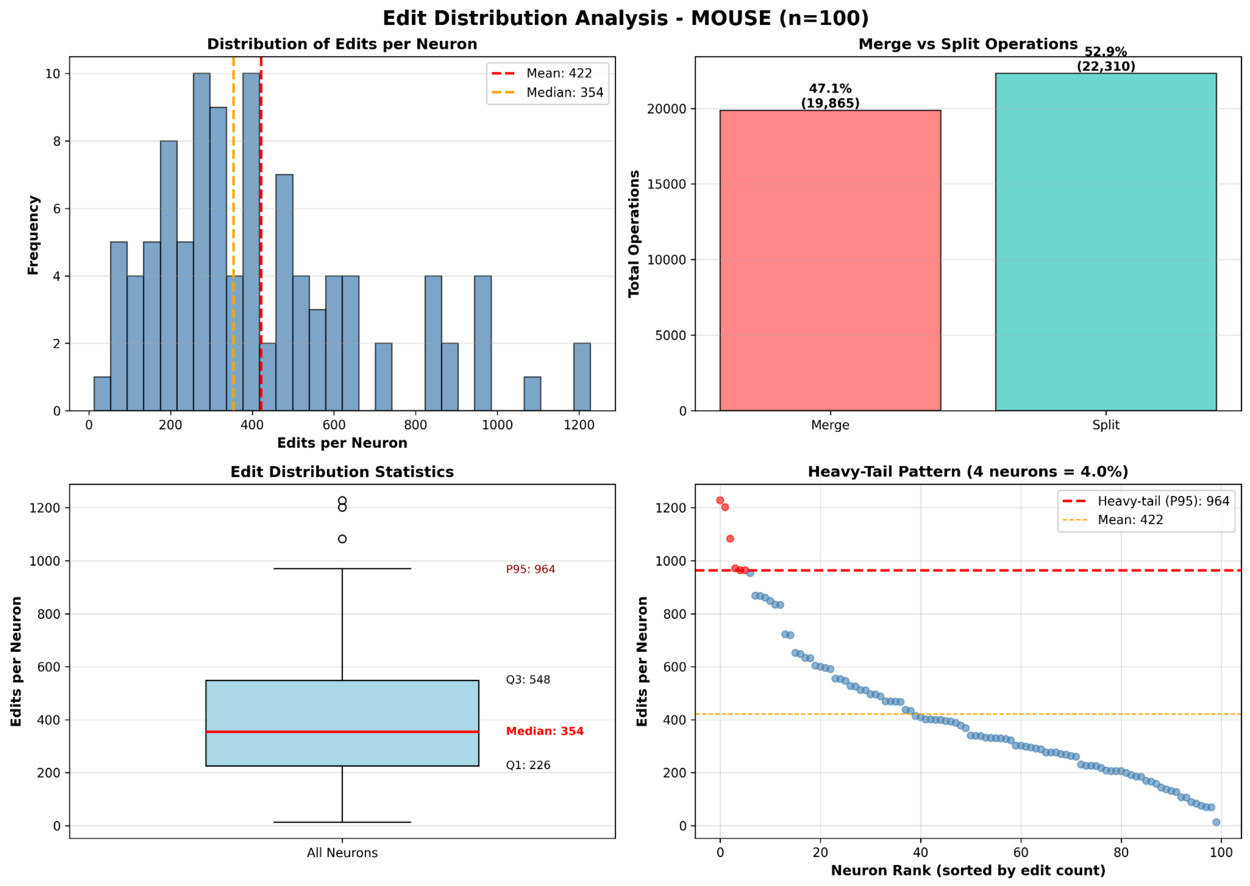


Figure 1 Caption:
Comprehensive 4-panel analysis of edit distributions for 100 sampled mouse neurons. Top-left histogram shows right-skewed distribution (mean=422 edits, median=354). Top-right bar chart shows balanced merge/split ratio (47%/53%). Bottom-left box plot displays quartiles and percentiles. Bottom-right scatter plot highlights heavy-tail neurons (>964 edits, n=4, 10.6% of edits).

Figure 2: Mouse Edit Distribution (n=500)
File path: reports/edit_distributions/figures/edit_distribution_mouse_n500.png

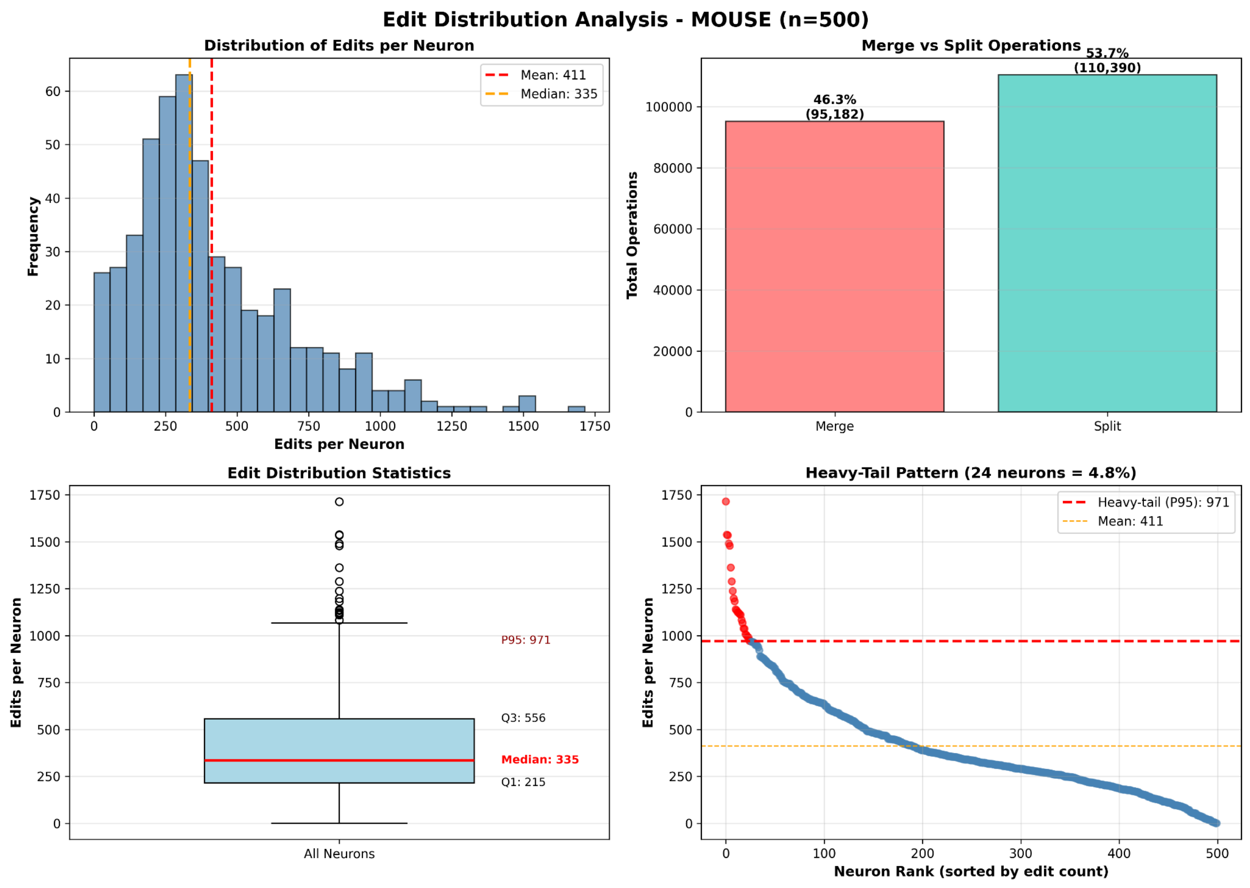


Figure 2 Caption: Comprehensive 4-panel analysis using larger n=500 mouse neuron samples to validate n=100 findings. Top-left histogram shows similar right-skew *… COMING SOON*

Figure 3: Fly Edit Distribution (n=100)
File path: reports/edit_distributions/figures/edit_distribution_fly_n100.png

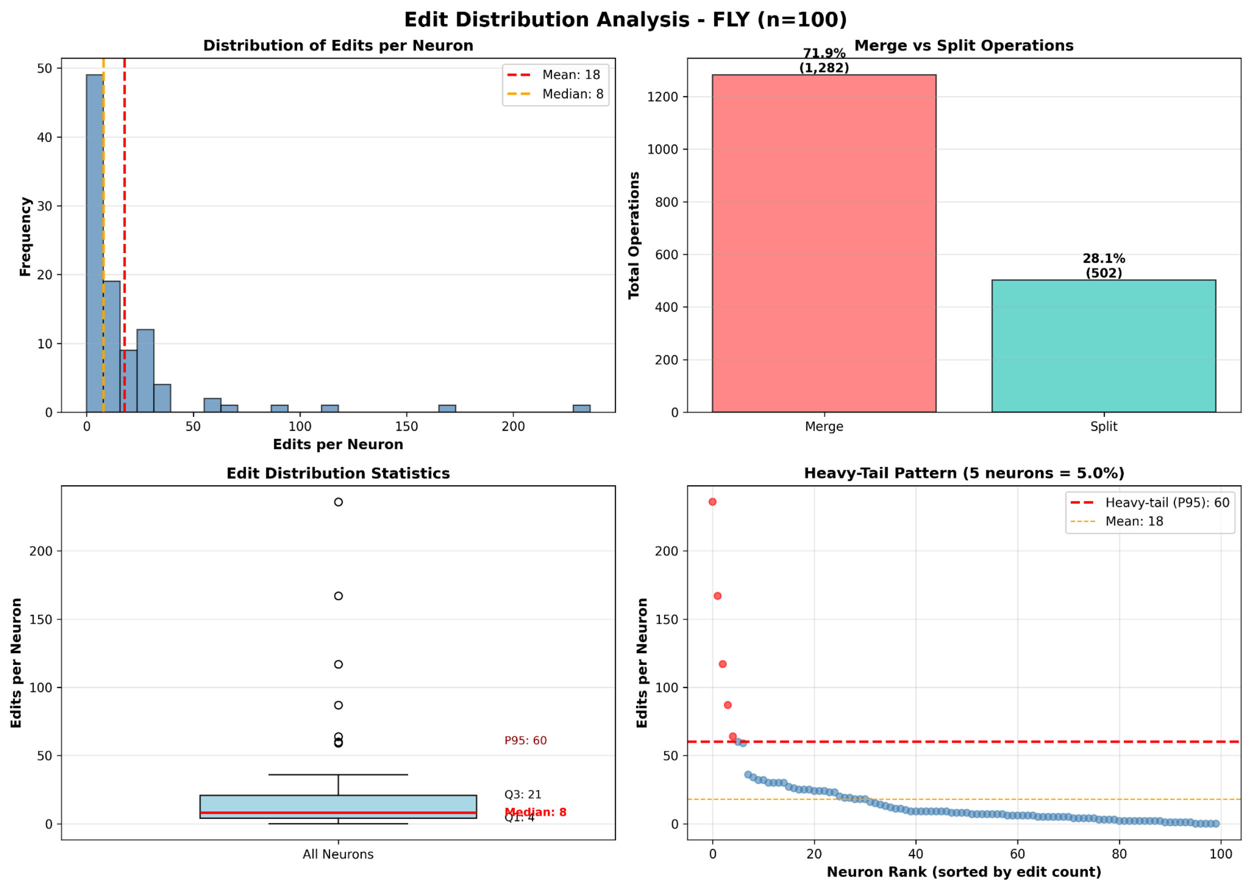


Figure 3 Caption:
Comprehensive 4-panel analysis of edit distributions for 100 sampled fly neurons. Top-left histogram shows extreme right-skew with concentration near zero (mean=18 edits, median=8). Top-right bar chart shows merge dominance (72% merge, 28% split). Bottom-left box plot displays tighter quartile range. Bottom-right scatter plot highlights heavy-tail neurons (>60 edits, n=5, 38% of edits).

Figure 4: Fly Edit Distribution (n=500 - Validation)
File path: reports/edit_distributions/figures/edit_distribution_fly_n500.png

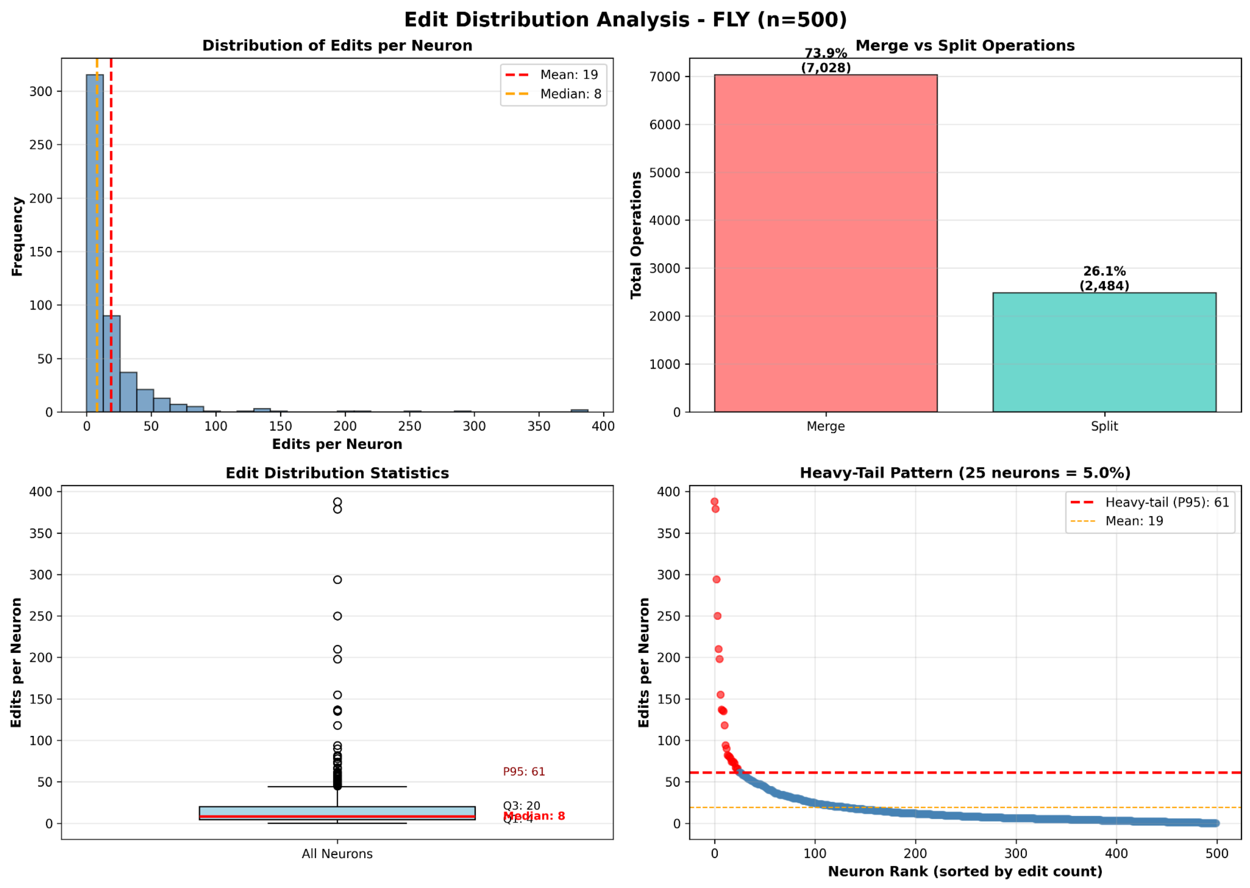
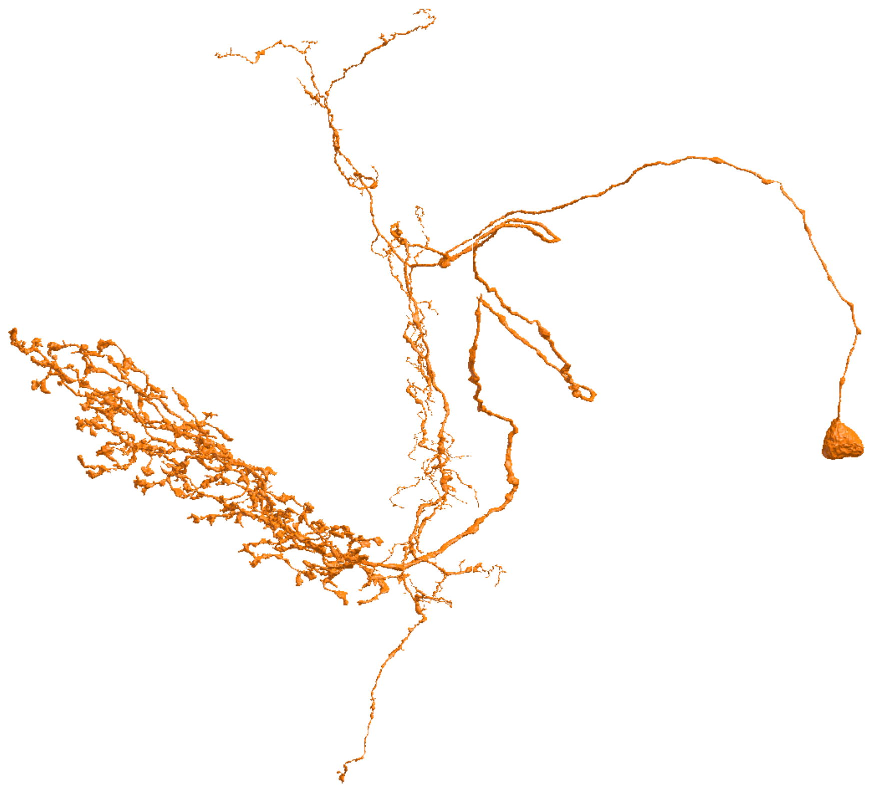
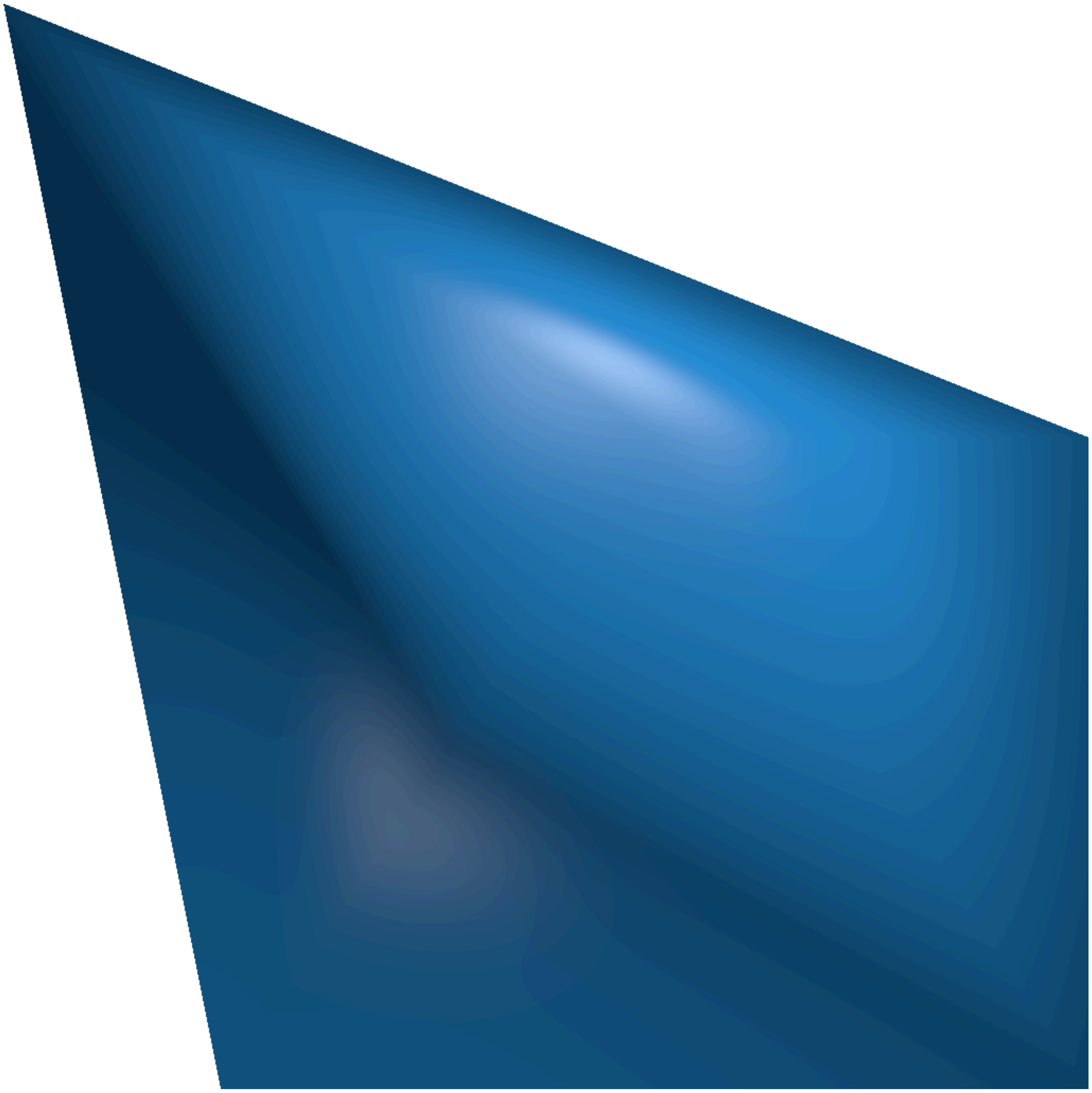


Figure 4 Caption:
Comprehensive 4-panel analysis using larger n=500 fly neuron samples to validate n=100 findings. Top-left histogram shows similar right-skew (mean=19 edits, median=8). Top-right bar chart confirms merge dominance (74% merge, 26% split). Bottom-left box plot matches n=100 percentiles. Bottom-right scatter plot shows 25 heavy-tail neurons (5.0%, 36% of edits), confirming outlier pattern.


3d_fly_neuron_720575940620919646_front


3d_mouse_neuron_front_864691128652664051.png

==

SECTION 7: LIMITATIONS & FUTURE WORK

==

7.1 CURRENT LIMITATIONS

LINEAR EXTRAPOLATION ASSUMPTIONS:
• Assumes per-neuron statistics are representative of full dataset
• May not account for systematic biases in sampling
• Cross-validation between n=100 and n=500 shows <7% difference (reassuring)

EDIT COMPLEXITY VARIATION:
• Analysis counts edit operations but not their computational complexity
• Some edits may require more time (e.g., large splits, complex merges)
• Per-operation cost likely varies by 10-100x depending on complexity

HEAVY-TAIL NEURON CHARACTERISTICS:
• Analysis identifies heavy-tail neurons but not their properties
• Unknown: What makes these neurons computationally expensive?

API RELIABILITY:
• FlyWire API occasionally returns errors
• Some neurons may have incomplete edit histories
• Error handling: Skip failed neurons, report statistics on failures

7.2 FUTURE DIRECTIONS

PRIORITY 1: COMPUTATIONAL BENCHMARKING
- Implement Qwen 32B inference profiling
- Measure per-operation inference time
- Quantify hardware costs (H100 GPU-hours)
- Enables accurate cost estimation and sensitivity analysis

PRIORITY 2: EDIT COMPLEXITY PROFILING
- Categorize edits by complexity
- Analyze relationship between complexity and inference time
- Build complexity-weighted cost model

PRIORITY 3: VALIDATION ON ACTUAL GPU DEPLOYMENT
- Run pilot proofreading system on 100-500 neurons
- Measure actual GPU time and costs
- Refine cost model with empirical data

PRIORITY 4: ADVANCED STATISTICAL MODELING
- Fit parametric distributions to edit counts
- Build Bayesian hierarchical model
- Quantify uncertainty in projections

For Jeff DATA AVAILABILITY & REPRODUCIBILITY

7.1 CODE REPOSITORY

GitHub: https://github.com/jffbrwn2/ConnectomeBench
Branch: compute-cost-analysis

Key implementation files:

- analysis/analyze_edit_distributions.py (main analysis script)
- src/connectome_visualizer.py (CAVEclient interface)
- examples/tutorial_edit_history_analysis.ipynb (example walkthrough)

7.2 DATA SOURCES & ACCESS

MOUSE (MICrONS):

- Website: https://www.microns-explorer.org/
- Datastack: minnie65_public
- Access: Public via CAVEclient (no authentication required)
- Neurons: ~2,314 extensively proofread

FLY (FlyWire):

- Website: https://flywire.ai/
- Datastack: flywire_fafb_public
- Access: Public via CAVEclient (no authentication required)
- Neurons: ~139,255 proofread (community effort)

7.3 REPRODUCIBILITY COMMANDS

ENVIRONMENT SETUP:

```
  git clone https://github.com/[organization]/ConnectomeBench.git
  cd ConnectomeBench
  python3 -m venv .venv
  source .venv/bin/activate
  pip install -r requirements.txt
```

MOUSE ANALYSIS:

```
  python analysis/analyze_edit_distributions.py --species mouse --sample-sizes 100 500
```

FLY ANALYSIS:

```
  python analysis/analyze_edit_distributions.py --species fly --sample-sizes 100 500
```

EXPECTED RUNTIME:

- Mouse n=100: 12-15 minutes
- Mouse n=500: 60-90 minutes
- Fly n=100: 3-5 minutes
- Fly n=500: 15-20 minutes
- Total (all analyses): 2-3 hours