OshonSoft AI Language Model Lab

User Manual

User Manual

OshonSoft Project
www.oshonsoft.com

Contents

1. Welcome
2. The main window
3. Guided Getting Started
4. Configuration
5. Model layout and memory
6. Text sources and input files
7. Tokens and context; Vocabulary
8. Forward inspection; Loss and gradients
9. Training
10. Conversation
11. Text continuation
12. Persistence
13. Project
14. Controlled training
15. Workflow and evaluation
16. Transformer inspection
17. Model Live View
18. Common questions and problems
19. Finish a session and continue learning
20. Evaluation sessions and license information

1. Welcome

OshonSoft AI Language Model Lab is a Windows application for learning how a small language model works. Prepare your own text, initialize a model, train it, inspect its calculations and try its replies. MLP and Decoder Transformer models share the same main workflows. Training and generation run locally on the CPU; no online AI service or pretrained model is required.

This manual covers the main window, Guided Getting Started and Model Live View. English control names match the interface. Start with a supplied guide, then use the page sections to look up individual fields and commands. A small model learns from the material and training you provide; useful replies must be checked on complete examples.

1.1 Installation and personal files

Use the Windows x64 installer on Windows 10 or later. The installed program includes examples, guides and PDF learning documents. Open the application from its Start Menu shortcut. The other shortcuts open Getting Started, OshonSoft AI Course and the uninstaller. This is a native Windows x64 application with no additional runtime dependencies.

Keep your projects and results in a writable personal folder. Installed examples are starting material. Guides that write results create a new session below the selected Results parent, using a name such as guide-07-name-0000001 and the first free number. Its training/, questions/ and runs/ folders belong directly to that session. Save important checkpoints and results before uninstalling or replacing a working project.

1.2 Three ways to start

Try a lesson: click Guided Getting Started, select a guide, read its first step and use Execute step. **Use learned weights:** on Persistence choose a supplied .osai and click Load checkpoint, then open Conversation. **Train your material:** apply dimensions, prepare text sources, initialize parameters and start Training. Loading a checkpoint already restores initialized weights; do not initialize it again just to generate a reply.

2. The main window

Control / display	Use
Conversation	Enter USER messages and generate MODEL replies.
Configuration / Model layout and memory	Choose a model layout; inspect active parameter groups and memory.
Text sources / Tokens and context / Vocabulary	Import text, inspect targets and browse the frozen dictionary.
Forward inspection / Loss and gradients	Inspect one prediction and its derivatives.
Training / Persistence	Learn, evaluate, save and restore learning state.
Text continuation / Workflow and evaluation	Continue a passage, compare predictions and measure speed.
Controlled training / Project	Run a saved training plan or a project with question checkpoints.
Transformer inspection	Browse attention, intermediate values and parameter traces.
Guided Getting Started / Model Live View / Close	Open either auxiliary window, or close the main application.
Status line and summaries	Read the result or error after each action. A disabled command usually needs a prerequisite or is locked by another operation.

Editable fields often describe a draft. The active summary describes the current model. Apply dimensions replaces the layout and clears the corpus. Prepare corpus replaces the corpus and vocabulary and invalidates initialized weights. Initialize parameters creates new seeded weights and resets learning progress. Save useful weights before these actions. Changing tabs or editing a draft does not itself perform them.

Training, automatic generation, evaluation and saved runs share one active model. Use the pause or stop command belonging to the operation that is running. A paused saved run still owns its plan. Old inspection data may become unavailable after a new target, update, initialization or load; perform the corresponding prediction or backward action again.

3. Guided Getting Started

A guide performs real operations in the Lab and explains each step. Save your current model before starting. Read the selected guide and keep its click targets visible. Automatic steps move the mouse and click application controls, so leave the computer to the guide while it is running.

Control / display	Use
Guide list / explanation / status	Select a lesson; read its current action, expected result and completion status. Available lessons depend on the installed guide files.
Results parent	Choose the writable parent for new guide sessions. This is locked during execution.
Fast / Normal / Slow	Choose choreography speed. This does not change the learning rate or the update budget.
Execute step / Run automatically	Execute the current step, or continue the sequence automatically. Read the result before advancing manually.
Pause guide / Resume guide	Pause or resume the guide sequence. Work already started in the Lab may continue.
Stop	Stop the sequence and pause work started by this guide. Read the main operation status before proceeding.
Open lab document	Open the selected lesson document when supplied; the button is unavailable when no valid document exists.

Lessons include narrative and mixed text, conversation, Transformer inspection and recovery, bit tasks and elementary arithmetic. Use the recipe printed in the lesson, including its initialization choice. A successful example demonstrates the specific tested task and lengths. Try longer inputs separately when you want to investigate length generalization.

4. Configuration

Choose the draft settings below, read the estimates and click Apply dimensions (zero arrays) to apply them. A preset fills fields only. It does not load weights, import sources or start training. MLP is the startup architecture.

Control / display	Use
Model preset / Custom	Select a supplied configuration or enter dimensions manually. Identify a preset by its displayed V/C/D/H, heads and blocks.
Architecture	MLP uses concatenated embeddings and a tanh hidden layer. Decoder Transformer uses causal attention, learned positions, normalization, residuals and a ReLU feed-forward layer.
Vocabulary slots V	Maximum token capacity, including seven controls. The populated count P can be smaller.
Context tokens C	Number of preceding tokens available to one prediction. Longer visible text can be outside this rolling window.
Embedding width D / Hidden neurons H	MLP: embedding width and tanh layer size. Transformer: model width d and feed-forward width F; F must equal 4d.
Transformer heads / Transformer blocks	Attention heads and decoder blocks. d must divide evenly by the head count. Unused for MLP.
Initialization seed	Integer 0..2147483647. Together with the applied layout and initialization choice, sets reproducible starting weights.
MLP initialization	Standard: embeddings +/-0.1. Wider embeddings: +/-1.0. Custom: -a to +a. All use Xavier weights and zero biases. Wider is used by the supplied XOR lesson.
Custom a	MLP embedding half-range, 0.001..10, with a decimal point. Applied only with Custom. This field does not modify already learned weights.
Draft summary / active summary	Compare proposed dimensions and memory with the configuration actually in use. The MLP initialization setting is saved with the checkpoint.

The current MLP preset list uses the following V/C/D/H dimensions. General MLP limits are V 7..1000000, C 1..4096, D 1..4096 and H 1..65536, subject to allocation guards. Use modest configurations first; field limits are not hardware recommendations.

Control / display	Use
MLP compact inspection	V256 / C16 / D16 / H64.
MLP compact conversation	V256 / C32 / D16 / H64.
MLP mixed sources	V1024 / C64 / D16 / H64.
Bits - NOT16 / Copy16	V64 / C64 / D8 / H64.
Bits - AND16 / XOR16	V32 / C64 / D8 / H192.
Arithmetic - Complement8 / Columns	V32 / C32 / D8 / H192.
Arithmetic - Local steps	V32 / C16 / D16 / H192.
English lesson - Getting Started / Live View	V256 / C32 / D16 / H96.

Transformer limits are V 7..4096, C 1..256, d 2..1024, heads 1..4 and blocks 1..2. F must equal 4d, d must divide evenly by heads, and d/heads must not exceed 256. Configurations must also fit 20 million trainable parameters, 8 billion main multiplications for a full-window forward calculation and 512 MiB of numerical workspace payload. Its two presets are listed below. Each prediction recomputes the current window; there is no KV-cache setting.

Control / display	Use
English lesson - Transformer	V256 / C32 / d16 / F64 / 2 heads / 1 block.
Model Live View - Transformer	V256 / C16 / d16 / F64 / 2 heads / 1 block.

5. Model layout and memory

The group table lists each active parameter group, its shape, zero-based index ranges, value count and FP32 bytes. MLP groups are E, W1, b1, W2 and b2. Transformer groups include token and position embeddings, normalization parameters, attention projections, feed-forward parameters and output parameters. Shapes give counts: a dimension of 16 has indices 0..15.

The memory summary separates FP32 parameter payload, FP64 gradients, work arrays, update traces and corpus storage. These figures are not total Windows process RAM. Import and staged checkpoint loading can temporarily need additional storage. There is no editable model parameter on this page.

6. Text sources and input files

Control / display	Use
New file mode	Choose Narrative TXT or Conversation before adding a file. Different kinds can share one project.
Source family	Name related material with the same family; keep that whole family in one split. Family is not a training weight.
Add TRAIN file / Add VALIDATION file	Add a draft row with its current parser and family. Only TRAIN builds the dictionary and receives SGD.
Remove selected / Clear draft	Remove one draft row or all draft rows. The prepared corpus stays active until another successful preparation.
Select mixed example / Select Conversation example	Fill source drafts from available supplied examples. Review them, then prepare explicitly.
Tokenizer: Words and punctuation	Keep number strings as whole tokens; split punctuation and normalize supported letters to lowercase.
Tokenizer: Words, punctuation and digits	Split ASCII digits, including digits inside identifiers. Standalone carry signs ↑0 and ↑1 each remain one token. Join markers preserve other digit adjacency when ordinary rendering would insert a space.
Preserve line breaks	Off by default. When enabled, body line breaks become learned <lf> tokens; EOT still ends the reply.
Prepare corpus	Read and validate the draft; on success replace corpus/vocabulary and reset learning. Initialize afterward. A rejected import preserves the previous project.
Source table / corpus summary / rules	Check split, kind, family, path, document and token counts, unknowns and input rules. To change a row, remove and add it with corrected choices.

6.1 File limits and boundaries

Use nonempty UTF-8 TXT; an initial UTF-8 BOM and LF or CRLF are accepted. The importer accepts 1..50 files, up to 1 MiB per file and 20 MiB in total. Additional limits are 5000000 lexical body tokens, 6000000 stream tokens, 100000 conversation documents, 200000 messages and 256 UTF-16 code units in one lexical token. Canonical checkpoint text has a separate 60 MiB limit. A narrative file forms one document; blank lines do not split it. Context stays within a document, but can cross earlier messages in the same conversation.

6.2 USER/MODEL conversation format

Start headers at the beginning of a line, in uppercase. Start with USER, alternate roles and finish with a nonempty MODEL body. Message bodies may span lines. Put === on a line between complete conversations; do not put it before the first or after the last conversation.

```
USER: Which connector does the amber beacon use?
MODEL: The amber beacon uses a round connector and a blue key.
USER: When may the technician move it?
MODEL: Move it after the connector is locked and the card is signed.
===
USER: Where is the inspection card kept?
MODEL: Keep the card in the cabinet beside the east entrance.
```

To include a literal body line beginning with USER:, MODEL: or ===, prefix that reserved line with one backslash. The parser removes the escape. Interactive Conversation input is already a USER body; marker-looking text there remains literal text.

6.3 Keep evaluation material separate

Assign whole files and related families to one split before import. The importer checks shared families, duplicate paths and identical message bodies across TRAIN and VALIDATION; repeated occurrences inside TRAIN are allowed. It cannot detect every paraphrase, shared subsequence or equivalent answer. Review those yourself. Keep final TEST questions outside repeated validation and model selection.

7. Tokens and context; Vocabulary

Control / display	Use
Document dropdown	Select a document by split, source, number and line span.
Token table / Previous / Next	Browse up to 200 stream tokens per page. Select a target row to inspect its preceding context. BOS is not a prediction target.
Normalized preview	Read up to 4096 characters of normalized text, rather than original file formatting.
Context / example summary	Read the preceding context newest first, with original position labels descending from C-1. PAD rows follow real tokens when the context is short. The target is excluded; at most 64 positions are shown. The numerical context still runs oldest to newest.
Vocabulary table / Previous / Next / summary	Browse frozen IDs, spellings, TRAIN occurrences and kind. Unknown counts explain material encoded as UNK.

Control IDs are fixed: PAD=0 (padding), UNK=1 (unknown), BOS=2 (document start), EOS=3 (document end), EOT=4 (message end), USER=5 and MODEL=6 (roles). Lexical IDs start at 7. Typing a control name in ordinary text does not insert that structural ID.

In the token table, Position locates the token in the document; Normalized spelling shows the imported spelling; ID is its encoded value; Vocabulary decode shows what that ID means to the model; Kind distinguishes lexical and structural tokens. A spelling can remain visible even when its encoded ID is UNK.

TRAIN frequency ranks lexical tokens; first TRAIN occurrence resolves ties. Only the available V-7 lexical slots are filled. Validation-only and unfamiliar prompt words become UNK. Tokenization normalizes supported case and punctuation; it does not translate or transliterate. Changing the draft V cannot expand a loaded dictionary. Preparing a new vocabulary changes token meanings and requires new weights.

8. Forward inspection; Loss and gradients

For a new project, click Initialize parameters (active seed). For an initialized or loaded model, select a target on Tokens and context and click Predict selected context. This computes one prediction without learning or generating a reply. On Loss and gradients click Compute backward (no update) to calculate its derivatives.

Control / display	Use
Forward context / Position p	Choose a zero-based context position and read its token ID.
D page (200) / embedding table	Browse embedding components; i=p*D+d identifies their concatenated MLP input coordinates.
Hidden page (200) / hidden table	Browse MLP weighted sums and tanh outputs.
Probability table / forward summary	Read logits and top 20 raw probabilities, target probability and loss. Softmax uses every populated ID, including controls.
Gradient group / Page (200) / derivative table	Choose a parameter group and page; read row/column, current FP32 value and analytic FP64 derivative. No parameter is changed.

A derivative tells how this example's loss changes with a parameter. Repeated context token IDs contribute to the same embedding row. A later update or changed selection can invalidate the selected calculation. Transformer intermediate arrays are on Transformer inspection.

9. Training

Control / display	Use
Learning rate	SGD step size, greater than 0 and at most 1. Use an ordinary decimal with a point, for example 0.003; no sign, decimal comma or exponent.
Blocks/epochs	Total goal, 1..1000000. Increase a completed goal to continue. This is not an additional-update field.
Learning targets	Narrative body/EOS + MODEL body/EOT learns narrative and replies; USER text provides context. Full document streams also learns USER bodies and framing. Choose before learning.
Training order / Apply scheduler	Choose source-balanced visiting or global shuffled epochs, then explicitly apply. Change before learning, or initialize fresh weights first.
One TRAIN update	Perform one scheduled example and commit one atomic SGD update. The scheduled target can differ from the manually selected target.
Start / resume; Pause; Stop / cancel	Train toward the total goal; pause at an atomic boundary; stop training or cancel evaluation. Completed learning remains.
Fast training	Reduce frequent GUI refreshes and show progress about once a second. Pause to inspect the latest update. The SGD rule and target order stay the same.
Evaluate fixed weights	Measure TRAIN and VALIDATION without SGD. Wait for completed results.
Context / sample / parameter group / Page (200)	Inspect the actual last update and choose a parameter page. Before FP32, Gradient FP64, After FP32 and Stored delta describe that update.
History / evaluation summary	History averages loss while weights were changing. Evaluation holds one weight state fixed and records its update count.

9.1 Source-balanced blocks and global epochs

Source-balanced visiting alternates narrative and conversation when both exist, and rotates sources within each kind. A short source may cycle repeatedly before a long one is exhausted. One block has the total selected TRAIN target count in updates, but need not visit each target exactly once. Per-source exposures in the summary show the actual mixture.

Global shuffled epochs visit every eligible TRAIN target once per epoch in a deterministic shuffled order. Longer sources then contribute more targets. The shuffle state and cursor are saved for exact continuation. Initialization, sampling and scheduling have distinct saved state; a seed alone is not a complete restart record.

10. Conversation

Load or train an initialized model, then type the USER body in the multiline input. Choose settings before starting a new reply. The transcript keeps earlier messages visible, but each prediction receives only the last C tokens. Generating a reply does not train weights.

Control / display	Use
Load example checkpoint	Load the supplied conversation example, replacing the current model after the usual ownership checks.
Policy	Greedy chooses the highest eligible score, with lower ID breaking ties. Seeded sampling draws from Sampling q.
Temperature	0.01..100. Lower values concentrate sampling; higher values spread it. Greedy choice still follows the highest score.
Reply tokens / Seed	Limit 1..4096 choices, including the stop token; seed 0..2147483647 starts a private stream for each new reply.
Send / continue reply	Prepare the entered USER message and generate; after One token steps, continue the same prepared reply. Its settings stay fixed.
One token / Stop	Prepare or advance one token choice; stop the reply and retain its partial text. A stopped reply cannot be resumed.
Add EOT to partial reply	Explicitly close a partial turn so another USER message can follow. The summary records the supplied boundary.
Bracket trace	On by default. Uses the training-style spacing for bracket traces when Words and punctuation and Preserve line breaks are active. It changes display only, not generated tokens or scores.
Clear conversation	Clear input/history without changing learned weights or update count. Required after EOS for a new conversation.
Summary / context table / candidates	Read stop reason, unknowns and context truncation. Before/Now compares windows around the latest choice. Candidates show top 20 Raw p, Sampling q and Choice.

EOT completes a turn, including a possible empty reply. EOS ends the document. A token-limit or user-stop outcome leaves a partial reply: add EOT or clear it. Maximum input is 32768 UTF-16 character units; a session holds at most 64 messages, 16384 stream IDs and 131072 body character units. Preparation reserves capacity for the requested reply.

Raw p covers populated vocabulary IDs. Sampling q masks PAD, BOS and role markers, then applies temperature; lexical IDs, UNK, EOT and EOS remain eligible. Sampling uses the full eligible vocabulary, not just the visible top 20. There is no top-k or top-p setting. A probability measures token preference, not factual confidence.

The Conversation input and transcript use a fixed Consolas font scaled for the display DPI. Conversation, continuation and Training context tables show the newest available positions first. Their position labels retain the actual numerical index; forward computation still reads context oldest to newest.

11. Text continuation

Control / display	Use
Load example checkpoint / Use narrative sources	Load learned continuation weights, or select narrative source drafts for later preparation. These are different actions.
Prompt	Up to 32768 UTF-16 character units. An empty prompt begins at BOS.
Policy / Temperature / Max tokens / Seed	Same ranges as Conversation. Settings become active with Prepare prompt.
Prepare prompt	Replace the continuation session and preview its first prediction without choosing a token or consuming a sampling draw.
One token / Generate / resume / Stop generation	Make one choice, run automatically or pause generation. A stopped continuation can resume with its saved context and RNG.
Output / summary / context / candidates	Read output and outcome; Choice and Rolling show pre/post contexts. Candidates add Logit z and eligibility to p and q.

EOS or the maximum-token limit finishes this session; prepare a new one to continue with different settings. Continuation masks EOT as well as PAD, BOS and role controls. Output uses normalized token spacing. Weights remain unchanged.

12. Persistence

Opening an .osai file from Explorer or the command line loads the checkpoint and shows Persistence. Read its restored state there, then select Conversation or another page. Automatic training and generation remain off.

Control / display	Use
Path / Choose save path / Choose load file	Enter an .osai path or select it. The save parent must already exist and be writable. Choosing a path alone performs no save or load.
Save checkpoint	Save model, vocabulary, embedded corpus, settings, scheduler, learning counters and saved training history. Verify the success summary.
Allow replacement of an existing file (keep a backup)	Permit deliberate replacement and retain the previous file. The option clears after the attempt; read the backup path.
Load checkpoint	Validate the whole candidate before replacing active state. Architecture and tokenizer restore from the file. Training remains idle.
Checkpoint summary	Read saved/restored update, active settings, path and any validation or file error.

MLP and Transformer use one current checkpoint format with explicit architecture, tokenizer, schedule and initialization fields. Use checkpoints supplied with the current release. The embedded corpus permits continuation without the original TXT files; reimporting would read the source paths again.

Interactive transcripts, prepared continuation sessions, inspection work arrays, last-update traces and Live View histories are transient. Save useful generated text separately. A checkpoint save does not save later updates automatically. For a saved controlled run, preserve its entire run folder and recover through the matching run page.

13. Project

Use Project to prepare a model, train it and retain complete replies at chosen checkpoints. The page follows Preparation, Training and Results. Work in a writable project folder in Documents/OshonSoft AI Language Model Lab, or another folder you choose.

13.1 Prepare the inputs and settings

Put training TXT files in training/ and Training checks in questions/. A question contains one USER prompt and one MODEL Gold answer; separate pairs with ===. Gold is used only for scoring. Optional final-only questions belong in questions/final/ before the training run starts. They are copied with the run and kept out of its routine checks.

Training files are conversations by default. For mixed input, training/source_kinds.tsv lists every TXT filename and its narrative or conversation kind, separated by an actual TAB. Up to 50 TXT files can be inventoried in each input folder; each file is limited to 1 MiB. The declaration is limited to 16 KiB. Overall corpus limits also apply.

Choose project... selects the folder. If experiment.json exists, its saved choices fill the draft controls. Review the architecture, dimensions, initialization, seed, tokenizer, Preserve line breaks, objective and training order. Prepare and initialize model then creates fresh weights and prepares the TRAIN corpus. Save any model you want to retain before preparing another project.

13.2 Train to a total target

Control	Use
Total target / Epochs / Updates	Set the total endpoint. Updates is an absolute count. Epochs is converted using this project's scheduled-target count.
Learning rate	Set the SGD rate for the run.
Reply limit	Maximum generated tokens per question, including stopping tokens. A limit stop differs from EOT.
Check every N epochs / update list	In Epochs mode, enter a positive interval. In Updates mode, enter increasing absolute checkpoints separated by commas. Starting and final checkpoints are always included.
Start NEW run	Start from the prepared zero-update model in a new run folder.
Pause + save / Resume	Save and continue this same run with its existing target and answer plan.
Stop + save	End this run while retaining its saved state and completed results.
Continue to target	From an opened completed or stopped run, create a linked run from its selected checkpoint to a larger total target.
Fast training	Use less frequent interface refresh while preserving the training calculation and order.

For example, a checkpoint at update 1000 continued to target 1500 receives 500 more updates. Resume keeps the original target; it does not add the entered number again. During answer generation, Pause + save retains completed replies. The interrupted reply restarts from its prompt on Resume.

13.3 Inspect and compare results

Control	Use
Choose saved run... / Open / recover run	Open run.json and its saved files. A resumable run opens paused; Resume continues it. A completed run opens for inspection.
Open run folder	Open the current run's files.
Use checkpoint...	Select a checkpoint from this run for Conversation, continuation or Final evaluation.
Read replies... / Errors only	Read the saved Prompt, Gold and Raw records, optionally showing only mismatches.
Compare runs / Epochs / Updates	Display saved scores and question hashes on the chosen axis. Compare matching question sets.
Summary	Read the current phase, completed updates and answer counts. Reaching the training target may still leave final replies to generate.

Keep run.json, settings.txt, summary.txt, inputs/, checkpoints/, answers/ and state/ together. Raw contains the actual generated tokens, including controls. Exact correctness requires the whole expected answer and a generated EOT. The application does not fill in missing answer tokens from Gold.

13.4 Use Final evaluation when needed

Choose the model using Training checks, open its run, select Use checkpoint... and then Final evaluation. This generates answers to the frozen final questions without training. The result is a separate run linked to the selected model. If no final questions were supplied, this step is unavailable and can be omitted.

The application rejects exact normalized final prompts already found in training or checks, including duplicates inside the final set. Review equivalent tasks yourself when preparing the splits. A previously evaluated final question set is marked diagnostic when used again; it is no longer an untouched final check for model selection.

13.5 Compare several configurations

Set the model configuration and learning rate, then Add configuration to comparison. Repeat for the configurations you want to compare, up to 20. Start comparison list trains them sequentially using the common Epochs target, check interval, tokenizer, objective and training order. Each configuration starts from its own initialization and produces its own ordinary run.

The comparison JSON in runs/ retains the current run and the remaining configurations. Open / recover run can recover this list; Resume continues it. Stop cancels the remaining entries. Clear comparison list clears an unstarted list. Input changes between child runs stop the comparison rather than silently comparing different training material.

13.6 Create addition material

Create addition... prepares plain three-digit addition TXT files in an empty writable folder. Enter training, check and optional final counts and a data seed. Their combined limit is 100000 pairs. The data seed controls the examples and is independent of the model initialization seed. Swapping the two operands does not move a pair into another split. After creation, review model settings and prepare the project; creating files does not train a model.

14. Controlled training

Controlled training continues the active initialized project and chooses a best checkpoint by fixed-weight VALIDATION loss on selected targets. It requires TRAIN and VALIDATION targets. It does not prepare a fresh corpus or reinitialize. Use Project instead when the purpose is to save complete question replies at fixed updates.

Control / display	Use
Select mixed example sources	Fill source drafts. Prepare and initialize through the ordinary workflow.
Stop mode / Additional updates / NEW total update target	Updates freezes current count plus the requested additional updates. Duration uses the time and block ceilings. The total preview is read-only.

Control / display	Use
Enable time safeguard / Maximum minutes	In Updates mode optionally add a 1..10080 minute ceiling. Duration mode requires it.
Enable block/epoch safeguard / Blocks/epochs	In Updates mode optionally add an absolute block/epoch ceiling, 1..1000000. Duration mode requires it.
Validate / updates	Periodic fixed-weight validation cadence, 1..10000000 updates.
Patience (0 = off) / Min improvement	Stop after this many validations without sufficient loss improvement. Zero patience disables early stopping. Minimum is an ordinary decimal 0..1.
Snapshots at absolute updates	Up to 16 unique ascending comma-separated update counts after the start and within the permitted run bound.
Output limit MiB / results path / Choose results folder...	Reserve output capacity 1..8192 MiB and select the run folder.
Check preflight	Review actual dimensions, active rate, objective, targets, limits and output allowance before starting. No SGD starts here.
Start NEW run / Pause run + save / Resume run / Stop run + final save	Freeze a new plan, save a pause, resume the existing plan or terminate with final save.
run.json path / Choose run.json... / Recover run (paused)	Restore the committed coordinator and checkpoint, paused, with the original absolute target.
Run summary	Read state, elapsed time, next validation, latest/best roles, stopping reason and COMPLETE/PENDING/UNAVAILABLE terminal metrics.

The run uses the active learning rate. Merely editing Training fields does not accept a new rate; Check preflight shows the actual value. If you deliberately accept a new rate with One TRAIN update first, count that update as already completed. Best selection uses lower selected-target validation loss; ties within 1e-12 keep the earlier checkpoint. A smaller improvement can become best without resetting patience. Elapsed time excludes explicit pauses and includes validation and file operations.

14.1 Files and restart

Keep run.json, initial.osai, latest_a.osai/latest_b.osai, best_a.osai/best_b.osai, final.osai when published, planned snapshots, events.tsv and run_report.txt together. run.json names the committed latest and best roles; do not choose slots by modification time. Best can be earlier than final. Reaching the final update can precede terminal validation and saving; wait for the final status.

To restart either saved-run workflow: pause and save if still open, reopen the program, select the run.json on its matching page, recover paused, verify the saved count and unchanged target, then Resume run. For a run starting at 100 with 900 additional updates, the target is 1000. Recovery at 450 leaves 550 updates. Do not add 900 again. Copy complete run folders only while inactive. Preserve a rejected folder and error; do not edit counters or swap checkpoint files to force recovery.

Completed or deliberately stopped plans are terminal and cannot be resumed through run recovery. Load a suitable saved .osai through Persistence for inference or a deliberately new training plan.

15. Workflow and evaluation

Control / display	Use
Narrative setup (draft) / Mixed setup (draft)	Select a proposed setup for later explicit preparation.
Compare fixed weights	Measure the current model and TRAIN-only add-one unigram reference; inspect full-stream and MODEL-reply diagnostics.
Stop comparison / benchmark	Cancel the active comparison or benchmark. Completed learning from a training benchmark remains.
Benchmark count	Requested 1..100000 updates or token choices.
Benchmark training (real updates)	Time real SGD with current Training settings. It can reach the block/epoch goal before the requested count.
Benchmark generation (private)	Time a private generation session using Text continuation prompt/settings without replacing its interactive session or changing weights.
Comparison / benchmark summaries	Read completed target counts, update stamp, settings and timing scope before comparing results.

Loss is mean cross-entropy in nats; perplexity is its exponential. Token accuracy counts raw-argmax correct targets. Evaluation supplies correct preceding tokens at each prediction, while a free reply uses its own earlier choices. Check complete reply correctness and ending separately. Full-stream, selected-target and MODEL-only results have different target sets. Source-balanced learning exposure also differs from evaluation that counts each target once.

The unigram reference ignores context. Compare matching scopes and completed metrics. Benchmark timings depend on architecture, dimensions, context, vocabulary and observation conditions. Training timing includes educational update traces; private generation timing excludes text rendering. Close Live View when comparing runs that should have the same observation overhead.

16. Transformer inspection

Control / display	Use
Block / Head / Query position	Choose zero-based decoder block, attention head and local query row. Real non-PAD positions rebase from zero in each rolling window.
Selected forward / backward	Use the prediction selected on Tokens and context and its backward calculation.
Latest private prediction	View the latest copied generation/evaluation prediction when available. Selection does not execute another prediction.
Matrix chooser / value table	Select an intermediate row; read each feature and its derivative when available. Normalized-coordinate views include inverse standard deviation in the extra final feature.
Attention table	Read key positions/tokens, score or causal mask, and attention weight for the chosen query/head.
Parameter group / page / trace choice	Browse current backward gradients or Last atomic SGD trace: before / gradient / after. These belong to their own recorded calculations.

The 17 matrix views are Token + position; LN1; LN1 normalized coordinates / inverse stddev; Q; K; V; Joined heads; Attention output projection; First residual; LN2; LN2 normalized coordinates / inverse stddev; FF1 preactivation; ReLU; FF2 output; Second residual / block output; Final LN; Final normalized coordinates / inverse stddev. Q/K/V feature arrays contain all heads side by side. Head selects the attention row. Use Compute backward to obtain selected derivatives and One TRAIN update to obtain a new update trace.

17. Model Live View

Model Live View observes successful training updates. Open it, enable Live and perform a TRAIN update to obtain a sample. Prediction or backward inspection alone does not fill it. The view copies real calculations; it does not run extra training or predictions. Training may complete many updates between observations.

Control / display	Use
Mode	MLP: Learning changes, Signal contributions or One SGD Step (MLP). Transformer: attention path. Architecture follows the active model.
Live / Freeze	Enable new samples or hold the current diagram/history. Freeze does not pause training; Live does not start it.
10 / 15 / 20 seconds	Choose how long a selected group remains before reselection on an eligible real update. Disabled while Timeline pins a group.
Capture next step	In MLP step mode, clear the old snapshot and wait for the next successful TRAIN update.
Previous phase / Next phase / Play / Stop	Browse eight phases of the retained calculation. Play advances every 2.5 seconds and stops at phase 8. It performs no SGD.
Show Timeline / Hide Timeline	Show five plots and pin a group, or discard that history. A new observed update supplies the first point.
New tracked group	Clear history/scales and request a group on a later eligible update; keep the Live/Freeze choice.
Resize / maximize / minimize / close	Enlarge for detail. Closing discards observer data and leaves the trainer under its main controls. Minimize alone does not disable capture.

17.1 MLP diagram

Read the sample update, target and loss in the header, then follow tracked token, embedding E[token,d], concatenated input x[pD+d], hidden tanh and output logit/raw p. Up to three identities per column are selected; the rest of the dense model still participates. A tracked token ID and a fixed context position are different identities: when another word occupies the slot, its lookup line disappears, while the learned embedding still exists.

Learning changes selects strong gradient paths and makes line width follow the actual rounded parameter delta. Signal contributions selects strong forward products and makes width follow their magnitude. Blue means positive pre-update weight, orange negative, grey lookup/zero. Contribution sign can differ from weight sign. The width reference is fixed for a group; compare printed values across different groups. A later target can differ from the group's fixed output IDs.

17.2 One SGD Step (MLP)

The retained phases show: 1 Context and target; 2 Embedding lookup and concatenation; 3 Hidden weighted sum and tanh; 4 Output scores and stable softmax; 5 Cross-entropy loss; 6 Output gradients and hidden chain rule; 7 Input gradients and shared embeddings; 8 One atomic SGD update. Omitted terms remain included in the recorded sums. Repeated token occurrences add into one shared embedding derivative.

Snapshot update identifies the retained calculation even if training continues. Activations, probability, loss and gradient are PRE-update; parameter AFTER is the rounded stored FP32 result of before minus learning-rate times gradient. A nonzero gradient can produce zero stored delta after rounding. Changing

phases only changes presentation. Entering this mode clears Timeline; loading/resetting a model clears the snapshot and capture request.

17.3 Transformer attention path

The header identifies block, head, query, V feature and real context length. The heatmap shows up to eight positions: numeric cells are real allowed attention weights, MASKED marks future keys, and n/a marks absent positions. The selected query row has a blue outline. The displayed subset is not renormalized, so its row need not sum to one.

Follow Q/K to scaled score, attention alpha and signed alpha*V. Shown mass plus omitted allowed mass describes the selected paths; shown signed contributions plus omitted signed terms give the copied head result. Alpha is nonnegative, while V and its contribution can be negative. Wq/Wk/Wv records are sampled learned projection weights, not Q/K/V activations. Attention alpha, output Raw p and generation Sampling q have different meanings.

17.4 Learning Timeline

The five plots show parameter AFTER, true derivative PRE, rounded delta, one activation and one output signal. MLP tracks fixed E/W1/W2, hidden tanh and raw output p. Transformer tracks Wq/Wk/Wv, pinned attention alpha and signed alpha*V. Blue/green/orange in the first three plots identify parameters rather than sign. Read the fixed indices and current token occupants beside the plots.

The X axis uses absolute update distances and retains 256 observed samples. Lines connect observations; skipped updates have no recorded values. Y scales are fixed from the first sample. A boundary square and CLIPPED mean a value exceeds the drawing range, not that the model value was clamped. An absent query/key creates a signal gap; a true zero is drawn as zero. Live capture and presentation are throttled around 125 ms, so this is a sampled view rather than a complete update log.

Freeze retains history; Hide Timeline, closing the window, entering step mode or replacing model/corpus discards the corresponding temporary data. Checkpoints do not save these plots. A changing TRAIN example can change p or attention even without improvement on a fixed question. Use evaluation and complete replies for that comparison.

18. Common questions and problems

Control / display	Use
Reply is poor although loss fell	Check unknowns, context truncation, training material and complete free replies. Try the supplied recipe before changing several settings together.
Many UNK tokens	Inspect Vocabulary and tokenizer. A larger V needs a fresh import and new weights; save the old model first.
Cannot send another message	Read the outcome. Add EOT to a partial reply, or Clear conversation after EOS.
Disabled controls / unavailable arrays	Finish or pause the owning operation. Predict/backward again for a changed selected target.
Live View is waiting or frozen	Enable Live and perform a real successful TRAIN update. Freeze holds observation only. In step mode capture a new MLP step.
Split import rejected	Read the reported family, file and lines. Move related material to one split and resolve duplicates before reimporting.
Save or recovery failed	Keep the full folder and exact error/path/Windows code. Check write access and files held by another program. Retry only after resolving the cause; do not edit run.json.
Run reached its update target but is still busy	Final question generation, validation or saving may still be running. Wait for the completed saved result.
Want to keep a useful answer or graph	Copy useful text or notes separately. Interactive transcripts and observer histories are not checkpoint data.

19. Finish a session and continue learning

Stop automatic generation. Pause and save a saved run through its own controls, or pause ordinary Training and save a checkpoint. Verify the path and update count, then close. Use the same run.json to continue an unfinished run; load an individual .osai when you only need its model state.

Getting Started supplies practical lessons. OshonSoft AI Course explains the mathematics. This User Manual remains the place to look up application fields and commands, including Model Live View.

20. Evaluation sessions and license information

The evaluation copy allows 30 starts, with 180 minutes per session. An internet connection is needed to confirm each start. Cancelling the initial question does not use a start. Help > View Evaluation Information shows the current start number and session duration.

A warning appears three minutes before the session ends. At expiry the application stops active work and opens Evaluation Information. Click Save model before closing, choose a writable .osai path and wait for the saved-and-verified message. A recovery-save error is reported explicitly; keep the application open

and save the model to another file before closing. A saved Project can be recovered in the next session and resumed to its original target.

For a licensed copy, Help > View License Information shows the owner and license details. Covered releases remain usable after the update period ends. An institution license has permanent updates. If license entry is requested, copy the exact License Owner and the 25-digit License Code from the email.

Personal and commercial activation is attempted automatically. When it succeeds, the information window contains only license details. If it could not be completed, the same window offers an online retry and the offline owner-hash/token fields. For offline activation, send the displayed hash to OshonSoft and enter the returned 12-character token. Institution licenses do not show these activation controls.