Bracket Logic

OshonSoft Project
www.oshonsoft.com

The task and measured result

Give the model a string containing round and square brackets. It reads from left to right, records the open brackets and remaining input, and generates a verdict. It must identify the first incorrect closing bracket, or list the closing brackets still needed for an incomplete string.

The final MLP has **26,910 parameters** and learns from **4,000 conversations**. Both the primary seed and the independent confirmation generated **1,200/1,200** complete final replies correctly, including EOT. The measured task covers strings of one to ten bracket characters.

The primary and confirmation experiments have been repeated with the current Lab training and generation code. Both reproduced 1,200/1,200 exact final replies and 4,000/4,000 exact training replies. The download includes their initial and selected models and complete saved answers. The final questions had already been evaluated in the original study, so this repetition checks reproduction of that result.

Read the trace

The left field shows open brackets, most recent first. The right field shows the remaining input. Dashes fill unused cells. The `count` line gives the model a short number sequence for aligning its copy of the input. Positions begin at 1.

A wrong closer produces `invalid`, the first error position and the expected closer. If no bracket is open, the expected value is `none`. An `incomplete` result gives the remaining closers in the order they should be appended.

These are actual final-line excerpts from saved native replies to two inputs that were absent from training:

```
USER ([])[]([]]
result: invalid at 10, expected ).

USER ([])[]([])
result: valid.
```

The reply is generated by the model. The Bracket trace display changes spacing for reading; it does not supply or repair tokens. Exactness is measured on the full token sequence and correct stopping.

What changed during the study

The same sampling rules generated training, Training checks and Final evaluation inputs within each recipe. There were 4,000 training strings, 600 check questions and 1,200 final questions. A string and the version with round/square symbols exchanged stayed in the same split.

The original 4,000-example recipe reached 1,199/1,200 on two independent tests. One deep closing step still failed after both four and six epochs. That training set contained only two legal closes at position 10 with eight openings left. A later deep-close recipe passed with seed 12345, but its confirmation missed two traces with ten simultaneous openings; only six such strings appeared in training.

The final sampler increased deep opening and closing cases in all three folds. Its training set contains 225 strings with ten simultaneous openings and 64 legal closes at position 10 with eight openings left. The comparison below keeps the original results.

Material	Model / parameters	Epochs	Training checks	Final evaluation
Random 2,000	MLP 100,820	4	596/600	1,192/1,200
Random 2,000	MLP 26,910	4	596/600	1,191/1,200
Random 4,000	MLP 100,820	2	600/600	1,198/1,200
Random 4,000	MLP 26,910	2	600/600	1,198/1,200
Shared random comparison	MLP 26,910	4 / 6	600/600	1,199/1,200
Shared random comparison	Transformer 25,720	1	416/600	791/1,200
Deep opens and closes	MLP 26,910	6	600/600	1,200/1,200

Rows sharing a named question suite are paired comparisons. Different material recipes had their own frozen final suites. The Transformer row is its selected first-epoch checkpoint from a two-epoch run.

Learning progression and confirmation

Checkpoint	Updates	Whole reply	Correct EOT
Before training	0	0/600	287/600
Early check	200000	484/600	600/600
Epoch 1	784721	600/600	600/600
Epoch 2	1569442	600/600	600/600
Epoch 3	2354163	600/600	600/600
Epoch 4	3138884	600/600	600/600
Epoch 5	3923605	600/600	600/600
Epoch 6	4708326	600/600	600/600

Every complete-epoch checkpoint passed all 600 check questions in the final recipe. The study selected the sixth epoch. Seed 23456 used fresh weights and shuffled order, with the same recipe, and also scored 600/600 on checks and 1,200/1,200 on final questions. All 4,000 known training replies were exact.

The primary final set contained 733 even-length and 467 odd-length inputs. All 930 wrong or unexpected closes were detected at the exact first-error position. Replies were also exact for all 217 cases whose error prefix was absent from training, and for both equal-symbol-count contrast pairs. These are measured subsets of the final 1,200 questions.

Files and settings

Each project has one `experiment.json`, a `training/` folder and a `questions/` folder. Final questions are under `questions/final/`. The released snapshots belong in `models/initial.osai` and `models/selected.osai`. A new execution creates its own `runs/run-000001/`, with settings, frozen inputs, checkpoints and complete Prompt/Gold/Raw answers. The download supplies those inputs and both snapshots for `bracket_logic` and `bracket_logic_confirmation`. Supplied replies are grouped by update under `answers/`; `answers/final/` contains all final replies. Files keep their original question names and complete Prompt/Gold/Raw records.

Use a writable folder such as Documents/OshonSoft AI Language Model Lab. The application stays in Program Files. Training results are written to the project folder and do not require administrator rights. Larger research projects are separate downloads; they are not part of the ordinary-example installer.

The project folder is `bracket_logic`. Preserve the original filenames `brackets_01.txt` through `brackets_20.txt`, `known.txt`, `dev_even.txt`, `dev_odd.txt`, `test_even.txt` and `test_odd.txt`. The `dev_` names denote Training checks; `test_` files stay under `questions/final/`.

Setting	Value
Architecture / dimensions	MLP; V40 / C28 / D12 / H70
Parameters	26910
Model seed / shuffle seed	12345 / 12346
Initialization	Custom embedding half-range 2.0
Tokenizer	Words and punctuation; Preserve line breaks on
Scheduler / objective	Global epochs; MODEL body and EOT
SGD rate	0.006
Targets / epoch	784721
Target	6 epochs = 4,708,326 updates
Early check	200000
Other checks	Every complete epoch
Generation	Greedy; 512 tokens
Confirmation seed	23456

Repeat the training

On an AMD Ryzen 7 7735HS laptop, the repeated primary run took about 2 h 46 min. These times include training and all configured checks while several projects shared the CPU. Time spent paused is excluded. Other hardware and workloads will give different timings.

1. Extract the chosen project into the writable application folder. Open the installed Lab and select the **Project** page.

2. Click **Choose project...** and select the folder containing `experiment.json`. The settings fill the controls as a draft. Review them before preparing the model.

3. Click **Prepare and initialize model**. Read the actual training-target count. The prepared recipes use globally shuffled epochs and MODEL body plus EOT targets.

4. Review **Total target**, its **Epochs/Updates** unit, **Learning rate**, **Reply limit** and **Check every N epochs / update list**. Start a fresh run. Update zero and the endpoint are included in the saved checks.

5. Follow the saved checkpoints and answers with **Open run folder** and **Read replies...**. Use **Pause + save** before closing. To recover, choose the same `run.json`, click **Open / recover run**, then **Resume**. Resume keeps the original target.

6. Apply the study's checkpoint-selection rule. Load that checkpoint with **Use checkpoint...**, then choose **Final evaluation**. Keep final questions outside the ordinary checkpoint checks. Record every wrong reply as well as the correct ones.

This project uses Updates, target `4708326`, with the extra list `200000,784721,1569442,2354163,3138884,3923605`. The final target is added automatically. For the independent confirmation, choose the supplied `bracket_logic_confirmation` project. It uses seed 23456 with the same remaining settings.

Execution time depends on the CPU and other running work. Follow the current epoch and update count on the Project page; Pause + save preserves the remaining work.

To try the supplied model, load `models/selected.osai` on Persistence. Use Greedy, reply limit 512 and Bracket trace on. Clear the conversation before each independent input, then click Send / continue reply. Compare the complete generated trace, verdict and EOT.