LLM Optimization 1

Haw-Shiuan Chang

Deadlines

- **https://people.cs.umass.edu/~hschang/cs685/schedule.html**

- **2/17:** Quiz 1 due
- **2/14**: HW 0 due
- **2/14**: Final project group assignments due
  - https://forms.gle/PKvJRxZkUMgFrkVG8
  - If your team is less than 4 people, you can fill NA. However, we might need to split some teams that are less than 4 people.
    - e.g., 3, 3, 2
- The link of the NLP seminar will be posted at Piazza

Logistics

- Office Hour Correction
  - Erica: Tues 4-5pm, CS207 Cube 2
  - Ankita: Wed 4-5pm, CS207 Cube 2
  - Haw-Shiuan: Thu 11AM-12pm, CS207 Cube 2
  - Nguyen: Fri 3pm-4pm, CS207 Cube 2

- The deadline for asking for the SAT/Fail score will be one week after the midterm scores are released.
  - The students who want to have the SAT/Fail score will need to send an email to cics.685.instructors@gmail.com.
  - After the deadline, you cannot switch to SAT/Fail score or switch back to the normal letter score.

Task -> Loss -> Model -> Optimization

- Task:
  - Predict the next token
- Loss:
  - Maximal Likelihood / **Cross-entropy**
- Model:
  - Tables -> Neural Network -> Transformer
- Optimization:
  - Counting -> **Gradient Descent**

PyTorch Optimizer

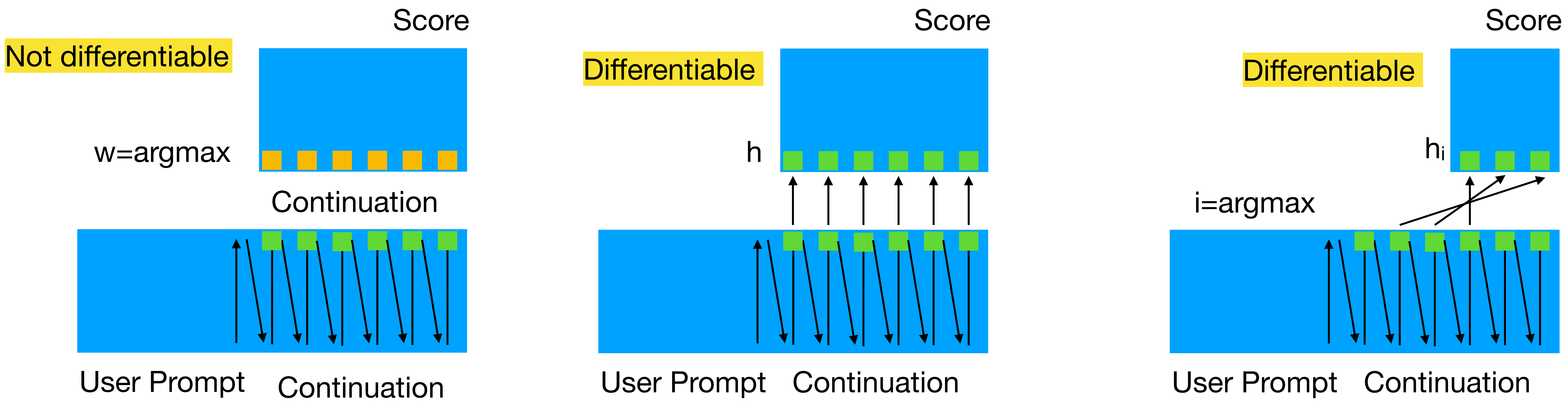
- Implementing forward pass and automatically generate the backward pass

- Tensorflow usually cannot dynamically adjust the architecture of NN, but it is easier to deploy.

```
# Clear the previously calculated gradient
model.zero_grad()

# Perform a forward pass (evaluate the model on this training batch).
outputs = model(b_input_ids,
                      token_type_ids=None,
                      attention_mask=b_input_mask,
                      labels=b_labels)
loss = outputs.loss
logits = outputs.logits

total_train_loss += loss.item()

# Perform a backward pass to calculate the gradients.
loss.backward()

# Update parameters and take a step using the computed gradient.
optimizer.step()
```

What is Differentiable?

- Gradient descent is the easiest and most stable way
- If you change a little, the output cannot change a little.
    - That is not differentiable



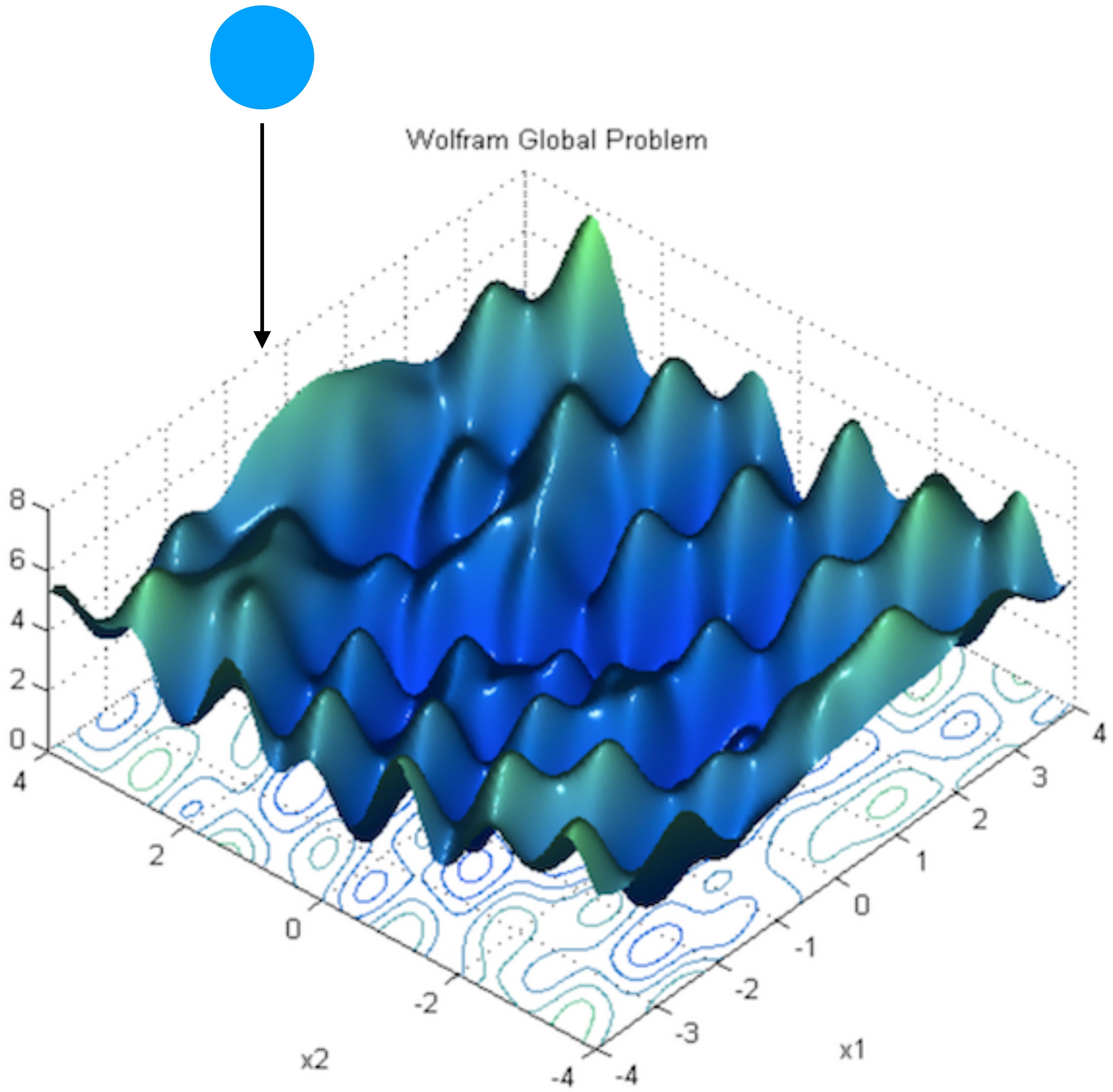
Gradient Descent is Not the Only Way

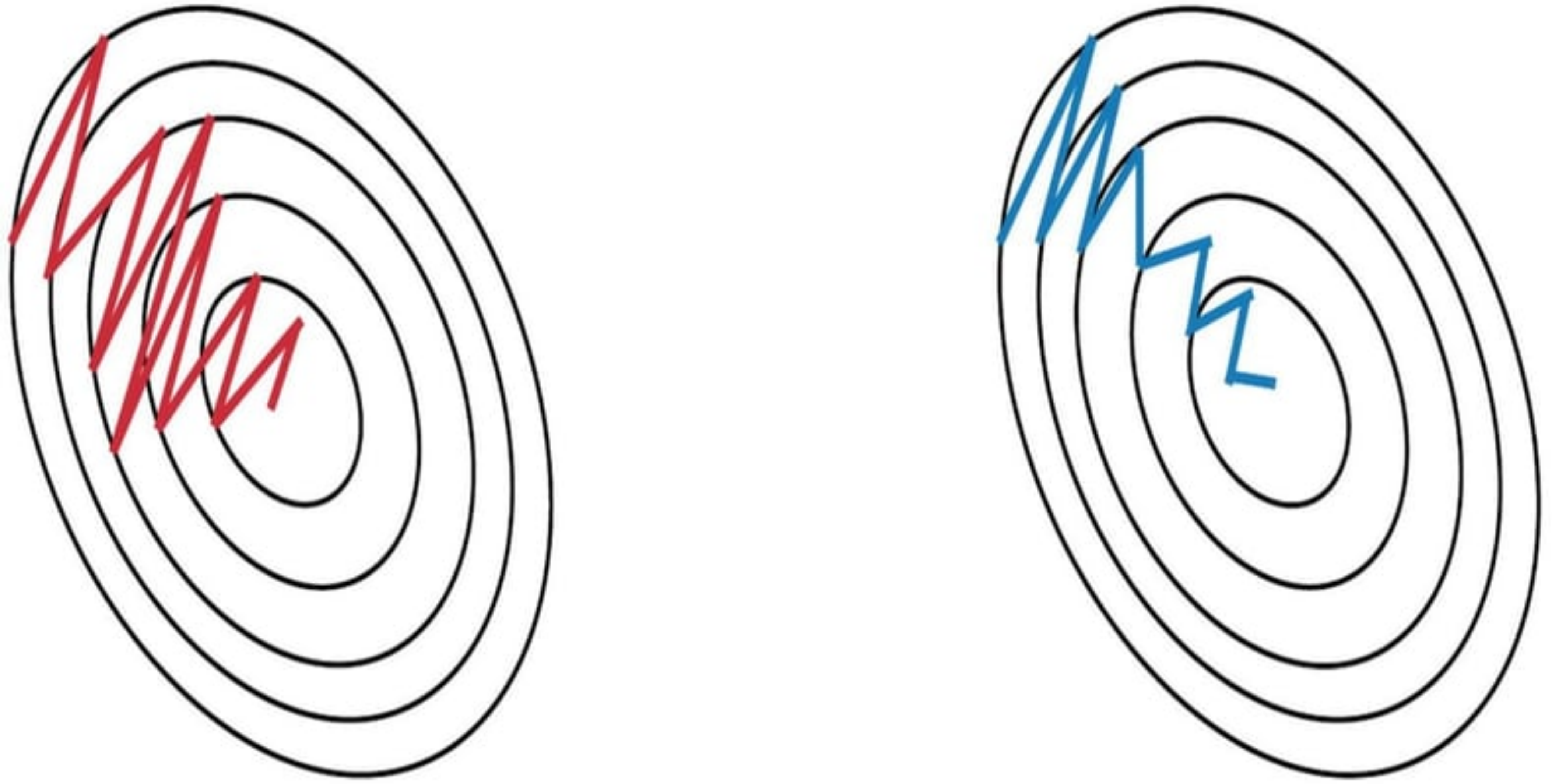
- Reinforcement learning
- Combinatorial optimization and integer programming
- Genetic algorithm
- Best-of-N
- ...

For small scale problems with constraints or not differentiable, consider to use

https://docs.scipy.org/doc/scipy/reference/optimize.html

Loss Surface and Momentum





Stochastic Gradient Descent **withhout** Momentum

Stochastic Gradient Descent **with** Momentum

https://ml4a.github.io/ml4a/how_neural_networks_are_trained/

https://dev.to/nareshnishad/day-25-optimizer-algorithms-for-large-language-models-llms-1p4

Batch Size and Learning Rate

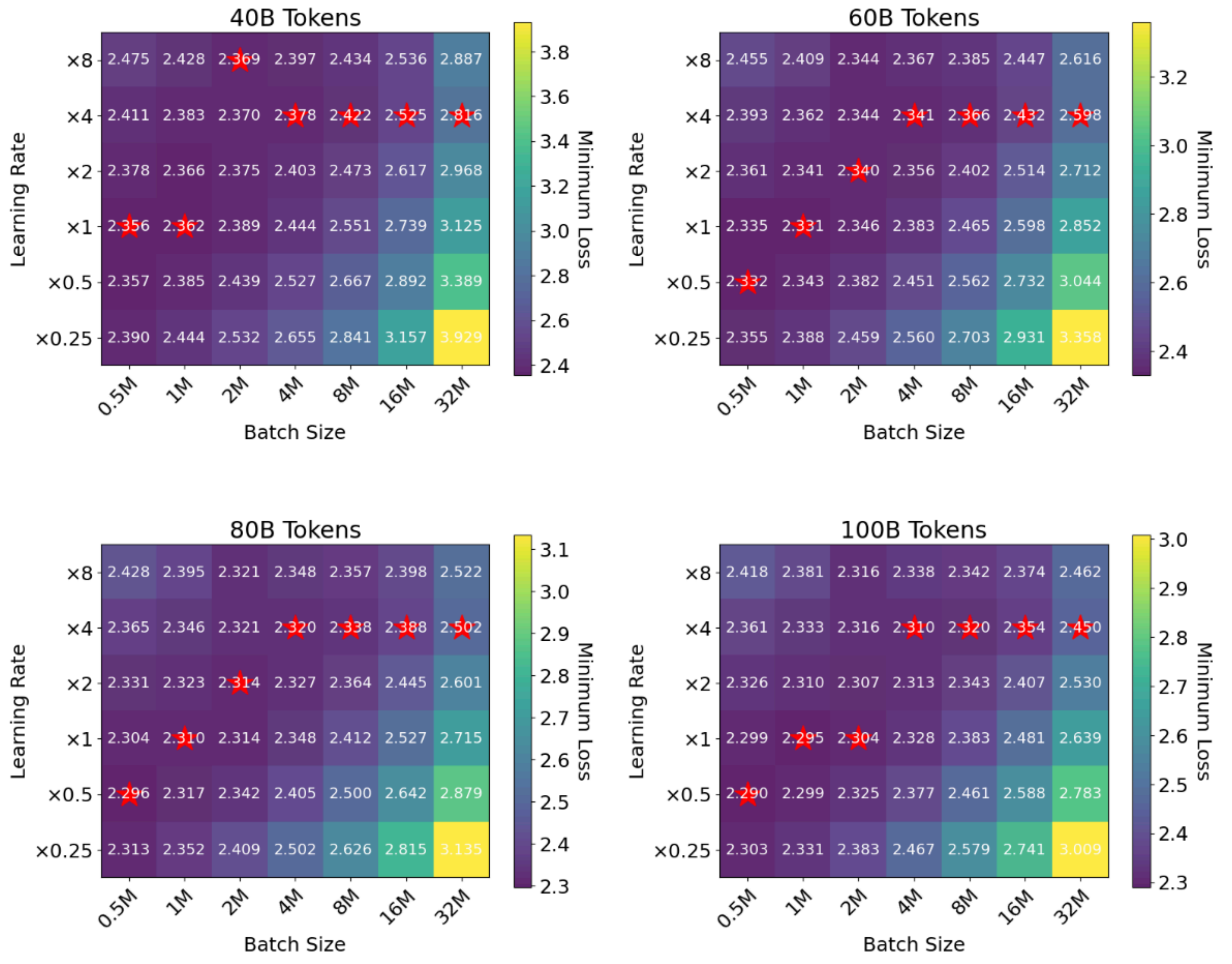


Figure 7: Training losses under different learning rates and batch sizes, with 10B to 100B training tokens. Red stars denote the lowest loss of a certain batch size. The ×1 learning rate means the corresponding one of 350M model in Table 6.

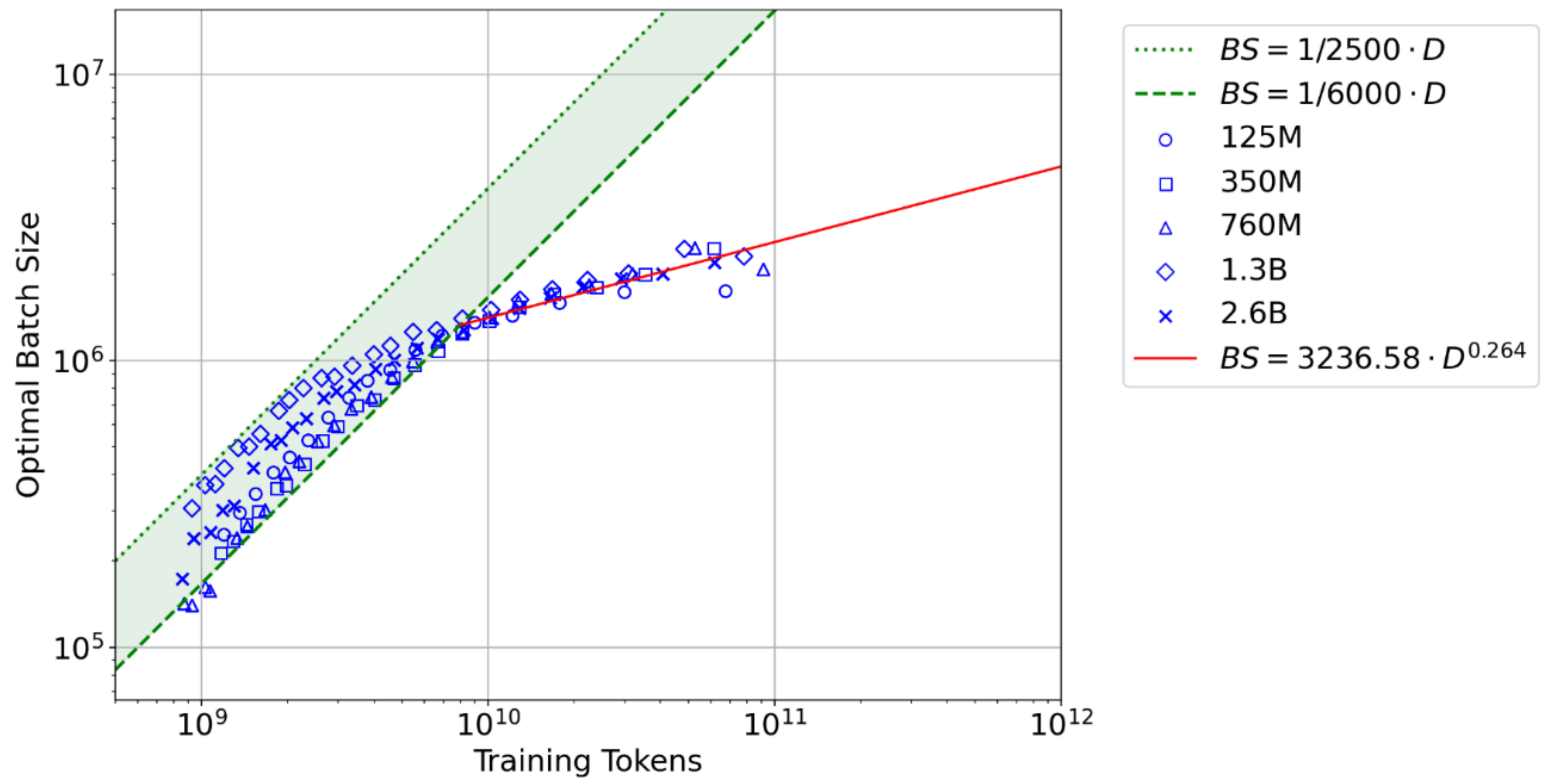


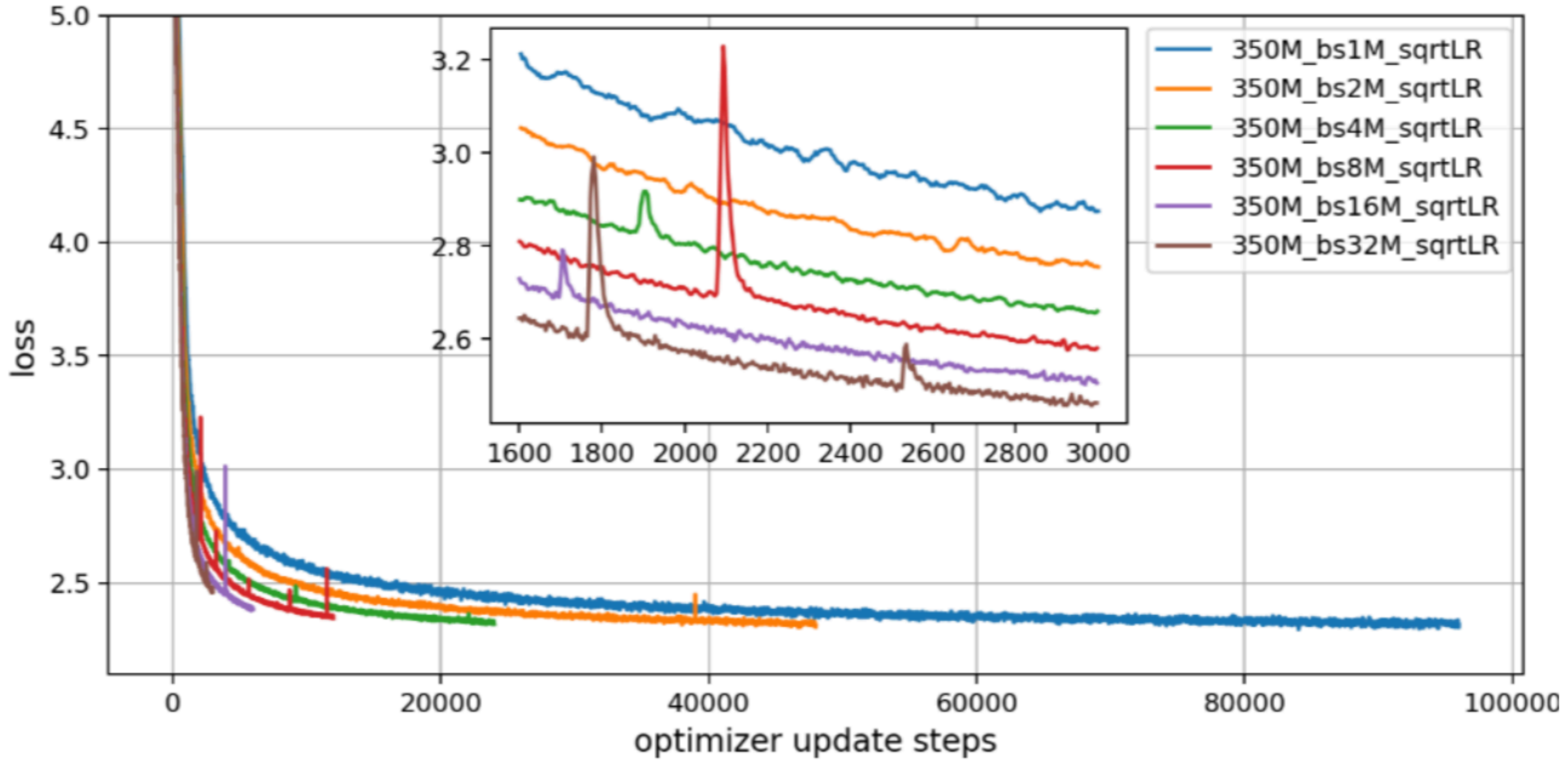
Figure 6: The optimal batch sizes against the available amount of training tokens. The data points mainly fall into two regions: within the green area and outside of it.

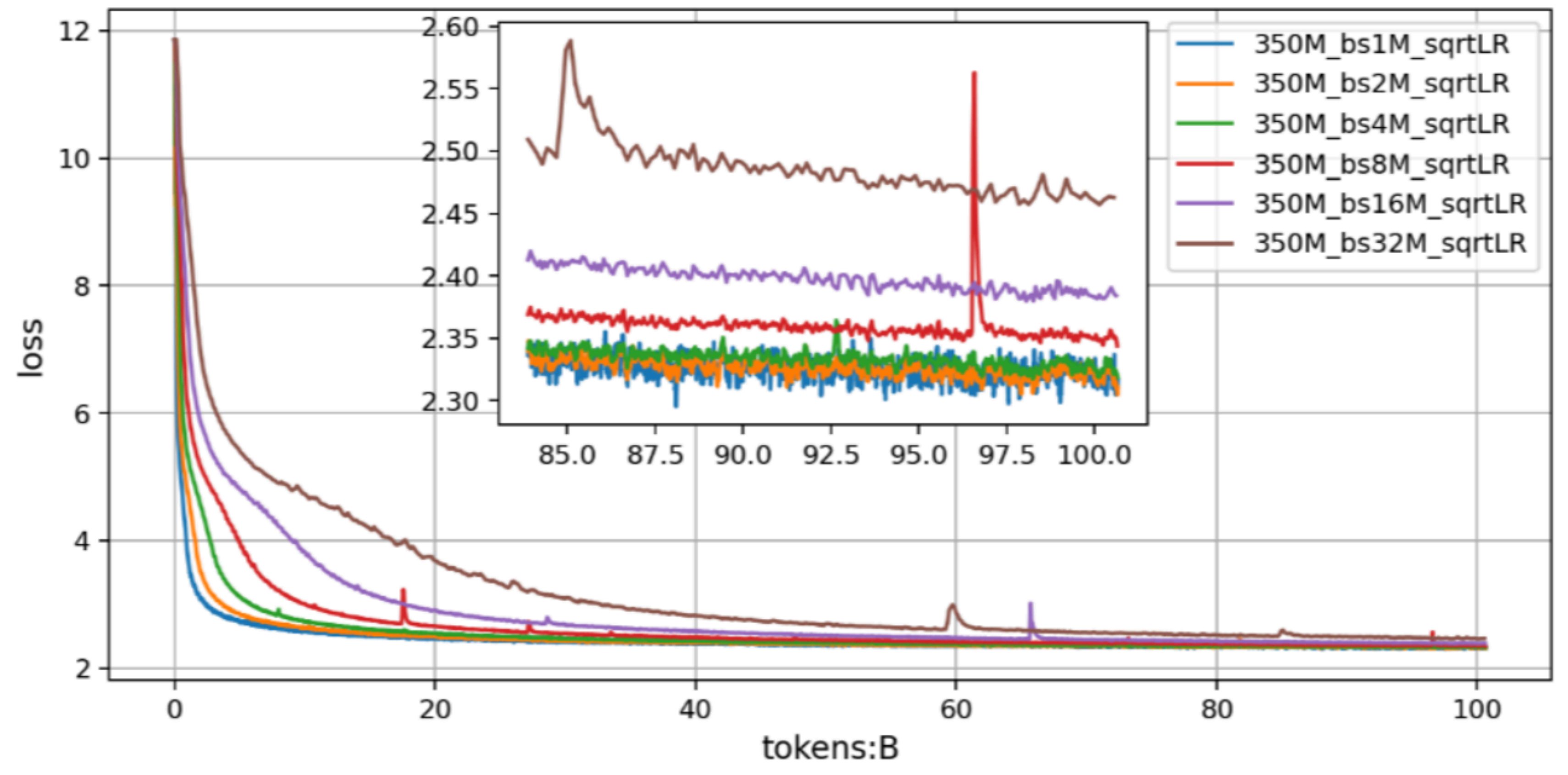
Smaller batch size:

1) Less confident gradient direction -> smaller optimal learning rate,
2) More memorization,
3) Larger training latency

Scaling Law for Language Models Training Considering Batch Size (https://arxiv.org/pdf/2412.01505v1)

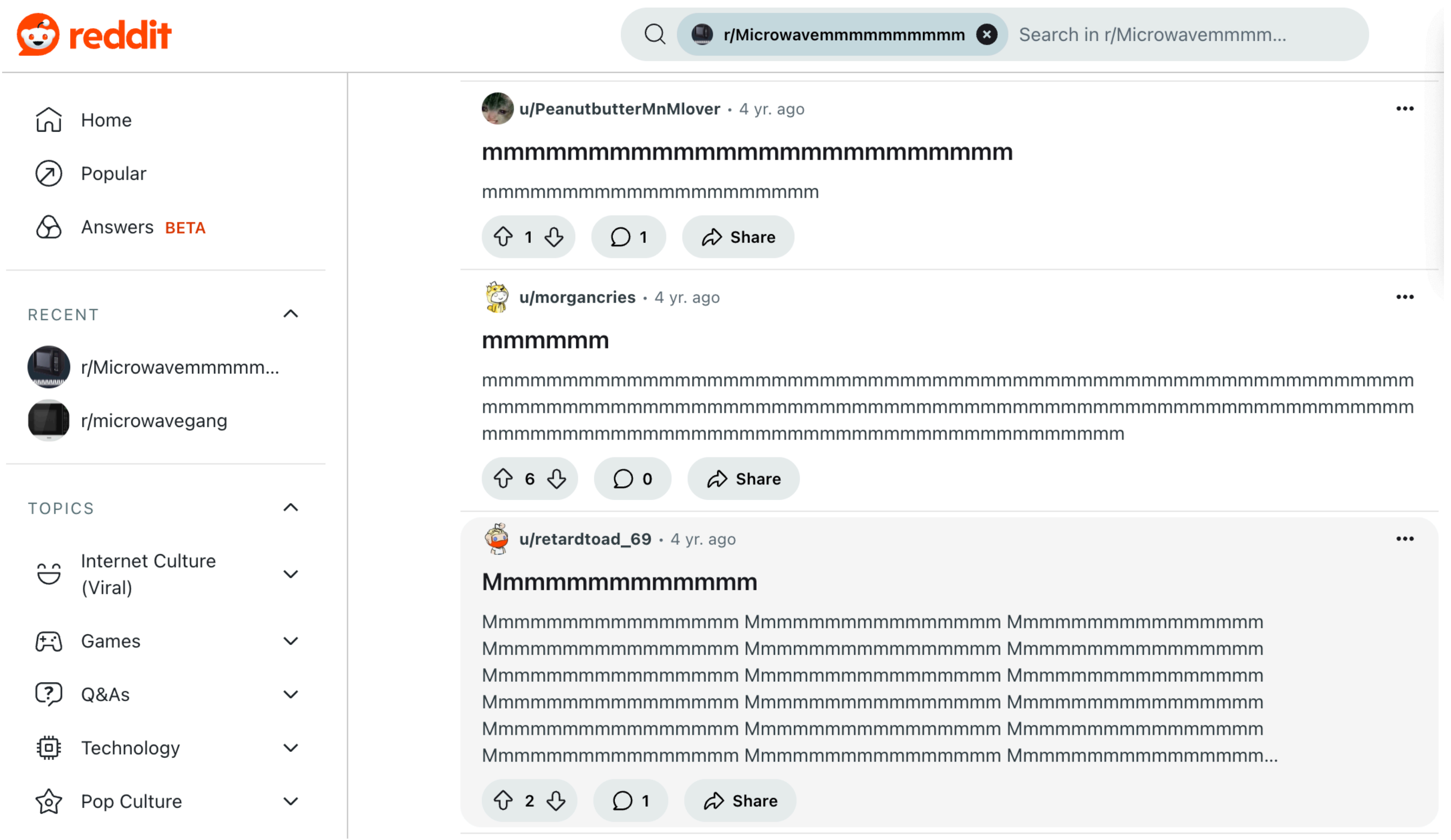
Batch Size and Training Instability





Same number of training tokens -> Smaller batch size is better
Same number of optimization steps -> Larger batch size is better

Larger batch size -> larger learning rate -> training unstable



Some bad batches could wash out everything it learns

Could be caused by a very unlikely sequence (example comes from a talk from Kyle Lo)

Scaling Law for Language Models Training Considering Batch Size (https://arxiv.org/pdf/2412.01505v1)

Less likely Sequence = Higher Learning Rate

$$\theta = \theta - \eta \frac{dL}{d\theta}$$

- Large losses tend to cause large gradients

exp(-100)

-10 $\quad$ 10 $\longrightarrow$ 9

$$L = -\frac{1}{N}\sum_t \log p(w_t \,|\, w_{<t}) = -\frac{1}{N}\sum_t \log \frac{\exp(h^T w_t)}{\sum_w \exp(h^T w)}$$

L ~= 100 $\longrightarrow$ 90

Bad initialization could also cause large gradient at the beginning

Learning Rate and Warmup

- Larger learning rate
  - Faster (and sometimes better)
  - More unstable
- The larger model is more sensitive to the learning rate
  - Probably because the unlikely sequences

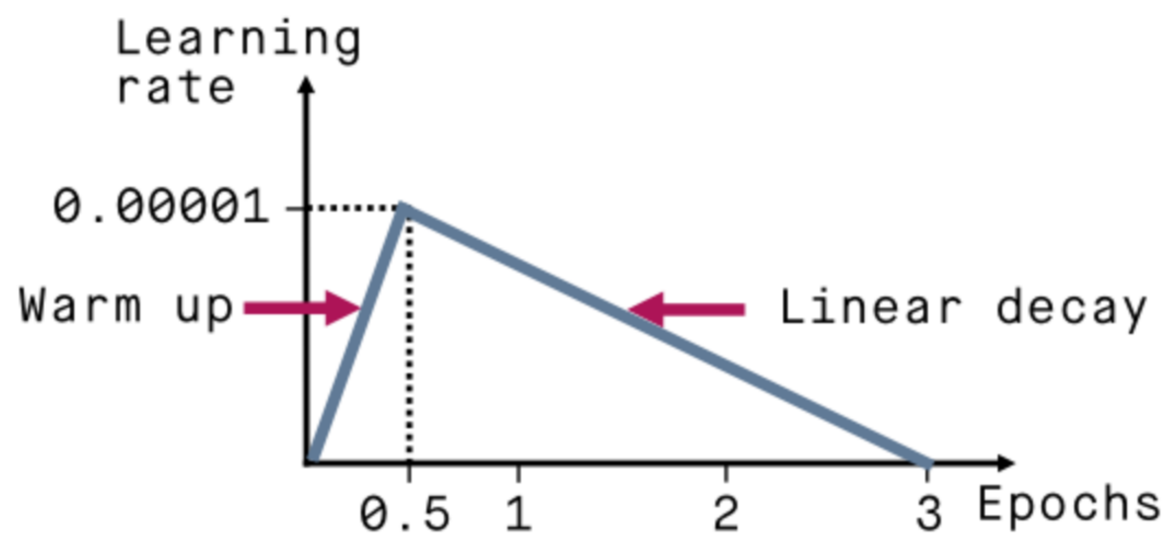
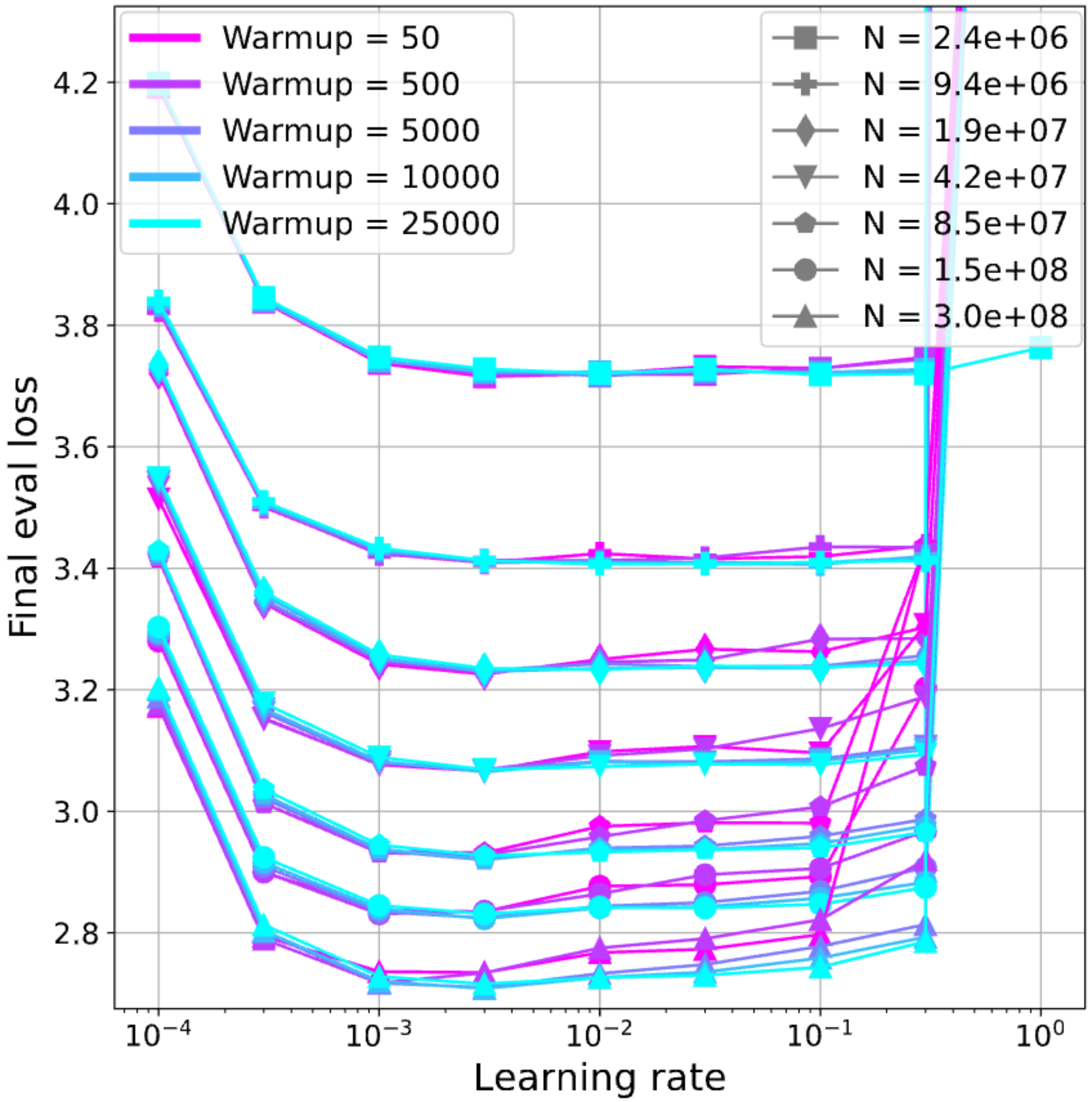


Figure 3. Triangle decay with *Warm-up proportion*=0.5, which means that the learning rate peak at 0.5. *Decrement per epoch*=0.000004. This lets the learning rate go to 0 after 3 epochs.



Small-scale proxies for large-scale Transformer training instabilities (https://arxiv.org/pdf/2309.14322)

How many Epochs?

- One epoch means training the whole dataset once
- In the language modeling task
  - Around 4 epochs is similar to 4 times the data
- If you have a smaller dataset, consider training for more epochs

Scaling Data-Constrained Language Models (https://arxiv.org/pdf/2305.16264)