Optimizer and LR Schedule

Outline

Introduction

Why?

- **Core component** of deep learning
  - Drives the entire training process
  - Determines how models learn from data
  - Critical for model convergence
- **Impact**
  - Training speed
  - Model performance
  - Final accuracy
  - Generalization

Challenges

- Complex loss landscapes
  - Non-convex optimization
  - Multiple local minima and Saddle points
- Training
  - Vanishing/exploding gradients
  - Slow convergence
  - Unstable
  - Overfitting

Optimizer vs Schedulers

Optimizer

- How parameters update
- adapt learning based on gradients

Learninig rate schedule

- Manage learning rate dynamics
- Balance exploration vs exploitation
- Convergence speed
- Final model perormance

Evolution

- Traditional
  - Basic Gradient Descent
  - Batch Gradient Descent
  - Stochastic Gradient Descent (SGD)
  - mini-batch
- Modern
  - Momentum
  - Adaptive learning rates
  - Combined strategies

Optimizer

Common Optimizers Family

- First Generation
  - SGD
  - SGD with momentum
  - Nesterov accelerated gradient

- Adaptive methods
  - AdaGrad
  - RMSprop
  - AdaDelta
  - Adam

- Modern
  - AdamW
  - Lion
  - Lamb
  - ...

Gradient Descent

$$\theta = \theta - \eta * \nabla J(\theta) \tag{1}$$

Where,

- θ: model parameters
- η: learning rate
- $\nabla J(\theta)$: gradient of loss function

Batch Gradient Descent

- Uses entire dataset for each update
- Slow
- High memory
- Deterministic updates

For a dataset with n sample, $i \in [1, n]$:

$$
\begin{aligned}
L &= \left(\frac{1}{n}\right) * \sum L(\theta; x_i, y_i) \\
\theta &= \theta - \eta * \left(\frac{1}{n}\right) * \sum \nabla L(\theta; x_i, y_i)
\end{aligned}
\tag{2}
$$

Batch Gradient Descent Pseudo code

```
for epoch in epochs:
  grads = grad(loss_fn(weights, all_n_samples))
  weights = weights - learning_rate * grads
```

SGD and Mini-batch

- Update parameters for each sample or a m-size batch

$$
\begin{aligned}
L &= \left(\frac{1}{m}\right) * L(\theta; x_i, y_i) \\
\theta &= \theta - \eta * \left(\frac{1}{m}\right) * \nabla L(\theta; x_i, y_i)
\end{aligned}
\tag{3}
$$

```
for epoch in epochs:
  for batch in get_batches(dataset, size=m):
    grads = grad(loss_fn(weights, batch))
    weights = weights - learning_rate * grads
```

AdaGrad (Adaptive Gradient)

- Adaptive learning rate for each parameter
- Larger updates for infrequent parameters
- Smaller updates for frequent parameters

$$
\begin{aligned}
r_t &= r_{t-1} + \nabla J(\theta_{t-1})^2 \\
\theta_t &= \theta_{t-1} - \left(\frac{\eta}{\sqrt{r_t + \varepsilon}}\right) * \nabla J(\theta_{t-1})
\end{aligned} \tag{4}
$$

AdaGrad (Adaptive Gradient)

- Adaptive learning rate for each parameter
- Larger updates for infrequent parameters
- Smaller updates for frequent parameters

$$
\begin{aligned}
r_t &= r_{t-1} + \nabla J(\theta_{t-1})^2 \\
\theta_t &= \theta_{t-1} - \left(\frac{\eta}{\sqrt{r_t + \varepsilon}}\right) * \nabla J(\theta_{t-1})
\end{aligned}
\tag{4}
$$

- Learning rate will decrease over time

Momentum

- Inpire by physics momentum
- Accumulate gradients history
- Helps overcome local minima
- Reduce training oscillations

$$
\begin{aligned}
v_t &= \beta v_{t-1} + (1-\beta)\nabla J(\theta_{t-1}) \\
\theta_t &= \theta_{t-1} - \eta v
\end{aligned} \tag{5}
$$

- v: velocity (momentum)
- β: momentum coefficient (0.9 in practice)

```
Without momentum (oscillating):

  /\    /\/\/\
 /  \/\/
/

With momentum (smooth):

  /^^^^^
 /
/
```

Weight Decay

- Prevents model overfitting
- Penalize large weights
- Encourage simpler models
- Reduces model's dependency on single features

Weight Decay

L2 norm in the loss function

$$L = L + \left(\frac{\lambda}{2}\right) \|\theta\|^2 \tag{6}$$

Standard Weight Decay in optimizer

$$\theta_t = \theta_{t-1} - \eta \nabla J(\theta_{t-1}) - \lambda \theta_{t-1} \tag{7}$$

Weight Decay pseudo code

```
for params, grad in zip(params, grads):
  param = param - lr * (grad + wd * param)
  # or
  param = param - lr * grad - lr * wd * param
```

Some Typical values for λ

Dataset Size	λ Value
Small ($<$ 10k sample)	1e-3
Medium (10k - 1M)	1e-4
Large ($>$ 1M)	1e-5

Architecture Type	λ Adjustment
CNN	Base λ
Transformer	$0.1 \times$ Base λ
ResNet	$0.5 \times$ Base λ

Some Typical values for λ

Data Type	**λ Adjustment**
Simple/Linear	$2 \times$ Base λ
Complex/Nonlinear	$0.5 \times$ Base λ

Training Length	**λ Adjustment**
Short (< 20 epochs)	$0.5 \times$ Base λ
Long (> 100 epochs)	$2 \times$ Base λ

SGD with Momentum

Add momentum term to vanilla SGD

$$\begin{aligned} v_t &= \beta * v_{t-1} + \nabla J(\theta_{t-1}) \\ \theta_t &= \theta_{t-1} - \eta * v_t \end{aligned} \tag{8}$$

Nesterov Accelerated Gradient

- Momentum: Current → Gradient → Momentum → Update
- Nesterov: Current → Look-ahead → Gradient → Momentum → Update

$$\begin{aligned} v_t &= \beta * v_{t-1} + \nabla J(\theta_{t-1} + \beta * v_{t-1}) \\ \theta_t &= \theta_{t-1} - \eta * v_t \end{aligned} \tag{9}$$

RMSprop

- Improves AdaGrads' declining learning rate with exponential moving average (EMA)

Before

$$\begin{aligned} r_t &= r_{t-1} + \nabla J(\theta_{t-1})^2 \\ \theta_t &= \theta_{t-1} - \left(\frac{\eta}{\sqrt{r_t + \varepsilon}}\right) * \nabla J(\theta_{t-1}) \end{aligned} \tag{10}$$

After

$$\begin{aligned} r_t &= \beta * r_{t-1} + (1-\beta) * \nabla J(\theta_{t-1})^2 \\ \theta_t &= \theta_{t-1} - \left(\frac{\eta}{\sqrt{r_t + \varepsilon}}\right) * \nabla J(\theta_{t-1}) \end{aligned} \tag{11}$$

Adam

- Adds momentum to RMSprop
- Maintains moving averages of gradients (momentum) and squared gradients (LR)

$$\begin{aligned} m_t &= \beta_1 * m_{t-1} + (1 - \beta_1) * \nabla J(\theta_{t-1}) \\ v_t &= \beta_2 * v_{t-1} + (1 - \beta_2) * \nabla J(\theta_{t-1})^2 \\ m_t &= \frac{m_t}{1 - \beta_1^t} \\ v_t &= \frac{v_t}{1 - \beta_2^t} \\ \theta_t &= \theta_{t-1} - \left(\frac{\eta}{\sqrt{v_t + \varepsilon}} \right) * m_t \end{aligned} \tag{12}$$

AdamW

- Add weight decay to Adam

$$\theta_t = \theta_{t-1} - \left(\frac{\eta}{\sqrt{v_t + \varepsilon}} \right) * m_t - \lambda * \theta_{t-1} \tag{13}$$

Lion (Google in 2023)

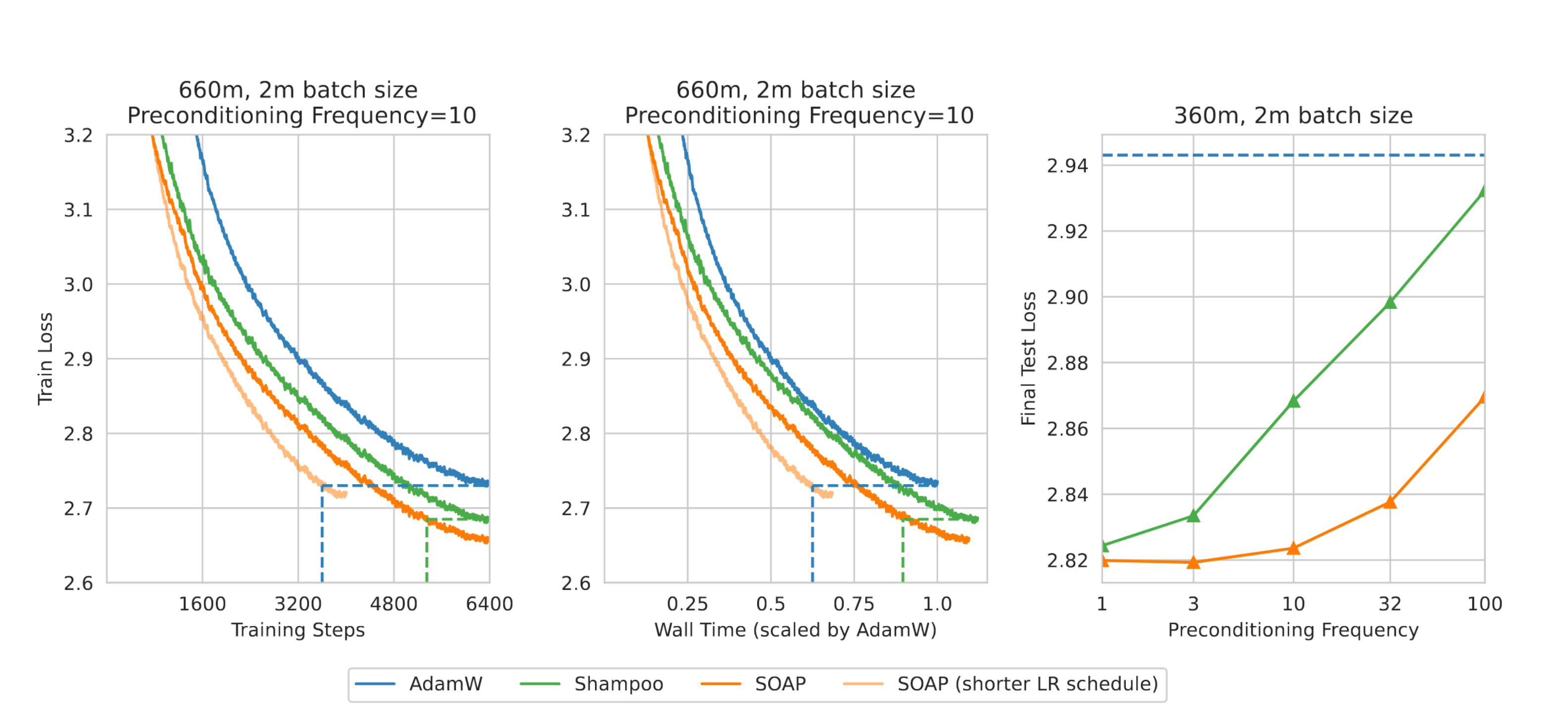
- Use sign gradient instead of raw gradient
- Reduce memory usage

$$
\begin{aligned}
m_t &= \beta_1 * m_{t-1} + (1 - \beta_1) * \operatorname{sign}(\nabla J(\theta_{t-1})) \\
\theta_t &= \theta_{t-1} - \eta * \operatorname{sign}(m_t)
\end{aligned} \tag{14}
$$

1. Chen, X. *et al.* Symbolic Discovery of Optimization Algorithms. (2023) doi:10.48550/arXiv.2302.06675

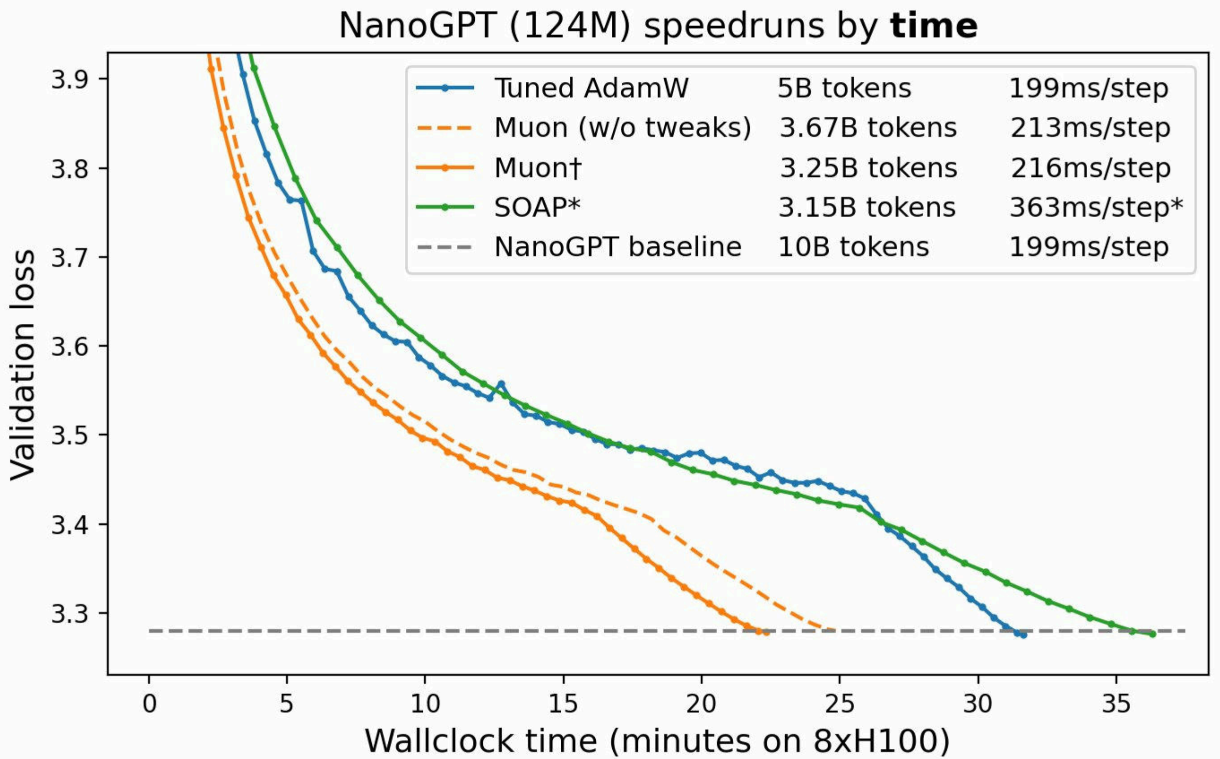
SOTA Optimizers

SOAP[2]



2. Vyas, N. *et al.* SOAP: Improving and Stabilizing Shampoo Using Adam. (2024) doi:10.48550/arXiv.2409.11321

Muon[3]



*SOAP is in active development and is not yet optimized for distributed training. Future versions will significantly reduce the wallclock overhead.

†Also sometimes known as Keller's meme optimizer

3. Jordan, K. KellerJordan/Modded-Nanogpt. (2024)

Schedule

Why?

Problems with fixed LR

- Large: training unstable
- Small: slow convergence

Common Schedules

- Step Decay
- Cosine Annealing
- Warm-up
- One Cycle
- Reduce on Plateau
- …

Step Decay

$$\eta = \eta_{\text{init}} \gamma^{\left\lfloor \frac{\text{epoch}}{\text{step\_size}} \right\rfloor} \tag{15}$$

Where γ is the decay factor and **step_size** is the number of epochs to decay.

```
lr = [
  0.1,  # epoch 0-30
  0.01,  # epoch 30-60
  0.001,  # epoch 60-90
  ...
]
```

Cosine Annealing

$$\eta_t = \eta_{\min} + \left(\frac{1}{2}\right) * (\eta_{\max} - \eta_{\min}) * \left(1 + \cos\left(\frac{T_{\text{cur}}}{T_{\max}}\pi\right)\right) \qquad (16)$$

Where t is the current epoch, T_{cur} is the current step, and $T_{\max}$ is the total number of steps.

4. Loshchilov, I. & Hutter, F. SGDR: Stochastic Gradient Descent with Warm Restarts. (2017) doi:10.48550/arXiv.1608.03983

Warm-up Strategy

- Stabilizes early training
- Prevents early divergence

```
if step < warmup_steps:
  lr = warmup_schedule(step)
else:
  lr = normal_schedule(step)
```

One Cycle

1. Linearly increase LR to max lr (warmup)
2. Linearly decrease LR to min lr
    - Optionally, use other annealing (like cosine) to decrease LR further

```
if step < warmup_steps:
  lr = linear_increase(step, lr)
else:
  lr = cosine(step - warmup_steps, total_steps - warmup_steps)
```

Reduce on Plateau

Adaptize Learning Rate Strategy

```python
def update(current_metric):
  # Check if we should change the learning rate or not by comparing the metrics
  is_better = compare(current_metric, best_metric)

  # Change the best metric
  if is_better:
    best_metric = current_metric
    num_bad_epochs = 0
  else:
    num_bad_epochs += 1

  # Change the learning rate if necessary
  if num_bad_epochs >= patience:
    lr = max(lr * factor, min_lr)
    num_bad_epochs = 0
```

Hyperparameter suggestion in LR Schedules

Step Decay

Parameter	Common Values
step_size	2000-4000 steps
gamma	0.1-0.5

Warm-up

Model Type	Warmup Steps
CNN	5% steps
Transformer	10% steps
Fine-tuning	6% steps
Large batch size > 1000	15% - 20%

Model Type	Warmup Steps
Transfer learning	3-5%
Small datasets	3-5%
Unstable training	Increase 5%

Learning Rate Range Suggestions

LR Range

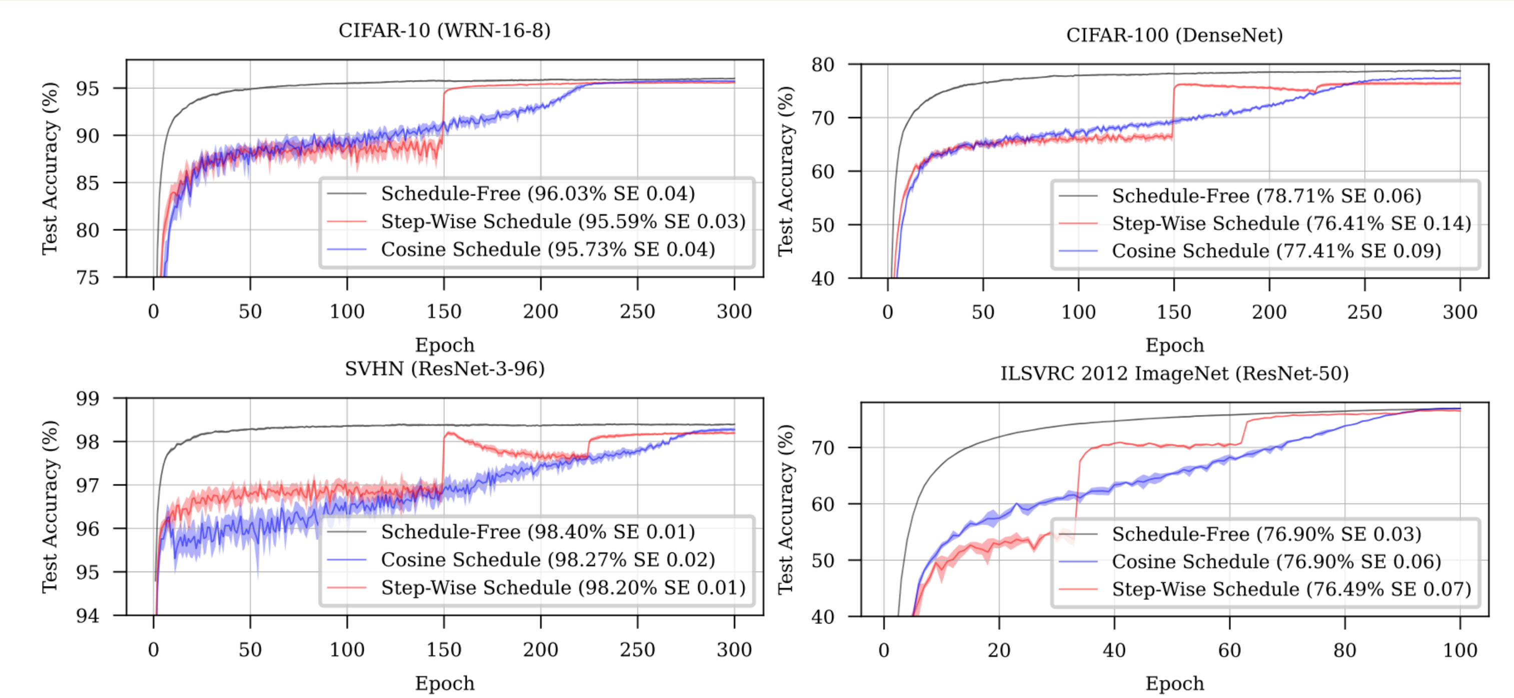
Optimizer	Learning Rate Range
SGD (no momentum)	0.1 - 1.0
SGD with Momentum	0.01 - 0.1
Adam/AdamW	1e-4 - 1e-3
RMSprop	1e-4 - 1e-3
AdaGrad	0.01 - 0.1
Lion	1e-4 - 3e-4

Learning Rate Range Suggestions

Adjustments

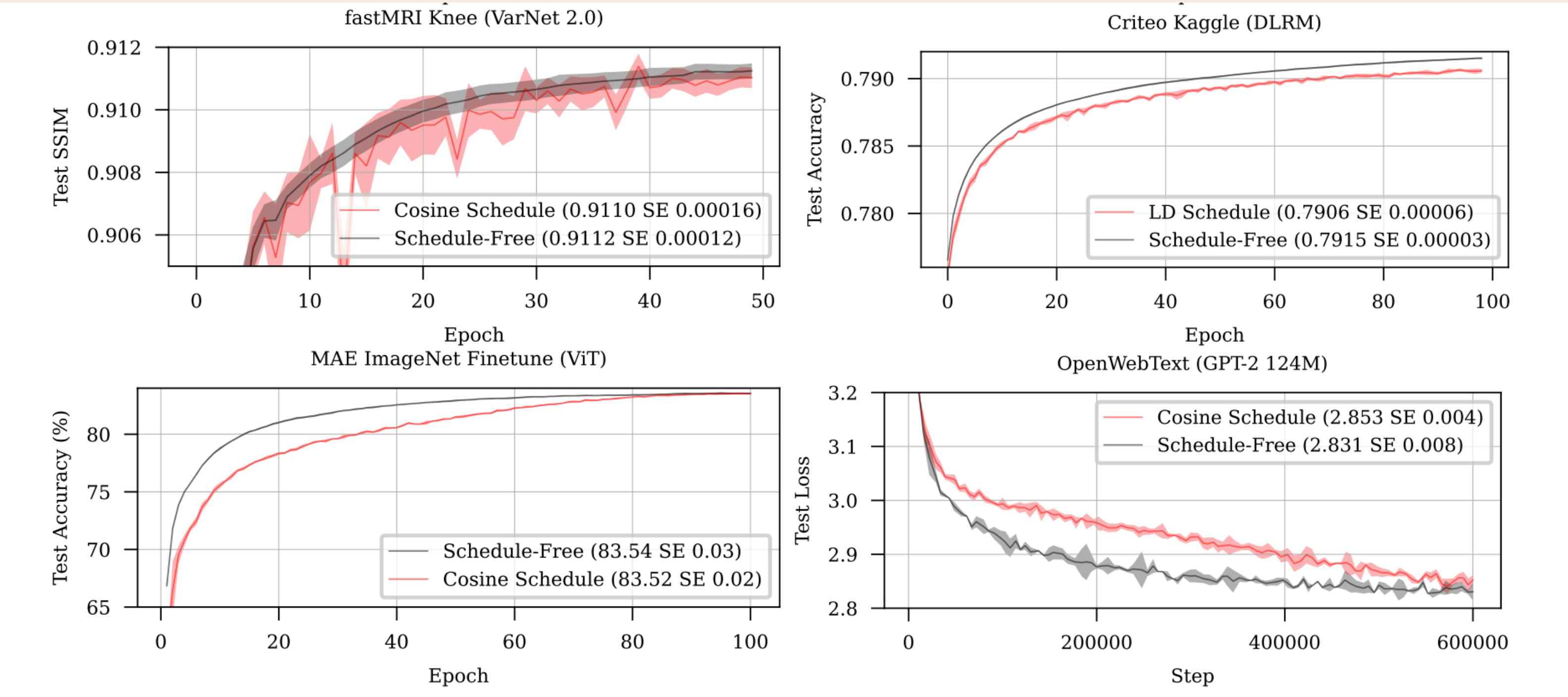
Scenario	LR Adjustment
Batch size doubled	$\text{LR} \times \sqrt{2}$
Deeper network	$\text{LR} \times 0.5$
Fine-tuning	$\text{LR} \times 0.1$
Unstable training	$\text{LR} \times 0.1$

SOTA Schedule The Road Less Scheduled[5]



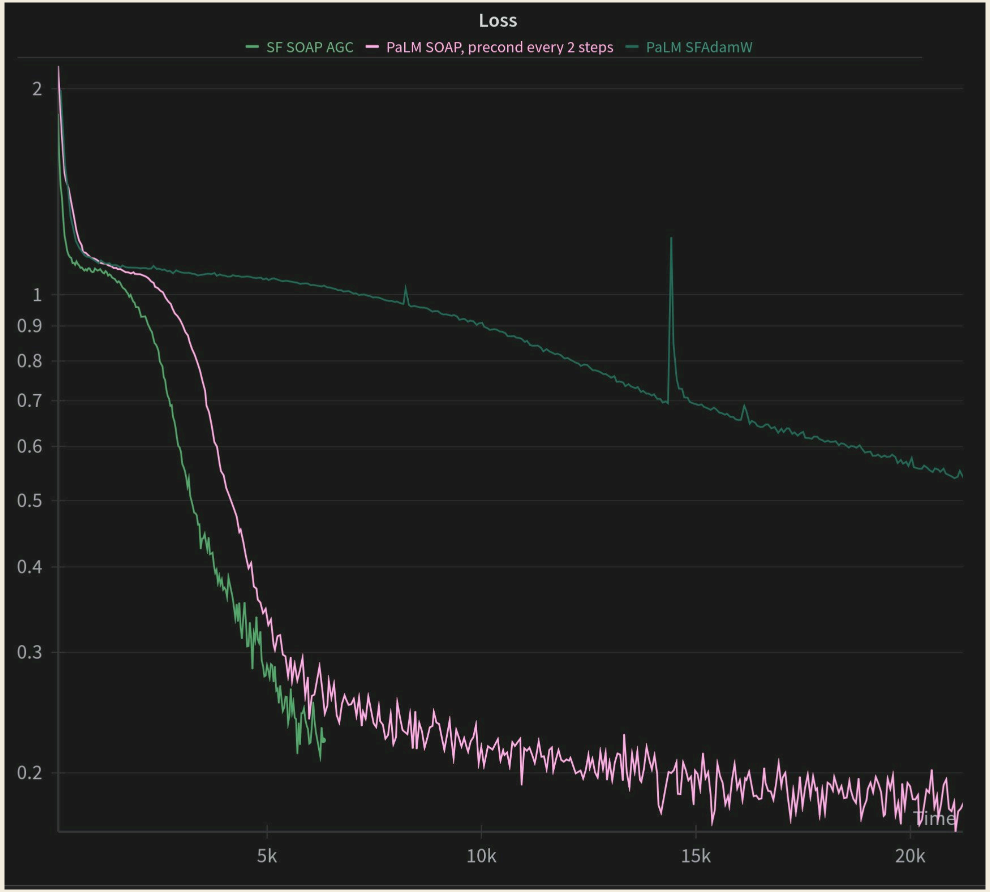
5. Defazio, A. *et al.* The Road Less Scheduled. (2024) doi:10.48550/arXiv.2405.15682

SOTA Schedule The Road Less Scheduled[5]



5. Defazio, A. *et al.* The Road Less Scheduled. (2024) doi:10.48550/arXiv.2405.15682

SOAP & Schedule Free



Reference

1. Chen, X. *et al.* Symbolic Discovery of Optimization Algorithms. (2023) doi:10.48550/arXiv.2302.06675
2. Vyas, N. *et al.* SOAP: Improving and Stabilizing Shampoo Using Adam. (2024) doi:10.48550/arXiv.2409.11321
3. Jordan, K. KellerJordan/Modded-Nanogpt. (2024)
4. Loshchilov, I. & Hutter, F. SGDR: Stochastic Gradient Descent with Warm Restarts. (2017) doi:10.48550/arXiv.1608.03983
5. Defazio, A. *et al.* The Road Less Scheduled. (2024) doi:10.48550/arXiv.2405.15682