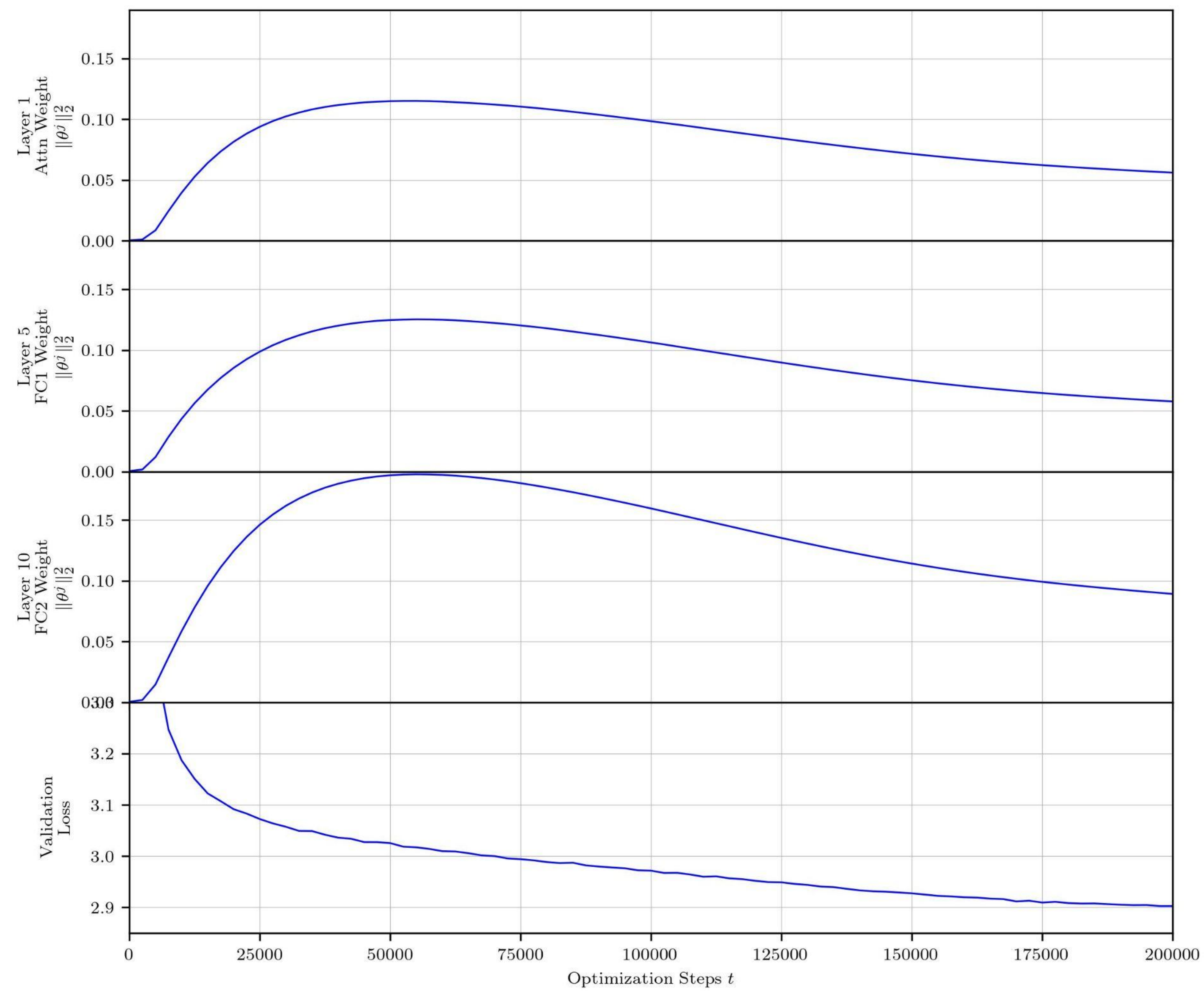
NEURAL INFORMATION PROCESSING SYSTEMS

Improving Deep Learning Optimization through Constrained Parameter Regularization

Jörg K.H. Franke[1], Michael Hefenbrock[2], Gregor Köhler[3], Frank Hutter[1,4]

[1]University of Freiburg, Germany
[2]RevoAI, Karlsruhe, Germany
[3]German Cancer Research Center (DKFZ), Heidelberg, Germany
[4]ELLIS Institute Tübingen, Germany

L2 norm in GPT2s/OWT training with AdamW



Reformulating Regularization

Optimization objective with L_2-regularization:

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \; L(\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{y}) + \gamma \cdot R(\boldsymbol{\theta})$$

(parameters $\boldsymbol{\theta}$, input data $\boldsymbol{X}$, target $\boldsymbol{y}$, regularization term R, strength γ)

Regularization as an inequality-constrained optimization problem:

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \; L(\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{y}) \quad \text{s.t.} \quad c_j\big(\boldsymbol{\theta}^j\big) = R\big(\boldsymbol{\theta}^j\big) - \kappa^j \leq 0, \quad \text{for} \quad j = 1, \cdots, J,$$

(constraint c, upper bound for R κ^j, parameter $\boldsymbol{\theta}^j$)

The augmented Lagrangian method

Our inequality-constrained optimization problem

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ L(\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{y}) \quad \text{s.t.} \quad R\big(\boldsymbol{\theta}^j\big) - \kappa^j \leq 0, \quad \text{for} \quad j = 1, \cdots, J$$

is addressed by an augmented Lagrangian update

$$\lambda^j_{t+1} \leftarrow (\lambda^j_t + \mu \cdot (R\big(\boldsymbol{\theta}^j\big) - \kappa^j))^+ \quad \text{for} \quad j = 1, \cdots, J.$$

$$\boldsymbol{\theta}^j_{t+1} \leftarrow \text{Opt}(\boldsymbol{\theta}^j_t, \boldsymbol{X}, \boldsymbol{y}) + \lambda^j_{t+1} \cdot \nabla_{\boldsymbol{\theta}}(R\big(\boldsymbol{\theta}^j\big) - \kappa^j)$$

(Lagrange multipliers λ^j, Upper bounds κ^j, update rate μ ($\mu = 1$))

AdamCPR

Our inequality-constrained optimization problem

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\ L(\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{y}) \quad \text{s.t.} \quad R(\boldsymbol{\theta}^j) - \kappa^j \leq 0, \quad \text{for} \quad j = 1, \cdots, J$$

is addressed by an augmented Lagrangian update

$$\lambda_{t+1}^j \leftarrow (\lambda_t^j + \mu \cdot (R(\boldsymbol{\theta}^j) - \kappa^j))^+ \quad \text{for} \quad j = 1, \cdots, J.$$

$$\boldsymbol{\theta}_{t+1}^j \leftarrow \text{Opt}(\boldsymbol{\theta}_t^j, \boldsymbol{X}, \boldsymbol{y}) + \lambda_{t+1}^j \cdot \nabla_{\boldsymbol{\theta}}(R(\boldsymbol{\theta}^j) - \kappa^j)$$

(Lagrange multipliers λ^j, Upper bounds κ^j, update rate μ ($\mu = 1$))

Algorithm 1 Optimization with constrained parameter regularization (CPR).

Require: Loss Function $L(\boldsymbol{\theta}, \boldsymbol{X}, \boldsymbol{y})$ with parameters $\boldsymbol{\theta}$, and data $\mathcal{D} = \{(\boldsymbol{X}_n, \boldsymbol{y}_n)\}_{n=0}^N$
Require: Hyperparameters: Learning rate $\eta \in \mathbb{R}^+$, Lagrange multiplier update rate $\mu \in \mathbb{R}^+ (= 1.0)$
Require: Optimizer $\text{Opt}(\cdot)$ for minimization, Regularization function $R(\boldsymbol{\theta})$ (e.g. L2-norm)

```
1: λ_t^j ← 0 for j = 1, ··· , J
2: κ^j ← Initialize(θ_0^j) for j = 1, ··· , J        ▷ Initializing the upper bound κ, see Section 4.3
3: for X_t, y_t ∼ D do
4:     θ_{t+1} ← θ_t + Opt(L(θ_t, X_t, y_t), η)       ▷ Classic parameter update using, e.g., Adam.
5:     for each regularized parameter group θ_t^j in θ_t do
6:         λ_{t+1}^j ← (λ_t^j + μ · (R(θ_t^j) − κ^j))^+
7:         θ_{t+1}^j ← θ_{t+1}^j − ∇_{θ^j} R(θ_t^j) · λ_{t+1}^j
8:     end for
9: end for
```

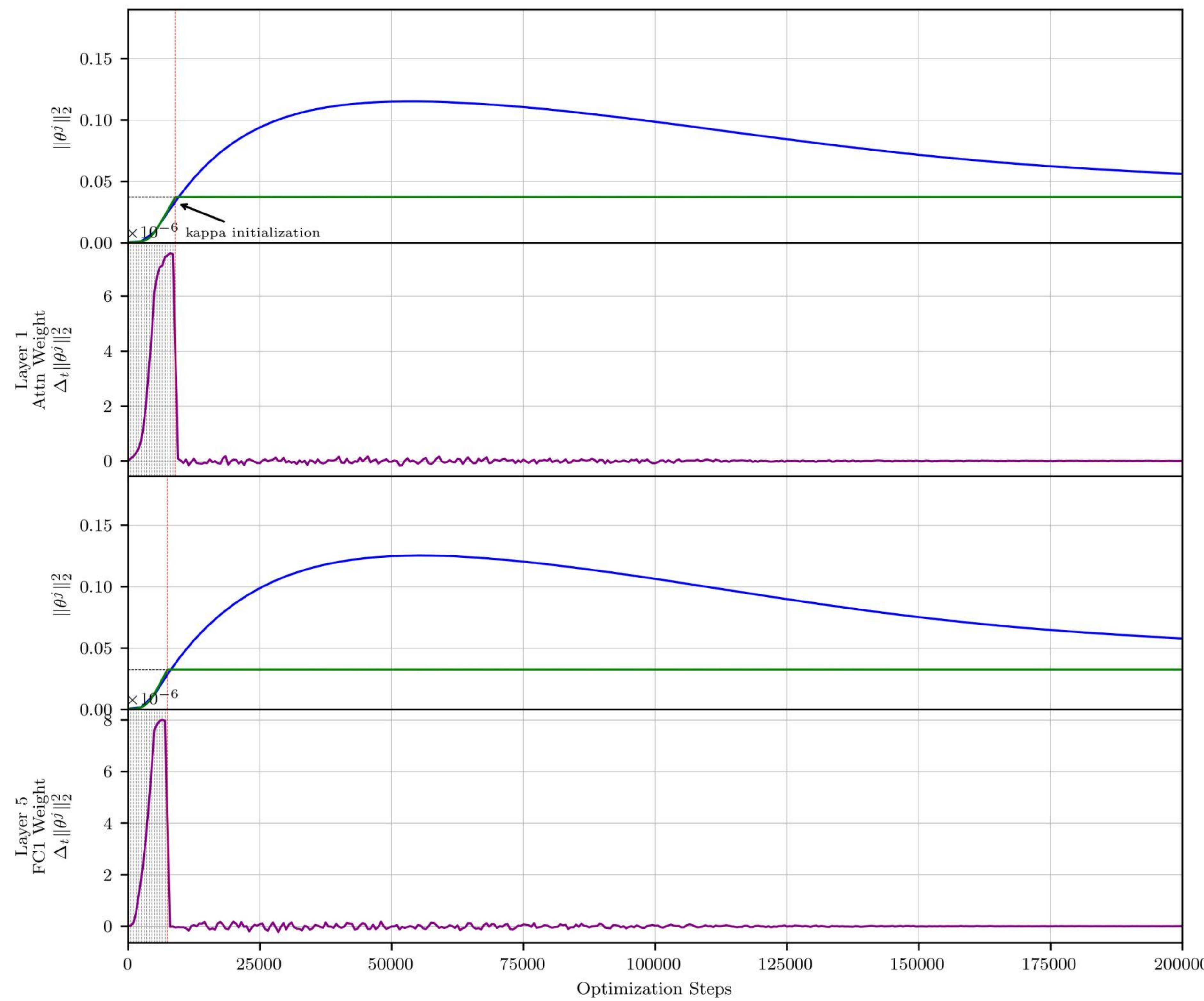
Initialization techniques for the Upper Bound

- Kappa-K
  - Set the upper bound to the same value for all parameter matrices.
  - $\kappa^j \leftarrow \kappa$
  - Hyperparameter: global upper bound

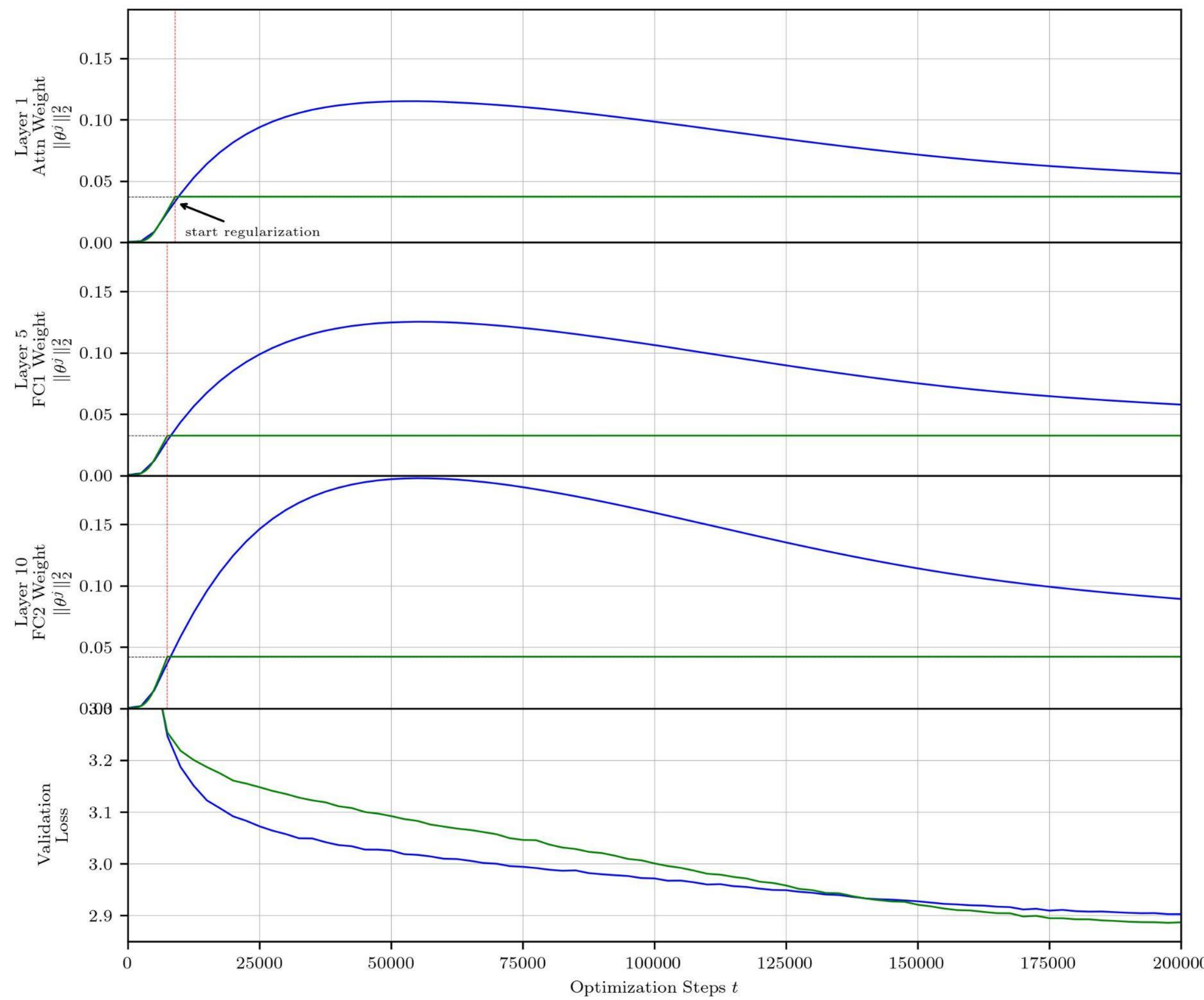
- Kappa-WS
  - Train the model for a specific number of warm start (WS) steps and then set the upper bound.
  - $\kappa^j \leftarrow k \cdot R(\boldsymbol{\theta}^j_{t=0})$, with $k \in \mathbb{R}^+$
  - Hyperparameter: warm start steps

- Kappa-IP (inflection point)
  - Use the first inflection point of the regularization to warm start each upper bound individually.
  - $\kappa^j \leftarrow R(\boldsymbol{\theta}^j_{t=i})$ where i is the first iteration where $\Delta_t \Delta_t R(\boldsymbol{\theta}^j) < 0$
  - Hyperparameter: -

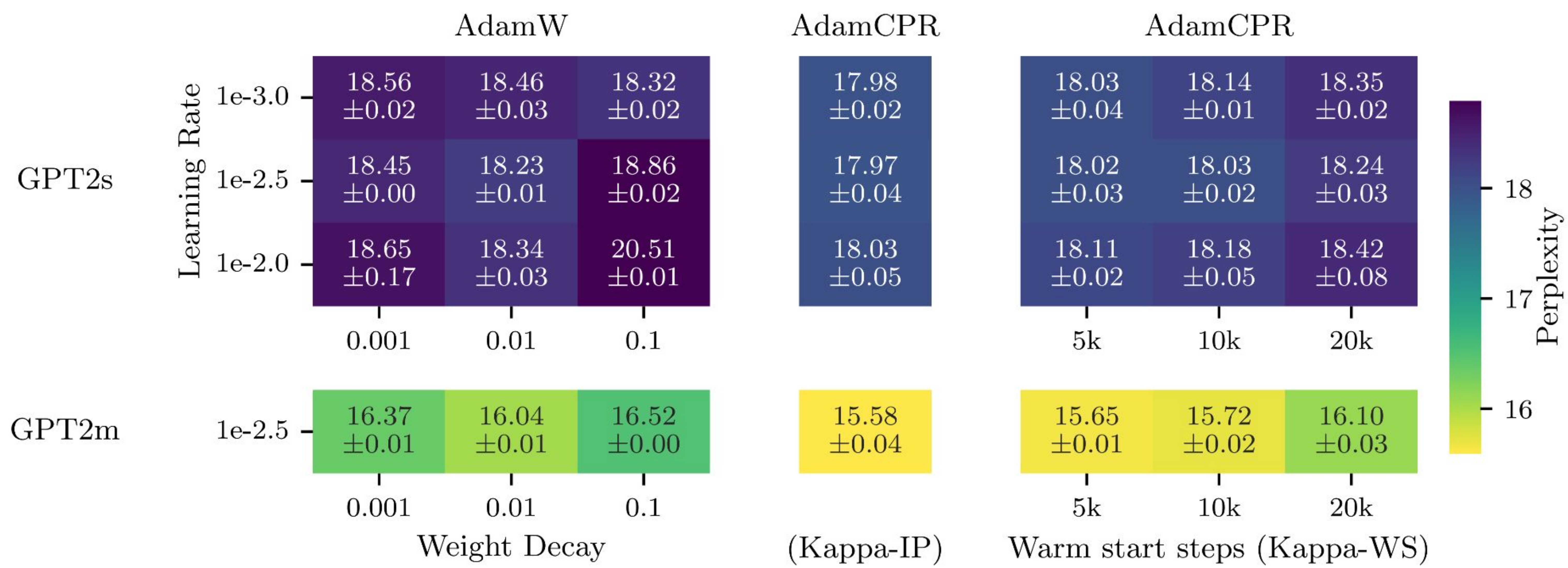
Set upper bound on inflection point



Constrained parameter regularization

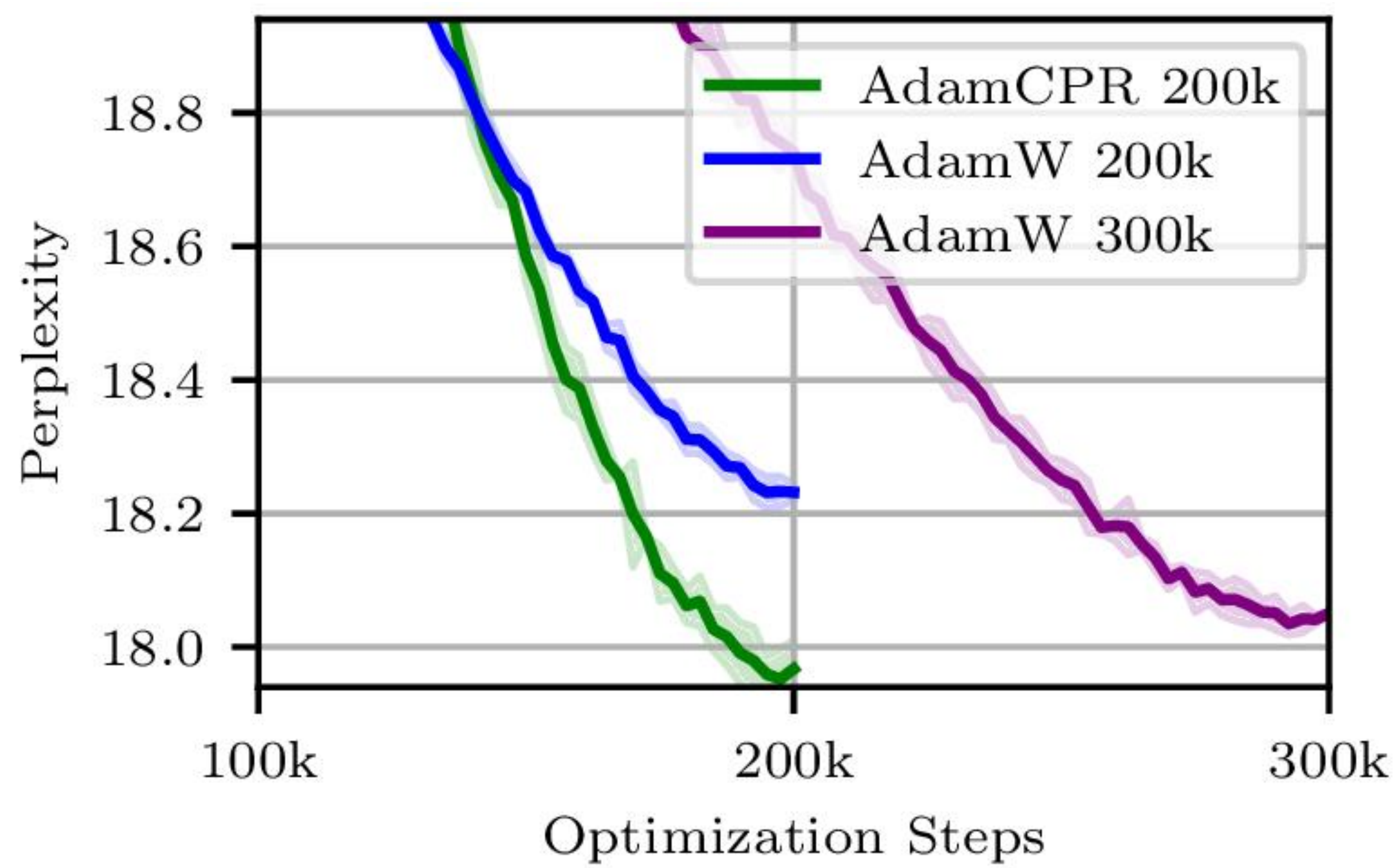


GPT2/OWT pretraining



Perplexity (↓) ± std across three random seeds of GPT2s and GPT2m trained on OpenWebText with AdamW (left) and AdamCPR with Kappa-IP (middle) and AdamCPR with Kappa-WS (right).
We use a learning rate warm-up of 5k steps.
The CPR with the hyperparameter-free strategy Kappa-IP outperforms weight decay.

GPT2/OWT pretraining



DeiT/ImageNet pretraining

ImageNet Pretraining		AdamW			AdamCPR			
		weight decay			Kappa WS (x lr-warmup)			Kappa IP
		0.005	0.05	0.5	1x	2x	4x	
DeiT-Small (22M)	Top-1 Acc. (%)	76.97	79.03	79.16	**79.81**	79.33	78.04	**79.84**
DeiT-Base (86M)	Top-1 Acc. (%)	76.19	78.59	80.56	**81.19**	79.61	TBA	**80.95**

Comparison of AdamW and AdamCPR in a DeiT vision transformer pertaining on ImageNet. We train a small (22M parameters) and a base model (86M) with different regularization parameters.

Conclusion

- CPR regularizes each parameter matrix individual
- CPR is a robust and efficient alternative to weight decay
- AdamCPR needs no additional or less hyperparameters than AdamW
- AdamCPR outperforms AdamW in various experiments.

arxiv.org/abs/2311.09058

github.com/automl/CPR