LightTune: Lightweight Online Fine-tuning for 6G

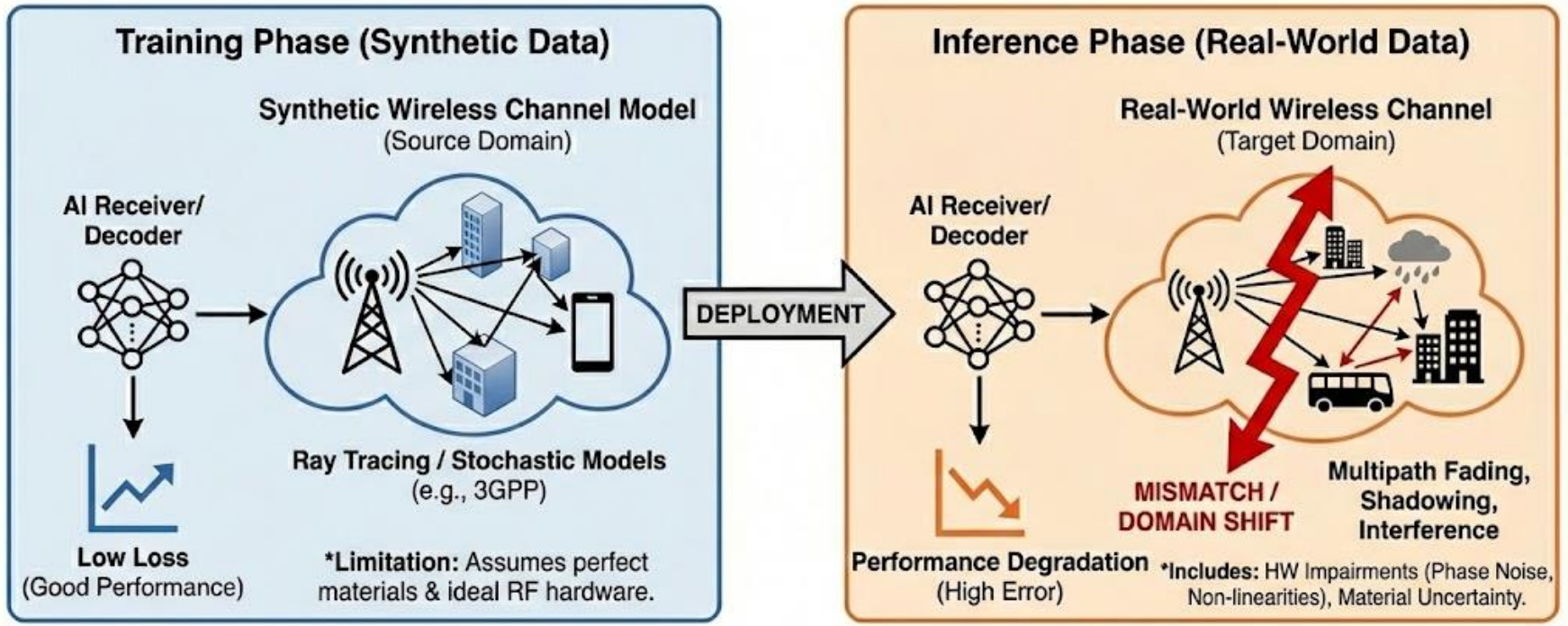
Ramy E. Ali, Federico Penna
Samsung Semiconductor, US
May 27th, 2026

Outline

1. Motivation & Related Works
2. Proposed Online Fine-tuning Algorithm: LightTune
3. Applications of LightTune in 5G/6G Link Adaptation
4. Takeaways

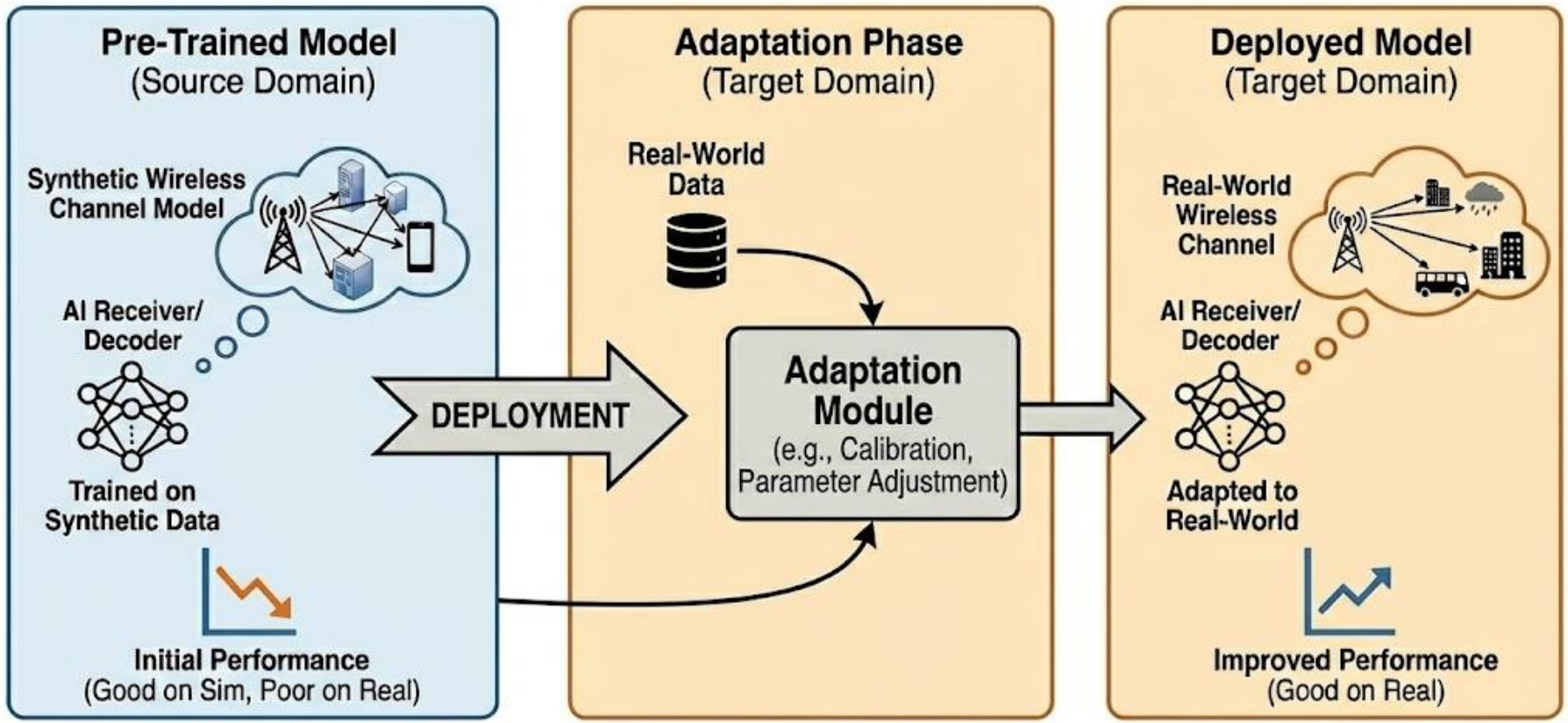
Motivation: Training-Test Mismatch

- ML training done with synthetic data and/or limited real data
- At inference, ML models degrade due to the training-test mismatch
  - Synthetic data does not capture real-world scenarios
  - Even if real-data is used in training, the mismatch can still happen



Motivation: Training-Test Mismatch

- ML models need to adapt while deployed
- For mobiles and edge devices, this needs to be done with minimal storage and computations



Outline

1. Motivation & Related Works
2. Proposed Online Fine-tuning Algorithm: LightTune
3. Applications of LightTune in 5G/6G Link Adaptation
4. Takeaways

LightTune: Big Picture

- Online fine-tuning framework for wireless modems and edge devices
  - Design: Lightweight, Forward-only (Backpropagation-free), …
  - Theory: Guaranteed convergence for ReLU MLPs

- 5G/6G Applications
  - Block Error Rate (BLER) Prediction
  - Link Adaptation
    - ✓ Problem: Baselines algorithms (e.g., outerloop-based) may lead to high BLER
    - ✓ Solution: Predicting BLER ahead & adjusting baseline decisions lowers BLER

- Results
  - Improves wireless throughput by up to 15.5%

Features of LightTune

δ: predefined fine-tuning threshold

1) Backpropagation(BP)-free

- ✓ Fine-tuning using forward passes only
- ✓ Wireless modems and edge devices typically do not support BP

2) Threshold-based

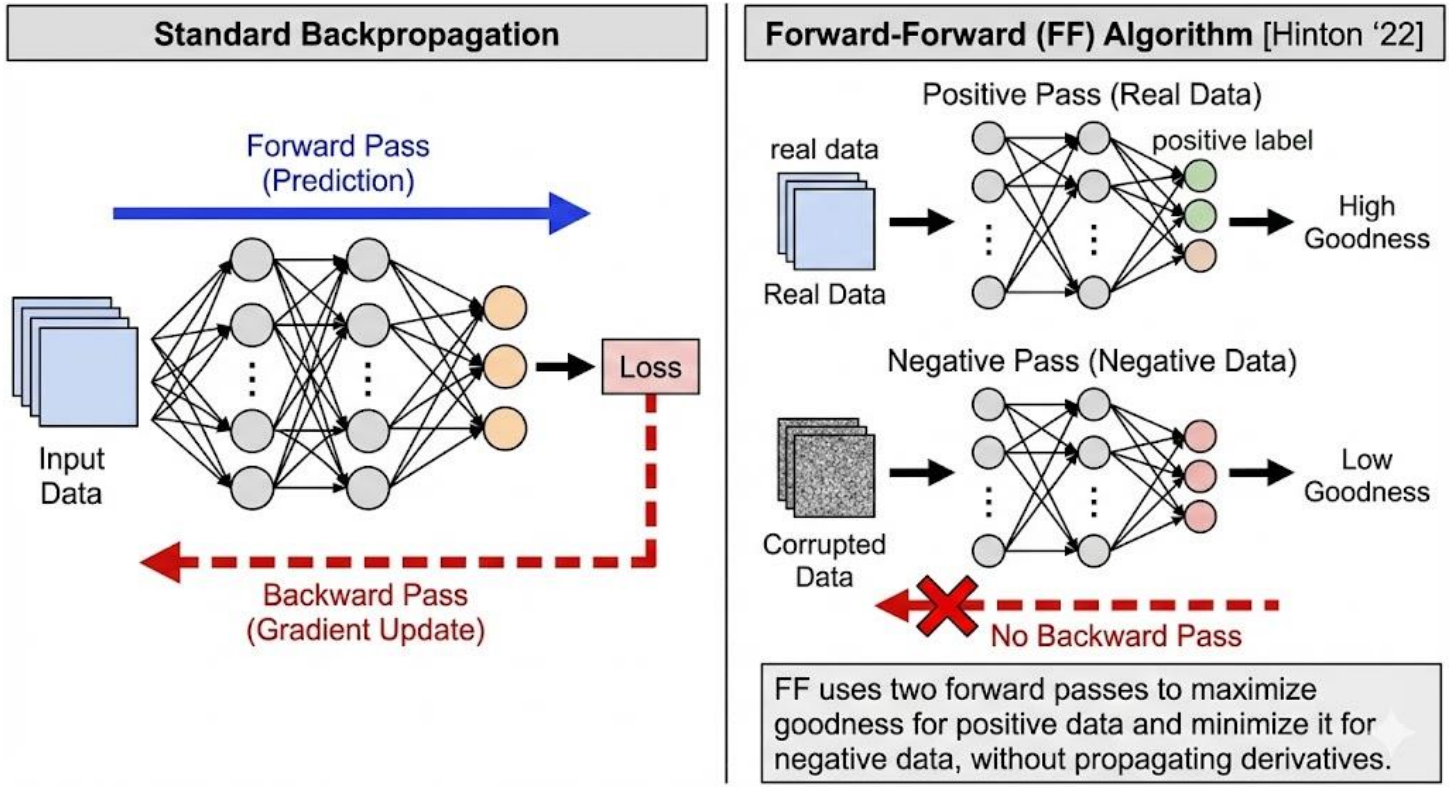
- ✓ Fine-tune only when needed (i.e., when prediction error $\geq \delta$)
- ✓ Ensures convergence (i.e., prediction error $< \delta$ eventually)

3) Buffer-less

- ✓ Sample-by-sample fine-tuning
- ✓ No replay buffer (unlike in RL)

LightTune: Detailed Algorithm

1) Backpropagation-free by leveraging the forward-forward (FF) algorithm



Used mainly in computer vision

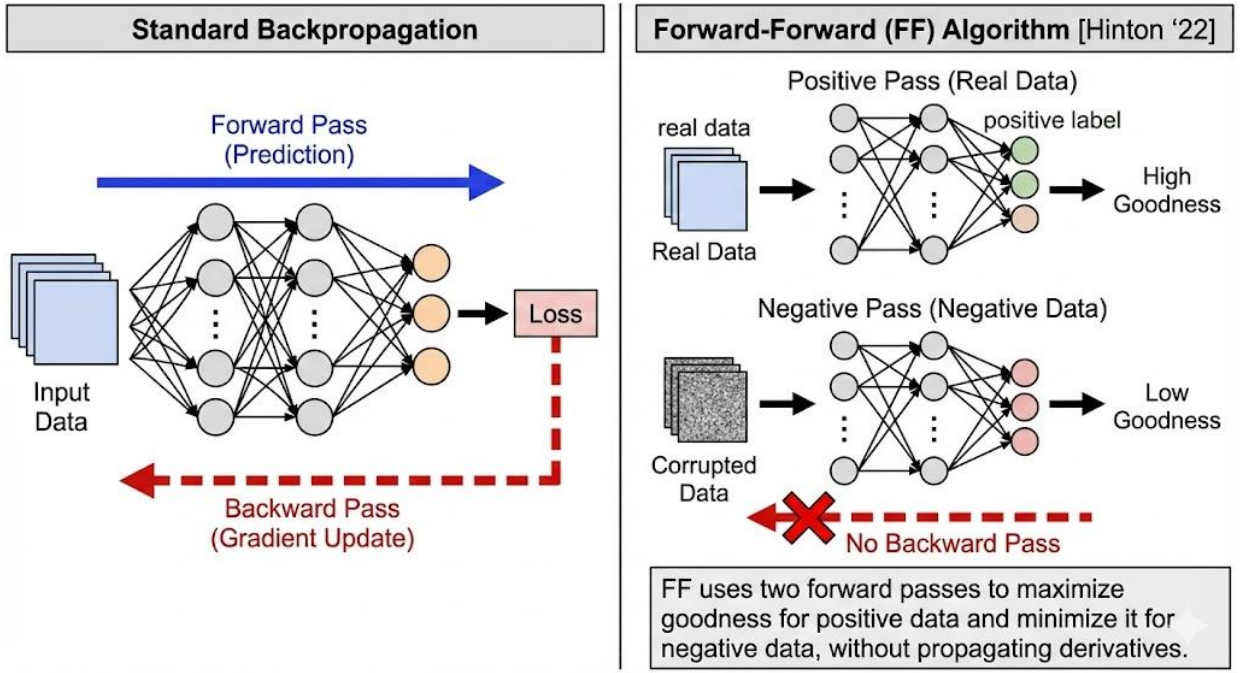
Companies started to use it to leverage the benefits

STMicroelectronics, Intel, Meta, Adobe, Samsung (this work), …

G. Hinton, "The Forward-Forward Algorithm: Some Preliminary Investigations," arXiv preprint arXiv:2212.13345, 2022.

LightTune: Detailed Algorithm

1) Backpropagation-free by leveraging the forward-forward (FF) algorithm



Feature	Standard Backpropagation	LightTune (FF-based)
Peak RAM	$O\left(\sum_{l=1}^{L} M_l\right)$	$O\left(\max_l M_l\right)$
Control Logic	Complex Autodiff	Local Training

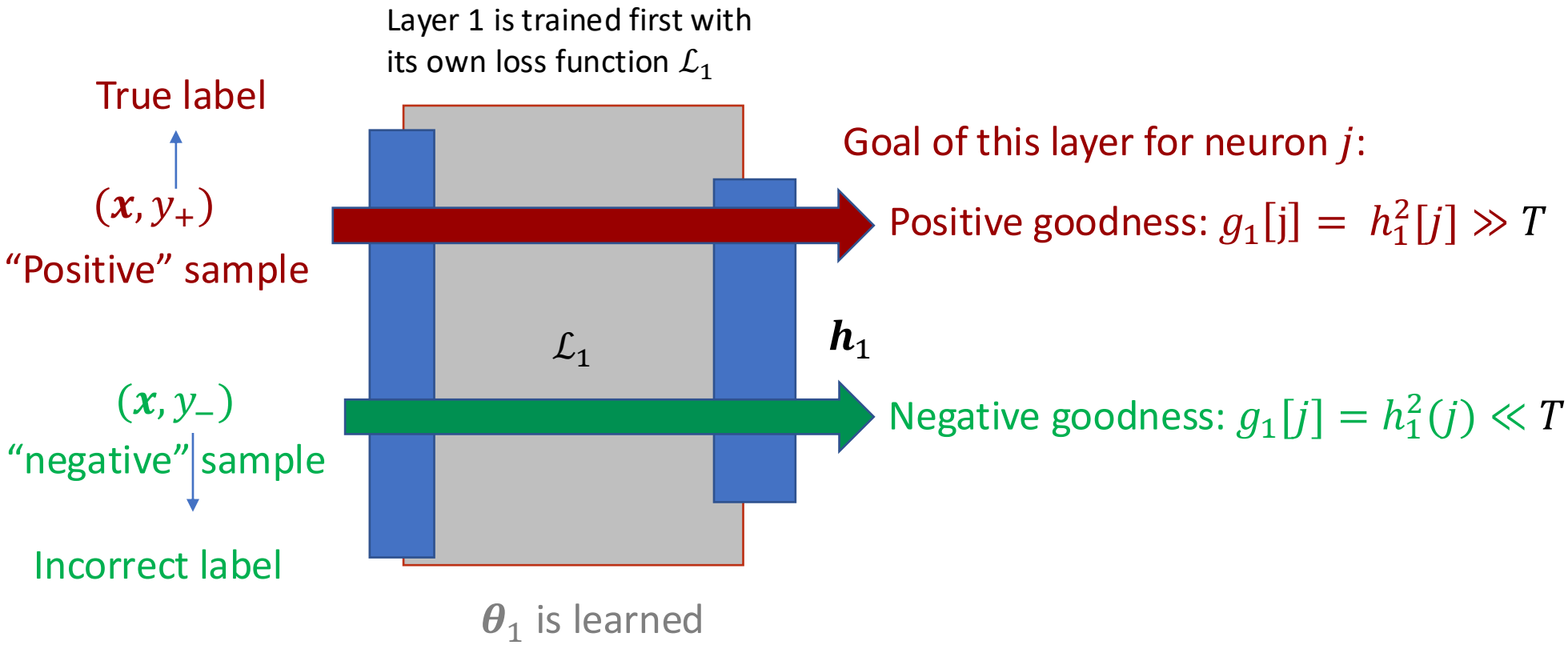
1) Memory-efficient
2) No Autodiff engines needed

TABLE I: Complexity analysis of BP vs. FF for an MLP with L layers where, M_l is the width of the l-th layer.

LightTune: Detailed Algorithm

Training in the forward-forward (FF) algorithm

T: Threshold, hyperparameter
$\boldsymbol{h}_l$: Output of layer l

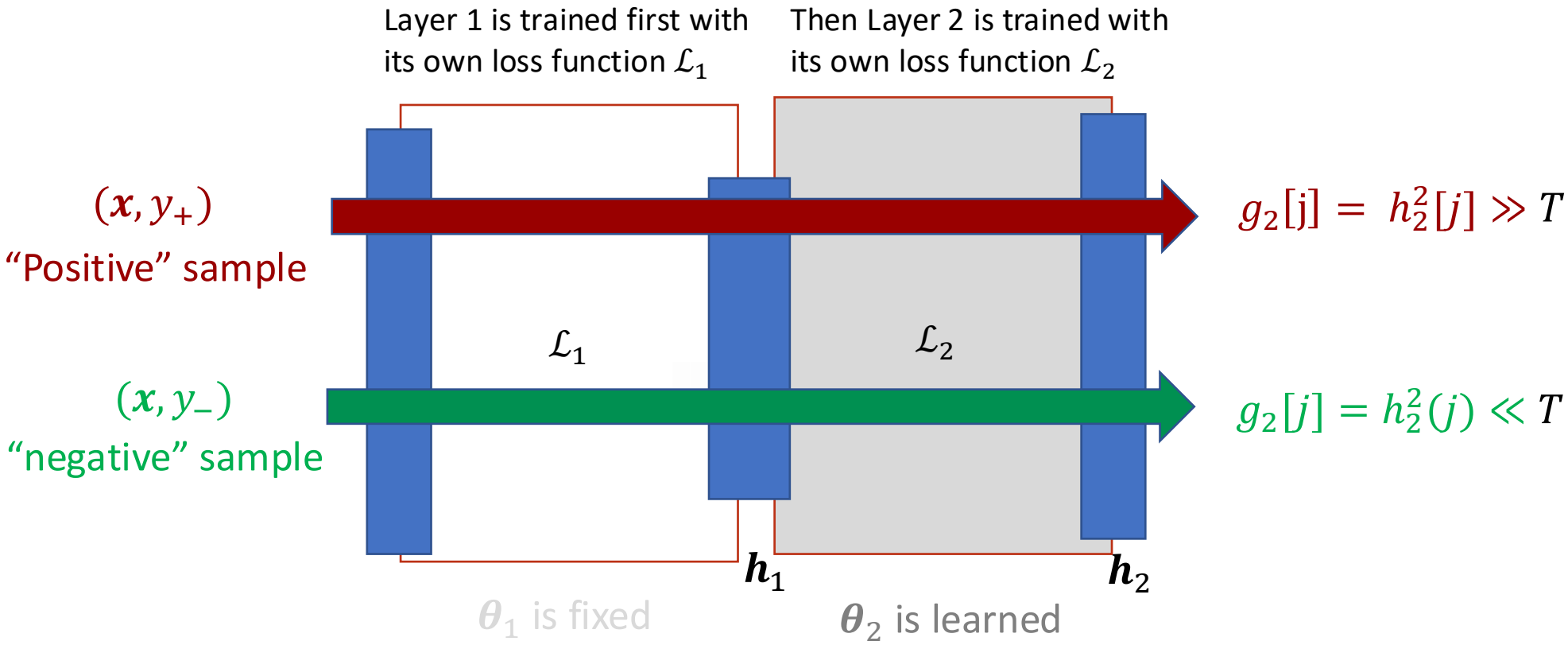


✓ Training is sequential, layer by layer

LightTune: Detailed Algorithm

Training in the forward-forward (FF) algorithm

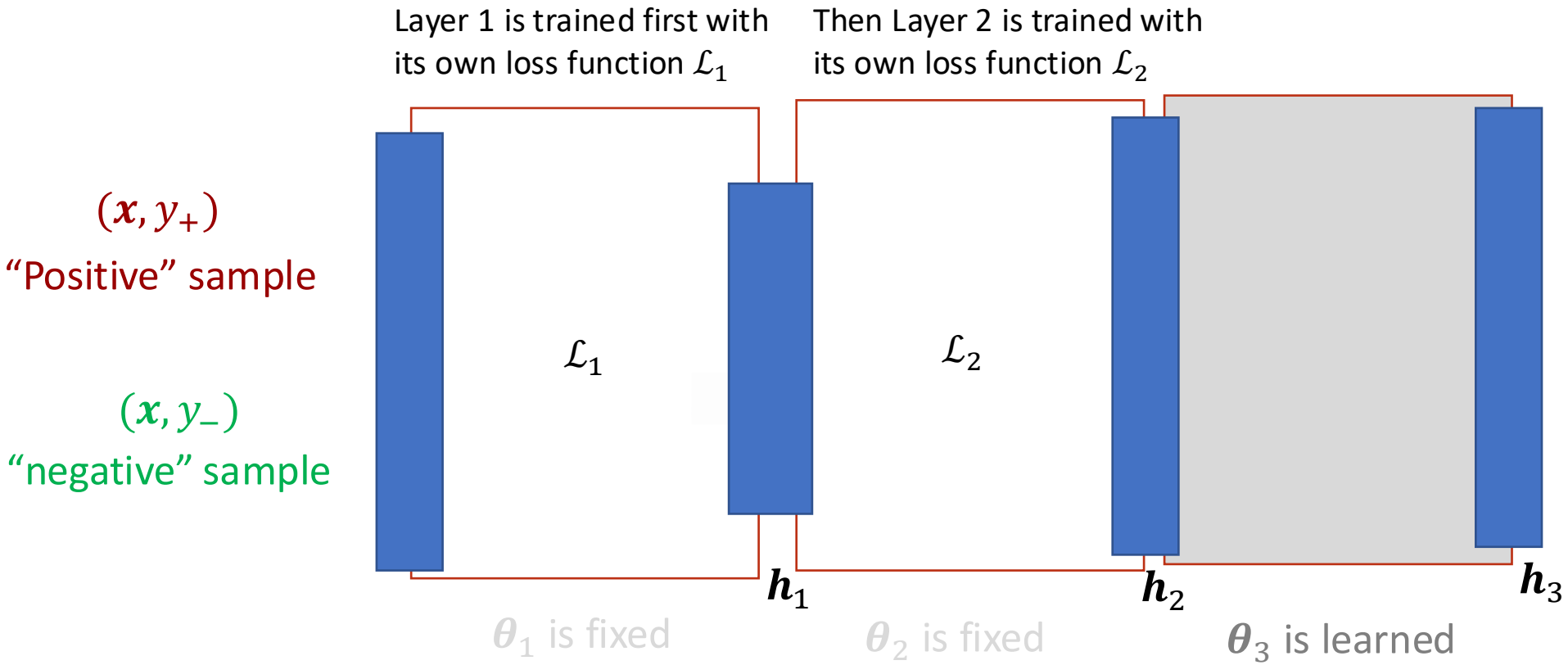
T: Threshold, hyperparameter
$\boldsymbol{h}_l$: Output of layer l



✓ Training is sequential, layer by layer

LightTune: Detailed Algorithm

Training in the forward-forward (FF) algorithm



✓ Training is sequential, layer by layer

LightTune: Detailed Algorithm

$G(x, y)$: goodness of label y when associated with x

Inference in the forward-forward (FF) algorithm

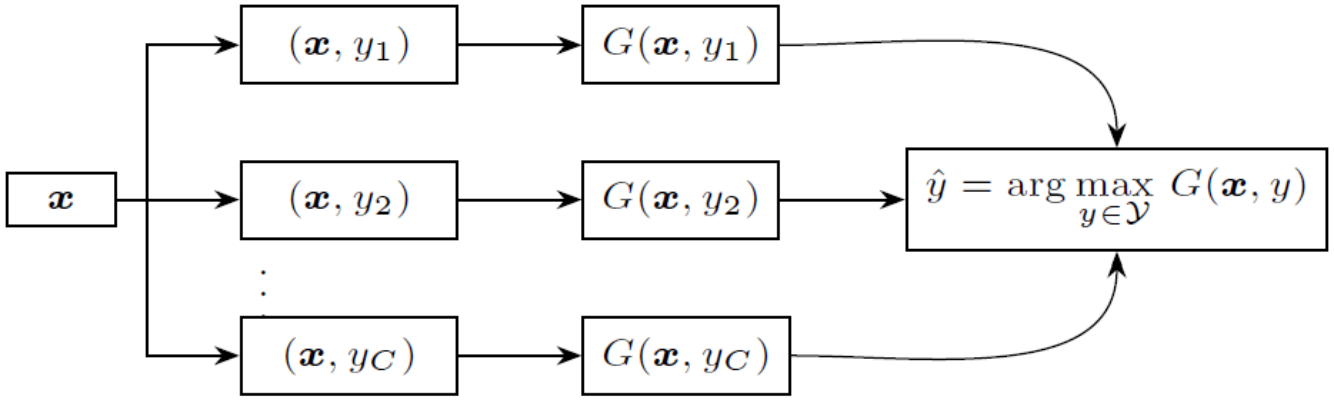


Fig. 1: Inference process in the FF algorithm: the input $\boldsymbol{x}$ is paired with each candidate label $y \in \mathcal{Y}$ and the label with highest goodness is selected.

$$\begin{aligned}\hat{y} &= \arg\max_{y \in \mathcal{Y}} G(\boldsymbol{x}, y) \\ &= \arg\max_{y \in \mathcal{Y}} \left\| \left(f_L \circ \cdots \circ f_1\right) \left([\boldsymbol{x}^\top, y]^\top\right) \right\|_2^2 .\end{aligned}$$

LightTune: Proposed Loss Function

$g_{+,l}[\mathrm{j}]$: positive goodness of neuron j in layer l
$g_{-,l}[\mathrm{j}]$: negative goodness of neuron j in layer l
$\mathcal{S}_+$: set of positive samples
$\mathcal{S}_-$: set of negative samples

1) Backpropagation-free by leveraging the forward-forward (FF) algorithm

- Softplus loss typically used in the FF algorithm

>>

$$\mathcal{L}_{\text{Softplus},l} = \frac{1}{|\mathcal{S}_+|M_l}\sum_{i\in\mathcal{S}_+}\sum_{j=1}^{M_l}\ln\left(1+e^{-(g_{+,l}^{(i)}[j]-T)}\right) + \frac{1}{|\mathcal{S}_-|M_l}\sum_{i\in\mathcal{S}_-}\sum_{j=1}^{M_l}\ln\left(1+e^{g_{-,l}^{(i)}[j]-T}\right), \quad (5)$$

<<

Encourages positive goodness to be above threshold T
& negative goodness to be lower than T

LightTune: Proposed Loss Function

$g_{+,l}[j]$: positive goodness of neuron j in layer l
$g_{-,l}[j]$: negative goodness of neuron j in layer l
$\mathcal{S}_+$: set of positive samples
$\mathcal{S}_-$: set of negative samples

1) Backpropagation-free by leveraging the forward-forward (FF) algorithm

- Softplus loss typically used in the FF algorithm

$$\mathcal{L}_{\text{Softplus},l} = \frac{1}{|\mathcal{S}_+|M_l}\sum_{i\in\mathcal{S}_+}\sum_{j=1}^{M_l}\ln\left(1+e^{-(g_{+,l}^{(i)}[j]-T)}\right) + \frac{1}{|\mathcal{S}_-|M_l}\sum_{i\in\mathcal{S}_-}\sum_{j=1}^{M_l}\ln\left(1+e^{g_{-,l}^{(i)}[j]-T}\right), \quad (5)$$

Needs exponentiations & divisions to compute derivatives !!

$$\mathcal{L}_{\text{Prop},l} = \frac{1}{|\mathcal{S}_+|M_l}\sum_{i\in\mathcal{S}_+}\sum_{j=1}^{M_l}\left((g_{+,l}^{(i)}[j]-T)^2-4(g_{+,l}^{(i)}[j]-T)\right) + \frac{1}{|\mathcal{S}_-|M_l}\sum_{i\in\mathcal{S}_-}\sum_{j=1}^{M_l}\left((g_{-,l}^{(i)}[j]-T)^2+4(g_{-,l}^{(i)}[j]-T)\right). \quad (7)$$

Lightweight & smooth
Smoothness is important for convergence (will see later)

LightTune: Lightweight Gradient Computations

1) Backpropagation-free by leveraging the forward-forward (FF) algorithm

- Closed-form gradients for each layer
- Just lightweight computations to fine-tune

$h_{+,l}$: output of layer l for the positive sample
$g_{+,l}$: goodness of layer l for the positive sample
$h_{-,l}$: output of layer l for the negative sample
$g_{-,l}$: goodness of layer l for the negative sample
T: threshold
$\boldsymbol{\Theta}_l$: parameters of layer l

$$\nabla_{\Theta_l}\mathcal{L}_l = \nabla_{\Theta_l}\mathcal{L}_{+,l} + \nabla_{\Theta_l}\mathcal{L}_{-,l},$$

gradient using the positive sample $(\boldsymbol{x}, y_+)$

gradient using the negative sample $(\boldsymbol{x}, y_-)$

$$\nabla_{\Theta_l}\mathcal{L}_{+,l} = \frac{4}{M_l}\left[(\boldsymbol{g}_{+,l} - T - 2) \odot \boldsymbol{h}_{+,l} \odot \mathbb{1}(\boldsymbol{h}_{+,l} > 0)\right]\tilde{\boldsymbol{h}}_{+,l-1}^{\top},$$
$$\nabla_{\Theta_l}\mathcal{L}_{-,l} = \frac{4}{M_l}\left[(\boldsymbol{g}_{-,l} - T + 2) \odot \boldsymbol{h}_{-,l} \odot \mathbb{1}(\boldsymbol{h}_{-,l} > 0)\right]\tilde{\boldsymbol{h}}_{-,l-1}^{\top}.$$

for ReLU MLPs

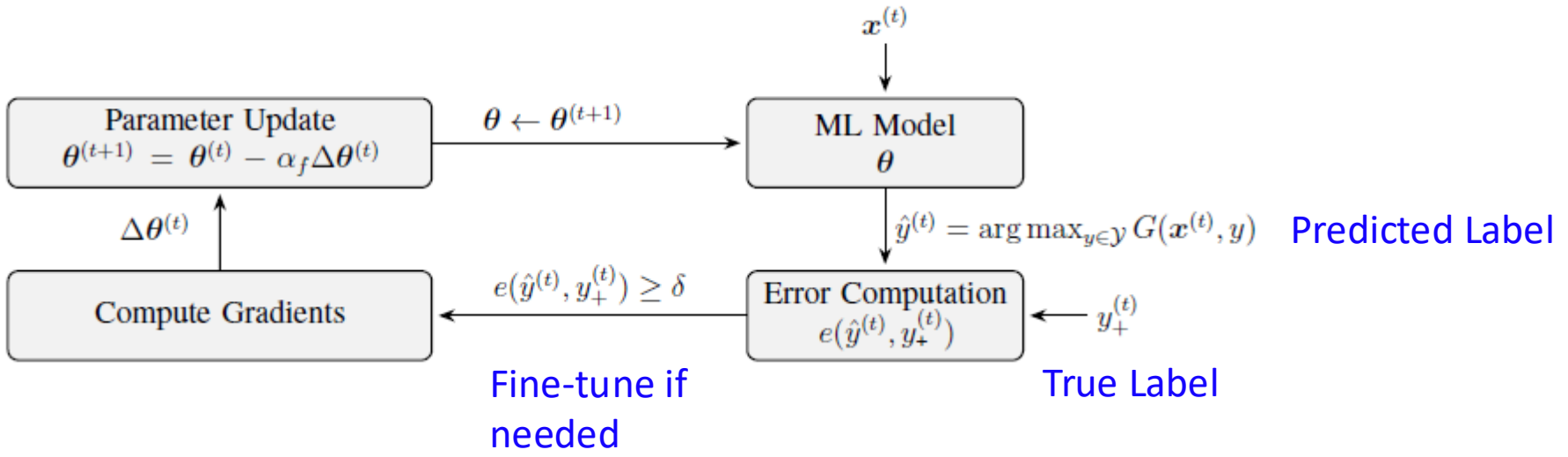
LightTune: Detailed Algorithm

2) Threshold-based

- Fine-tuning is only performed when needed
  - ✓ When error prediction error reaches a pre-defined threshold δ

3) Buffer-less

- Updates are performed on a sample-by-sample basis (i.e., no buffer)



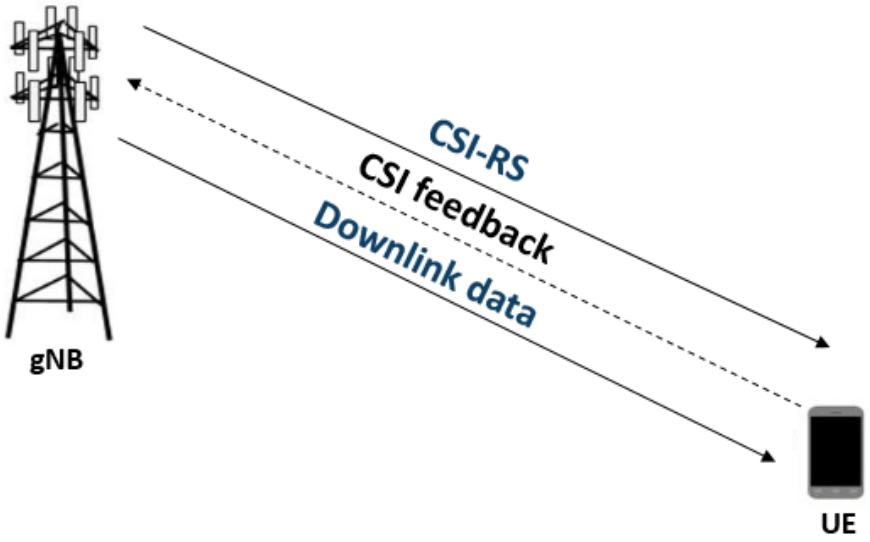
Outline

1. Motivation & Related Works
2. Proposed Online Fine-tuning Algorithm: LightTune
3. Applications of LightTune in 5G/6G Link Adaptation
4. Takeaways

Wireless Link Adaptation: Background

- Channel quality indicator (CQI) measures the quality of the channel between gNB and UE
  - UE reports this to gNB
  - gNB decides based on this what modulation and coding scheme to use
- Rank Indicator (RI) measures the number of useful spatial layers (streams) the channel can support
  - UE reports this to gNB
  - gNB decides based on this how many parallel data streams to transmit to the UE

Base station



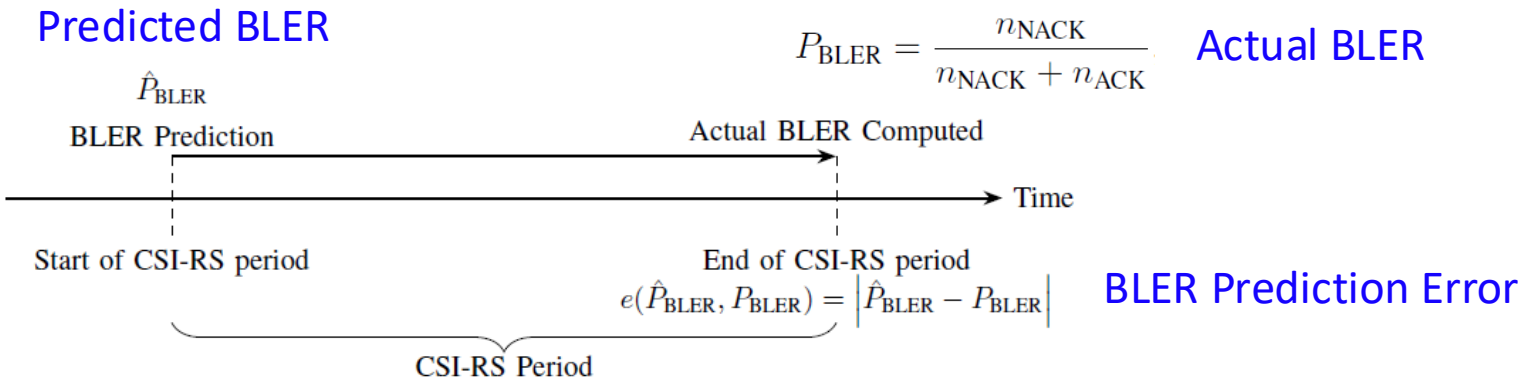
CSI feedback includes CQI, RI, PMI

User Equipment

Block Error Rate (BLER) Prediction using LightTune

- Idea
  1. ML model predicts the BLER of the baseline link adaptation scheme
  2. Adjust the baseline decision to prevent high BLER

- The ML model is fine-tuned using this ground-truth data



$$P_{\mathrm{BLER}} = \frac{n_{\mathrm{NACK}}}{n_{\mathrm{NACK}} + n_{\mathrm{ACK}}}$$

Actual BLER

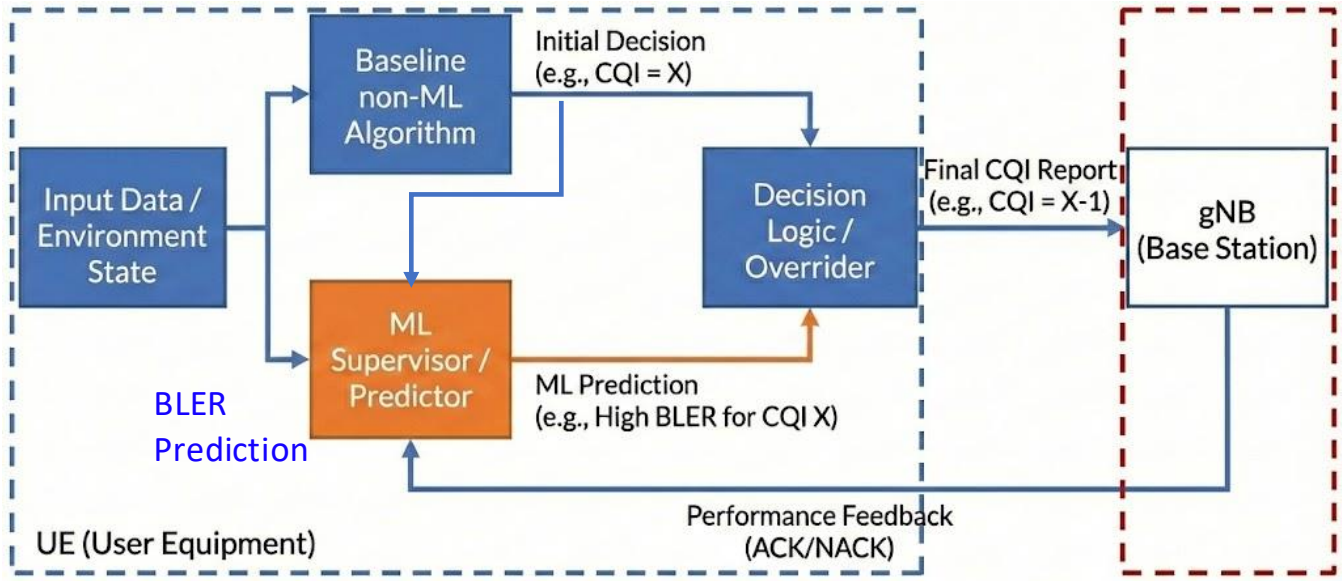
BLER Prediction Error

n_{NACK}: # of NACKS in CSI-RS period
n_{ACK}: # of ACKs in CSI-RS period

Link Adaptation: Conservative CQI Selection

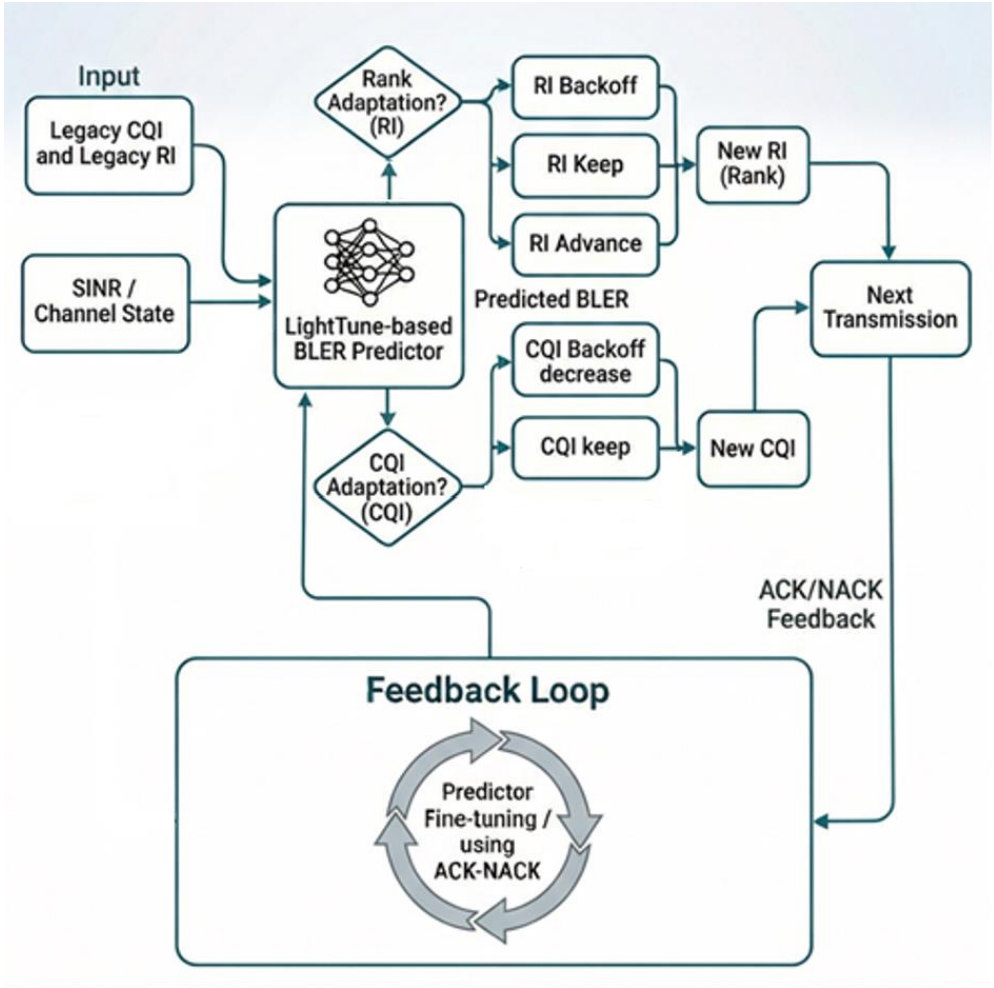
- Table-based baselines tend to select high CQIs.
- To mitigate this high BLER, we use our BLER prediction algorithm as a backoff mechanism.

CSI-RS SNR, …



Link Adaptation: Joint Rank Indicator (RI) and CQI Selection

- More generally, we apply LightTune for joint RI and CQI Selection.
- Reduced RI search is used to limit the complexity.



Simulation Results

- To simulate the training-test mismatch, we choose mismatched training and test conditions.

Parameter	Training	Test
Channels	TDL-A30	TDL-A10, TDL-A30, TDL-B50, TDL-B100, TDL-C200
SNR	Low (0–12 dB)	Low/Medium/High (0–40 dB)
Delay Profile	Low Delay	Low/High Delay
Doppler Frequency	Low (10 Hz)	Low/Medium (10–50 Hz)
Antenna Correlation	Low	Low/Medium/High
CSI-RS Period	80 ms	10, 40 or 80 ms

TABLE II: Training and testing configurations. The low SNR range is from 0 to 12 dB, the medium SNR range is from 16 to 24 dB and the high SNR range is from 28 to 40 dB

Correlation	BS Correlation (α)	UE Correlation (β)
Low	0	0
Medium	0.3	0.9
High	0.9	0.9

TABLE III: Antenna Correlation Scenarios [31], where α and β represent the BS and the UE antenna correlation coefficients, respectively

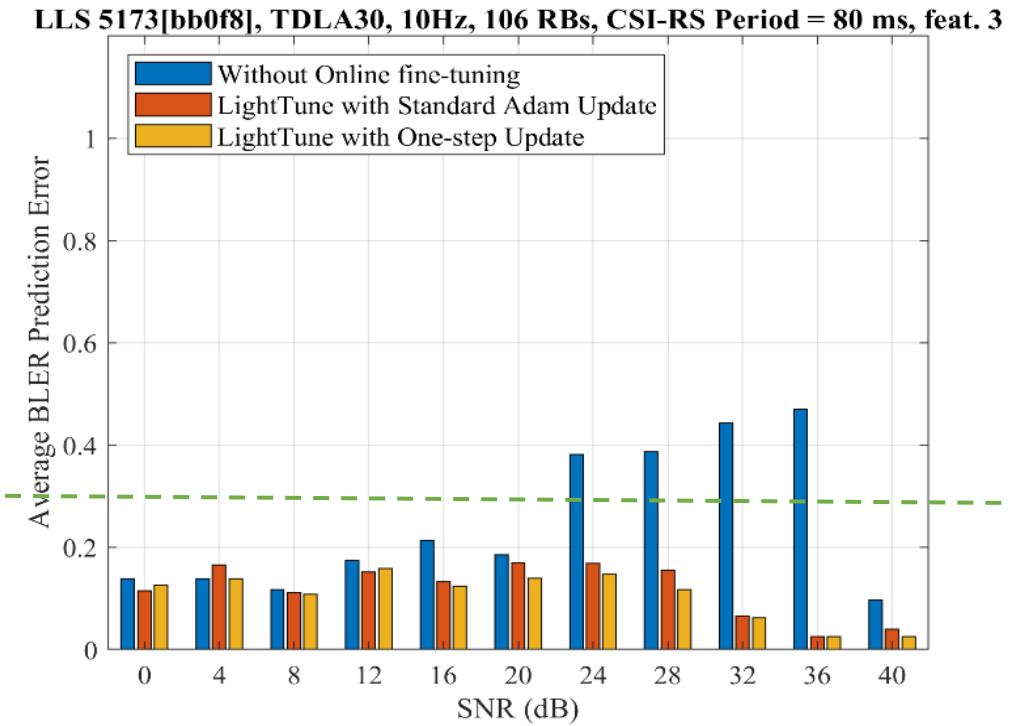
Parameter	Value
Neural Network Size	`[13, 32, 32]`
Activation Function	ReLU
Offline Learning Rate α	0.03
Online Learning Rate α_f	0.03
Fine-tuning Threshold δ	0.3
Training Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$)
Threshold T	9
Epochs	22,000
Training Samples	83,200
BLER Quantization Step	$\lambda = 0.1$

TABLE IV: Hyperparameters for the BLER prediction algorithm

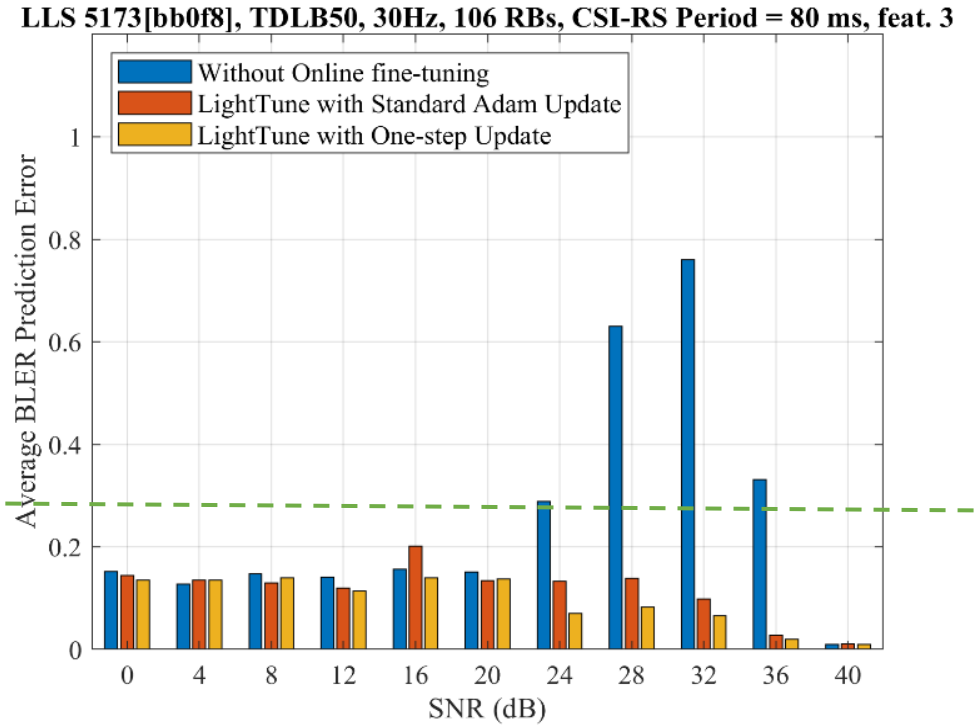
Lightweight
BLER Prediction
ML Model

BLER Prediction with LightTune

- Online fine-tuning is a must!



Trained Channel (0-12 dB)



Untrained Channel

fine-tuning threshold $\delta = 0.3$

Link Adaptation with LightTune

- RI-CQI-Tune yields up to 15.5% throughput improvement

CSI-RS Period	Channel Profile	Antenna Correlation	Medium SNR Gain		High SNR Gain	
			CQI-Tune	RI-CQI-Tune	CQI-Tune	RI-CQI-Tune
80 ms	TDL-A10, 20 Hz	Low	5.3%	2.6%	1.3%	2.6%
	TDL-B50, 30 Hz	Low	12.1%	8.1%	2%	1.1%
	TDL-B200, 50 Hz	Low	7.0%	6.3%	0.7%	11.0%
	TDL-C200, 50 Hz	Low	9.1%	8.5%	1.3%	10.9%
	TDL-B50, 30 Hz	Medium	1.1%	1.7%	0.8%	0.9%
	TDL-B200, 50 Hz	Medium	1.5%	6.2%	3.4%	15.5%
	TDL-B50, 30 Hz	High	0.6%	1.4%	0.1%	2.1%
	TDL-B200, 50 Hz	High	−0.6%	1.4%	0.7%	12.6%
40 ms	TDL-B50, 30 Hz	Medium	0.3%	1.6%	0.4%	5.2%
	TDL-B50, 30 Hz	High	0.5%	2.0%	2.4%	4.2%
	TDL-C200, 50 Hz	Low	3.0%	6.7%	0.7%	9.1%
	TDL-C200, 50 Hz	High	0.2%	5.2%	0.6%	11.5%
10 ms	TDL-B50, 30 Hz	Low	2.0%	1.0%	0.2%	1.8%
	TDL-C200, 50 Hz	Low	0.7%	0.1%	−0.2%	7.7%
	TDL-A10, 20 Hz	Medium	0.6%	0.2%	0.2%	3.0%
	TDL-C200, 50 Hz	Medium	0.0%	3.1%	2.2%	6.1%

TABLE 6: Throughput Gains of CQI-Tune and RI-CQI-Tune Across Various Scenarios

LightTune: Convergence

$e^{(t)}$: prediction error at time $t = |\hat{y}^{(t)} - y_+^{(t)}|$

Theorem: Assuming 1) ReLU MLPs, 2) boundedness of data & model,

3) offline & online distributions $\mathcal{D}_1$ & $\mathcal{D}_2$ and 4) gradient lower bound $\gamma_2(\delta)$, then

Theorem 1 (Convergence under Distribution Shift). *Suppose Assumptions 1, 2, 4, and 3 hold. For a fixed error tolerance $\delta > 0$ and a learning rate $\alpha_f \in (0, 1/\rho_L)$, LightTune satisfies for any $N \geq 1$:*

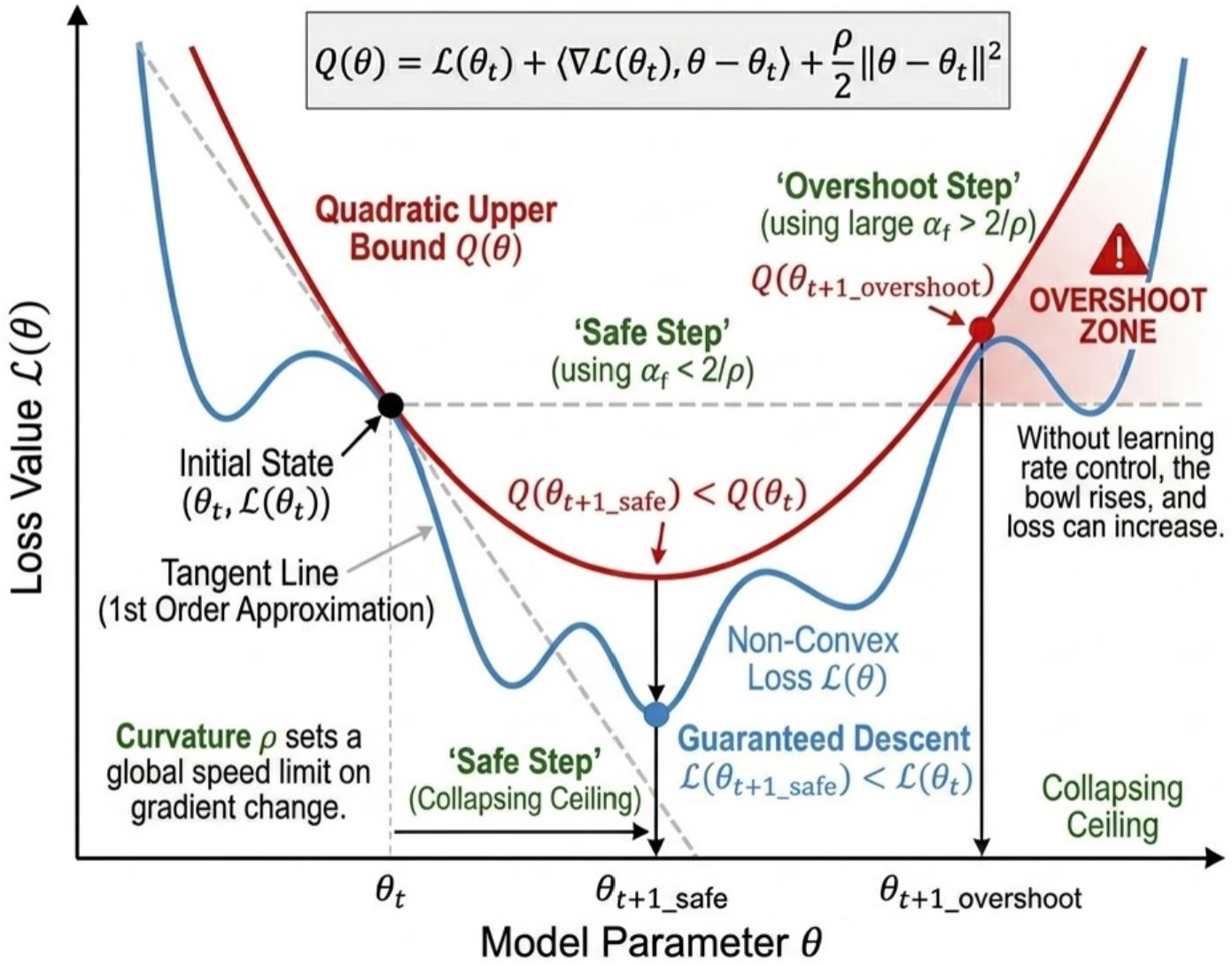
$$\frac{1}{N}\sum_{t=1}^{N} \Pr_{\mathcal{D}_2}\left(e^{(t)} \geq \delta\right) \leq \frac{2\left[\mathbb{E}_{\mathcal{D}_1}[\mathcal{L}_L^{(1)}(\theta_L^{(1)})] - \mathcal{L}_L^*\right]}{\alpha_f \gamma_2(\delta) N} + \frac{2M}{\alpha_f \gamma_2(\delta)}\sqrt{\frac{2D_{\mathrm{KL}}(\mathcal{D}_2 \| \mathcal{D}_1)}{N}}, \tag{27}$$

where $\mathcal{L}_L^* = \inf_{\boldsymbol{\theta}} \mathbb{E}_{\mathcal{D}_2}[\mathcal{L}_L^{(t)}(\theta)]$ is the minimum achievable expected loss under $\mathcal{D}_2$, the constant M satisfies $\sup_{t\geq 1}|\mathcal{L}_L^{(t)}(\theta_L^{(t)})| \leq M$, and $\gamma_2(\delta)$ is the gradient lower bound from Assumption 4.

Corollary 1 (Asymptotic Convergence of Average Error Probability). *Under the conditions of Theorem 1, for any fixed error tolerance $\delta > 0$, the average probability of a significant prediction error satisfies*

$$\lim_{N\to\infty} \frac{1}{N}\sum_{t=1}^{N} \Pr_{\mathcal{D}_2}\left(e^{(t)} \geq \delta\right) = 0. \tag{28}$$

Proof Intuition: Why smoothness implies convergence?



Proof Intuition: Why global ρ-Smoothness implies Descent

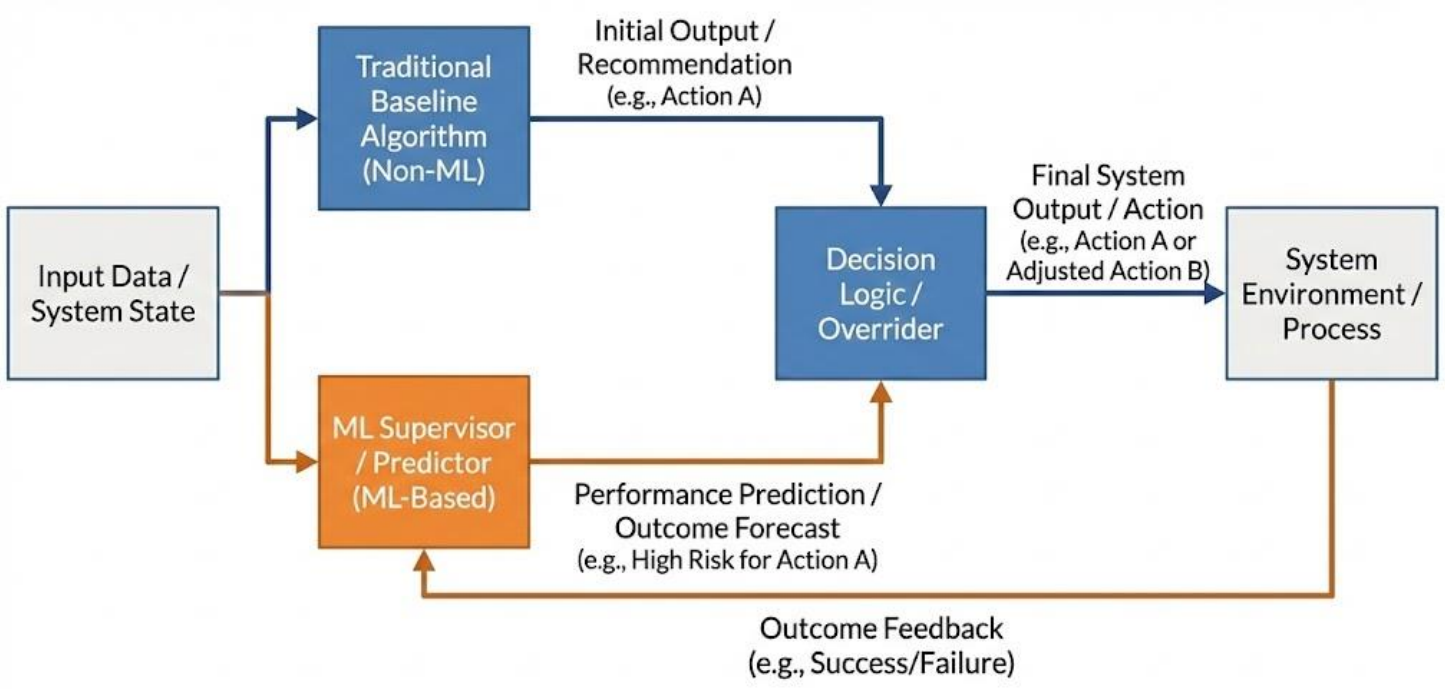
1. **Parabola Tangency:** $Q(\theta_t) = \mathcal{L}(\theta_t)$ at start.
2. **Safe Basin Step:** A controlled fine-tuning step size α_f ensures we land at θ_{t+1} where the parabola's height has physically dropped: $Q(\theta_{t+1}) < Q(\theta_t)$.
3. **The Trap:** Because $\mathcal{L}$ is trapped under Q globally, and Q just dropped in height, the original loss must drop:

$$\mathcal{L}(\theta_{t+1}) \leq Q(\theta_{t+1}) < Q(\theta_t) = \mathcal{L}(\theta_t).$$

SAMSUNG

Takeaways

1. Online fine-tuning is essential for ML-based wireless algorithms
2. Instead of using ML entirely, ML can serve just as a guide for the baseline algorithm
   - Less complexity than using ML entirely
   - Starting from good initial solution (i.e., baseline solution)



Thank you!
Questions?

Email: ramy.ali@samsung.com

Extended version: https://arxiv.org/pdf/2604.12406