LightTune: Lightweight Online Fine-tuning for 6G

Ramy E. Ali & Federico Penna

Cellular & Multimedia Lab, Samsung Semiconductor, San Diego

02-10-2026

To appear in IEEE ICC, 2026.

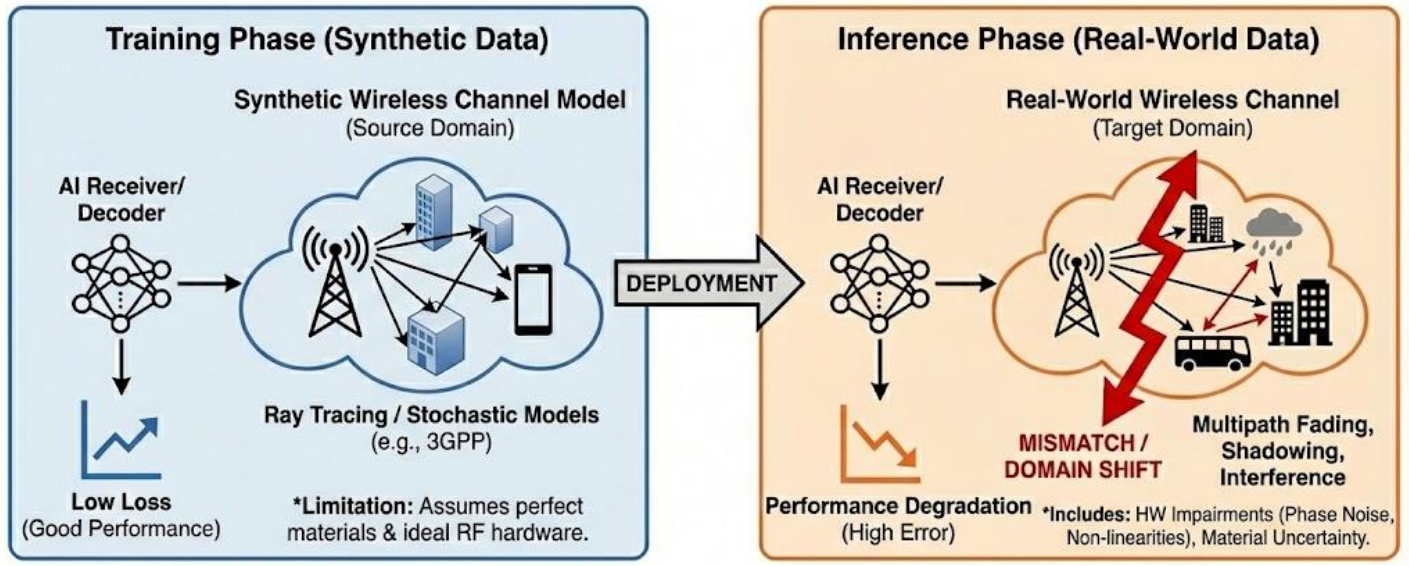
SAMSUNG

Outline

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

Motivation: Training-Test Mismatch

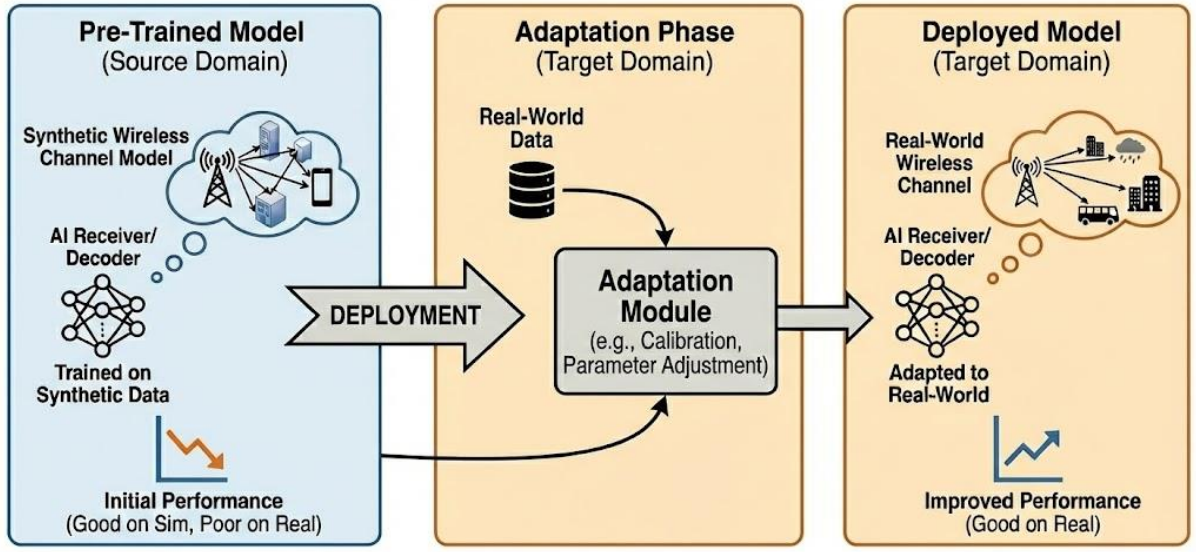
- ML training done with synthetic data and/or limited real data
- At inference, their performance may degrade due to the training-test mismatch
  - Synthetic data does not capture the real-world complexity
  - Even if real-data is used in training, the mismatch can still happen



SAMSUNG

Motivation: Training-Test Mismatch

- ML models need to adapt after being deployed
- For mobiles, this needs to be done with minimal storage and computations



SAMSUNG

Background and Related Works

- How can on-device ML models adapt with minimal overhead?

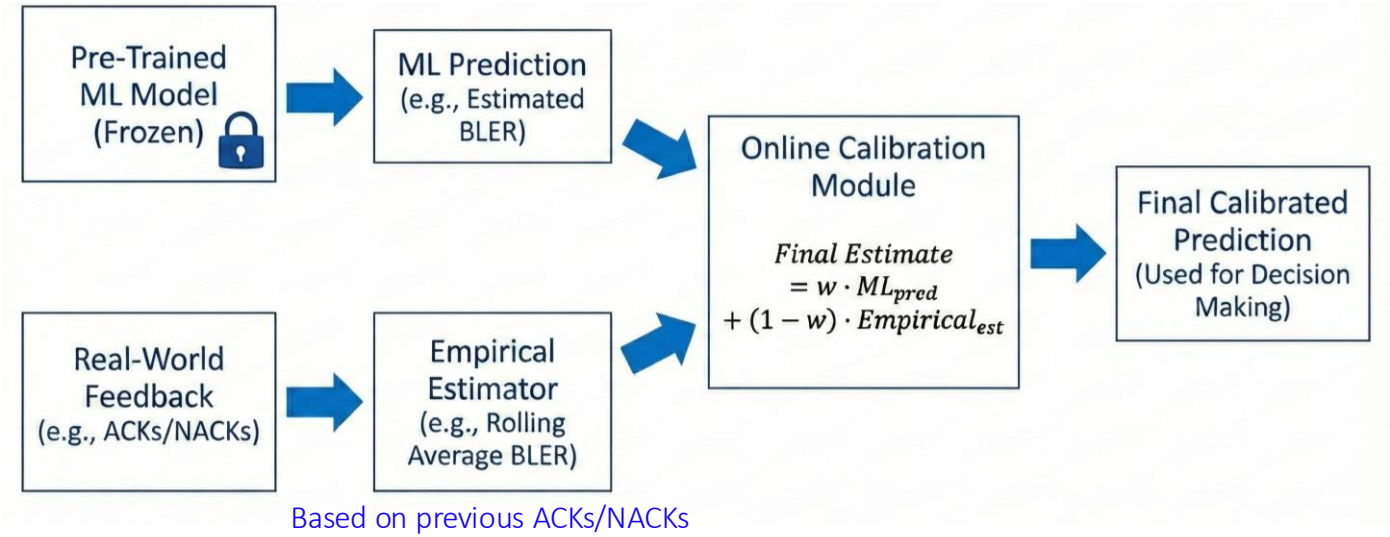
SAMSUNG

Background and Related Works

- How can on-device ML models adapt with minimal overhead?
- Prior works in ML-based Wireless
  1) Online Calibration/Adaptation
     - ML models are not updated
     - Then, ML predictions are successively calibrated based on the real-world observations

Background and Related Works

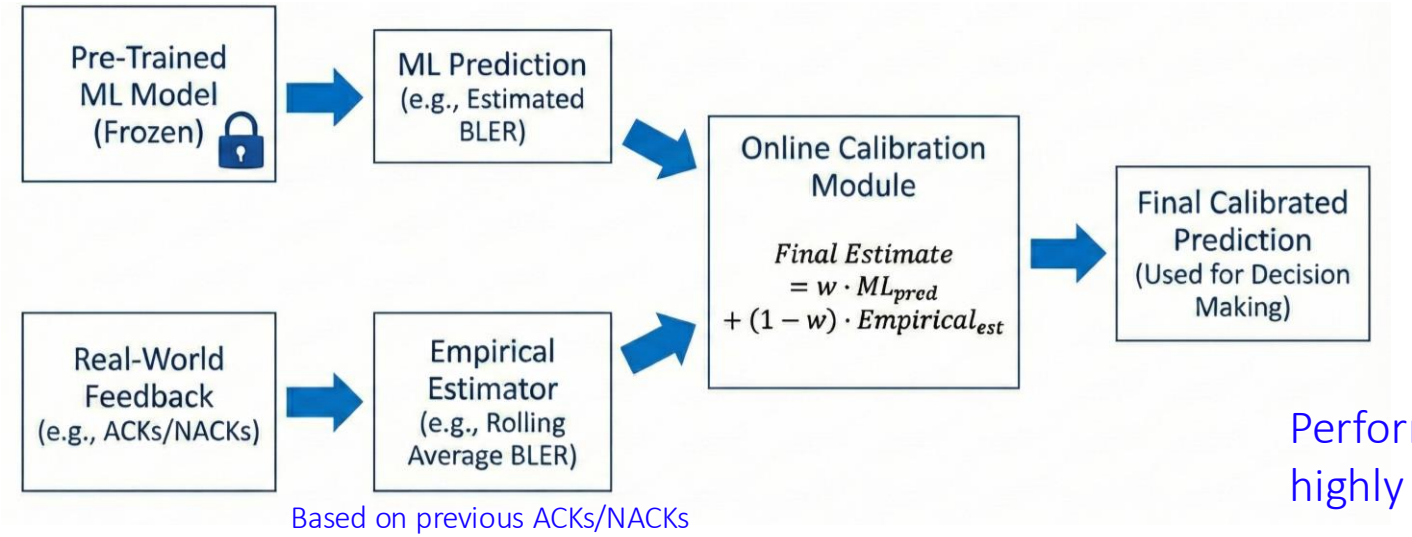
- How can on-device ML models adapt with minimal overhead?
- Prior works in ML-based Wireless
  1) Online Calibration/Adaptation
     - ML models are not updated
     - Then, ML predictions are successively calibrated based on the real-world observations
     - Example: Block Error Rate (BLER) Prediction



[A. Baknina et al., "Adaptive CQI and RI Estimation for 5G NR: A Shallow Reinforcement Learning Approach," in IEEE GLOBECOM, 2020]
[Y. Huang et al., "DELUXE: A DL-based link adaptation for URLLC/eMBB multiplexing in 5G NR," IEEE JSAC, 2021.]

Background and Related Works

- How can on-device ML models adapt with minimal overhead?
- Prior works in ML-based Wireless
  1) Online Calibration/Adaptation
     - ML models are not updated
     - Then, ML predictions are successively calibrated based on the real-world observations
     - Example: Block Error Rate (BLER) Prediction



Performance still degrades in highly dynamic environments !

[A. Baknina et al., "Adaptive CQI and RI Estimation for 5G NR: A Shallow Reinforcement Learning Approach," in IEEE GLOBECOM, 2020]
[Y. Huang et al., "DELUXE: A DL-based link adaptation for URLLC/eMBB multiplexing in 5G NR," IEEE JSAC, 2021.]

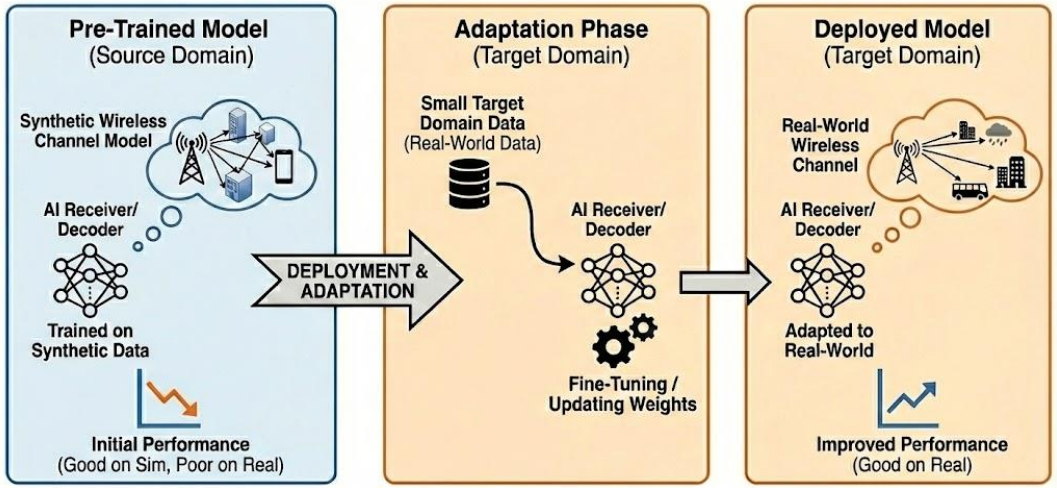
SAMSUNG

Background and Related Works

- How can on-device ML models adapt with minimal overhead?
- Prior works in ML-based Wireless
  - 2) Online Learning (e.g., Reinforcement Learning, Continual Learning)
    - Requires frequent model updates, online gradient computations, backpropagation, replay buffer, …
    - Mobiles (esp. in wireless modem) may not be able to support this

Background and Related Works

- How can on-device ML models adapt with minimal overhead?
- Prior works in ML-based Wireless
  - 2) Online Learning (e.g., Reinforcement Learning, Continual Learning)
    - Requires frequent model updates, online gradient computations, backpropagation, replay buffer, …
    - Mobiles (esp. in wireless modem) may not be able to support this
  - 3) Online Fine-tuning (Simpler)
    - Mobiles may not be able to support this as well



SAMSUNG

Our work: LightTune

- A lightweight online fine-tuning algorithm suitable for on-device learning
  - e.g., mobile phones (esp. wireless modem), edge devices, …
- Suitable when the ground-truth of the predicted metric becomes available after a delay
  - e.g., Throughput prediction, Block error rate (BLER) prediction, …
- As an application, we use LightTune for link adaptation in 5G/6G
  - Up to 9.5% average throughput improvement compared to baselines

Our work: LightTune

1) Backpropagation-free
   - ✓ Fine-tuning using forward passes only

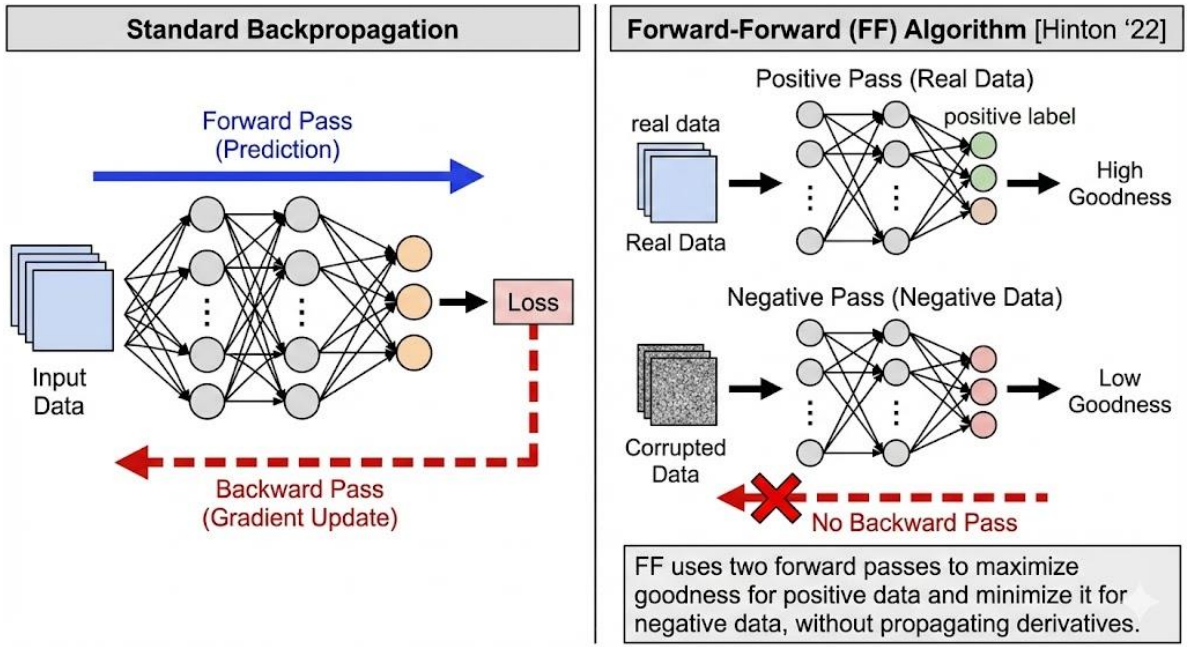
2) Threshold-based
   - ✓ Only fine-tunes when needed, not continuously

3) Buffer-less
   - ✓ Sample-by-sample fine-tuning

SAMSUNG

LightTune: Detailed Algorithm

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



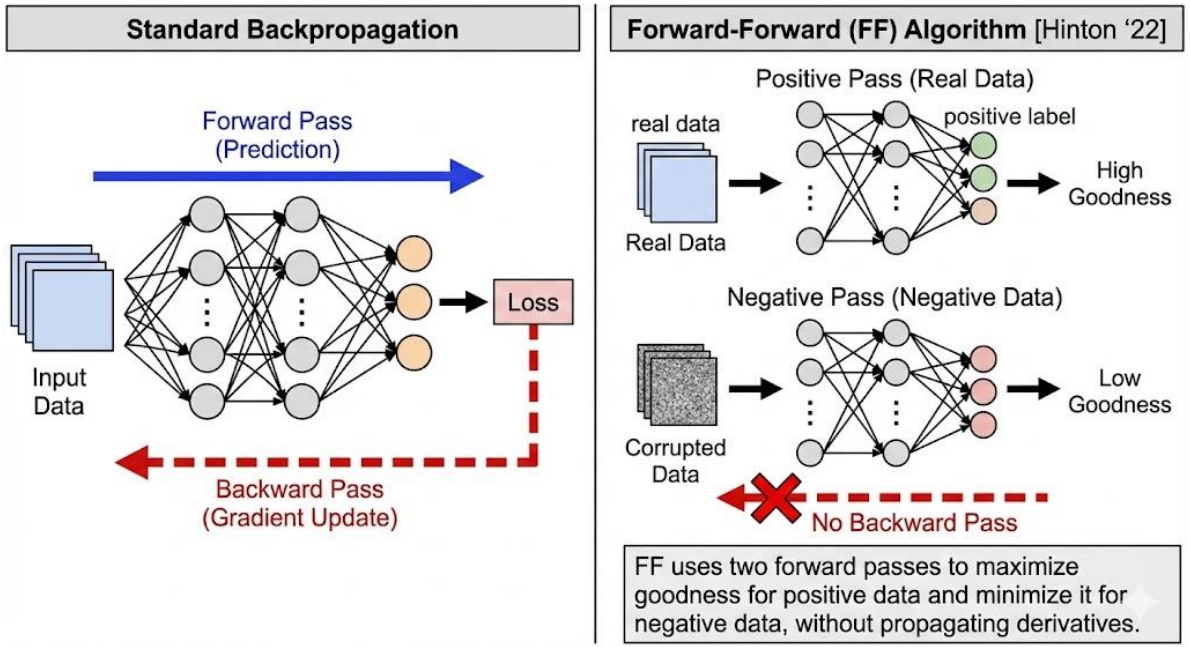
Used mainly in computer vision

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

SAMSUNG

LightTune: Detailed Algorithm

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

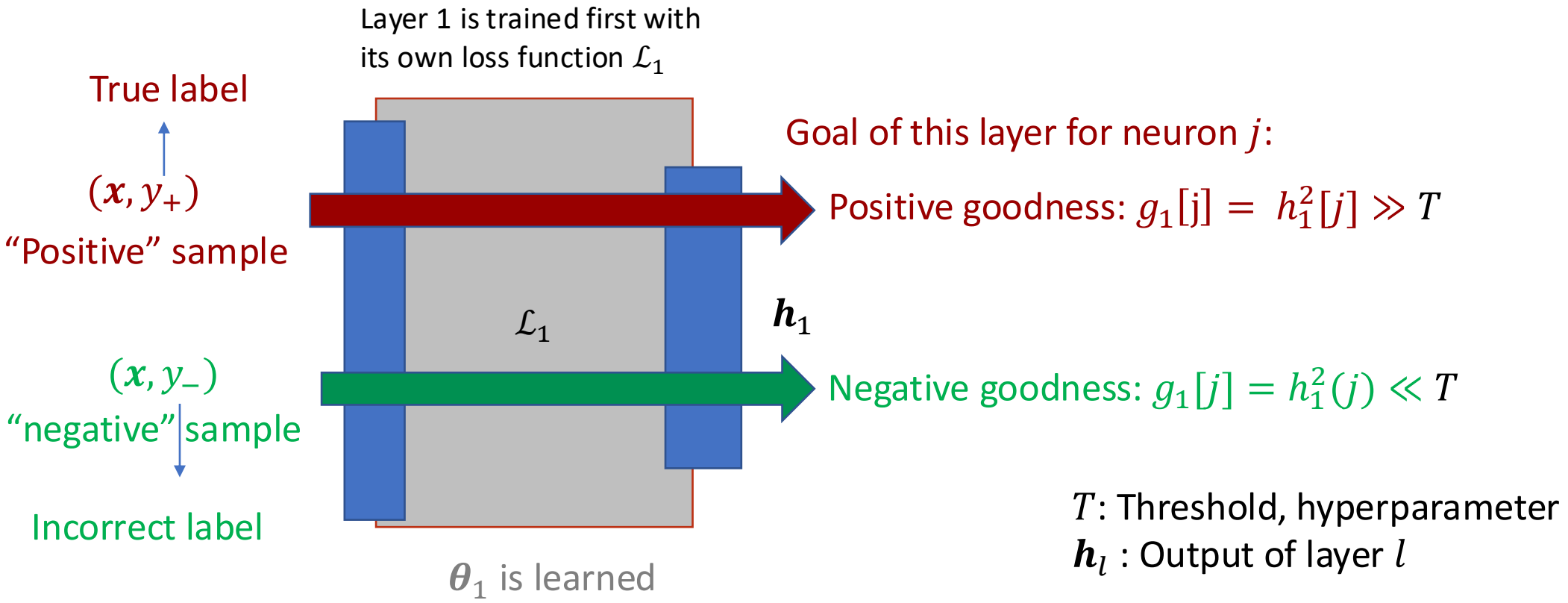


- ✓ Memory-efficient
  - In FF, RAM scales with the size largest layer
  - In backpropagation, RAM scales with depth of the network and layer sizes
- ✓ Simplicity
  - No computational graph nor Autodiff engines are needed

SAMSUNG

LightTune: Detailed Algorithm

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



T: Threshold, hyperparameter

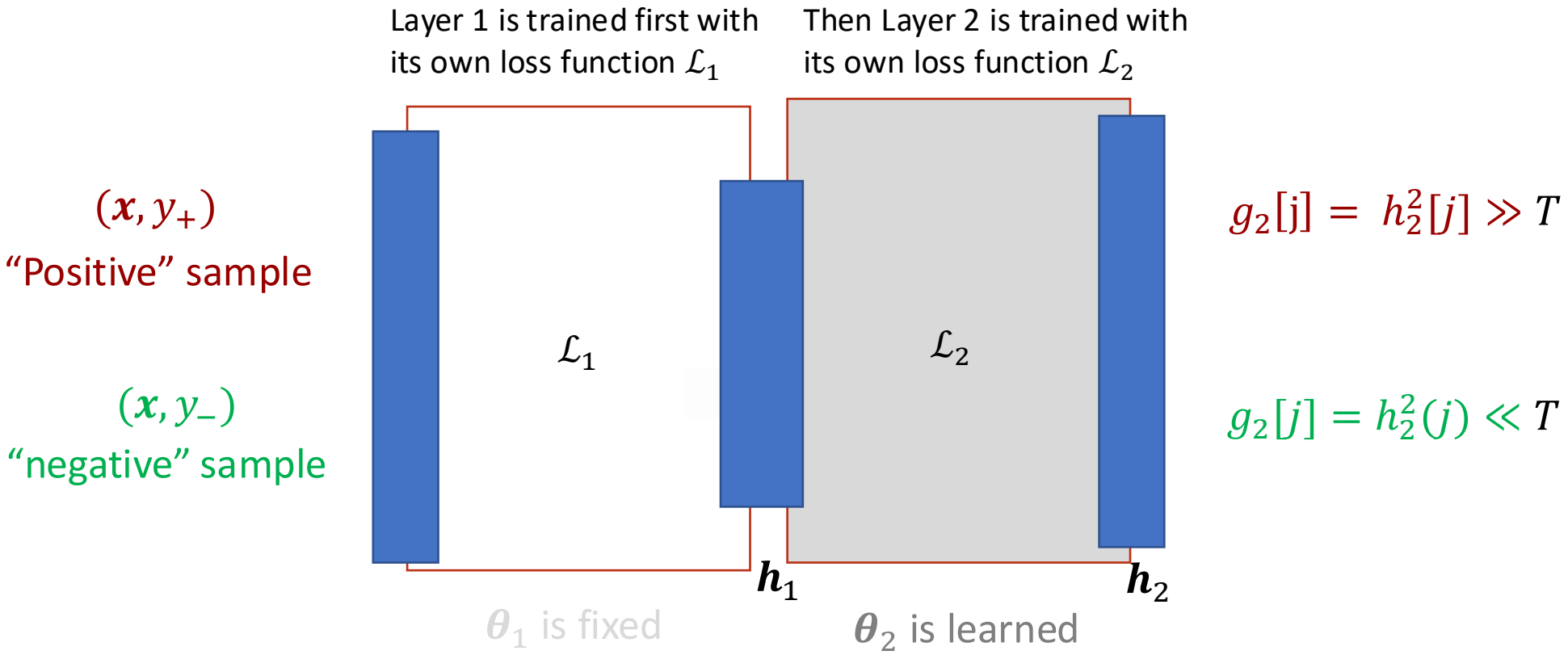
$\boldsymbol{h}_l$: Output of layer l

✓ Training is sequential, layer by layer

SAMSUNG

LightTune: Detailed Algorithm

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

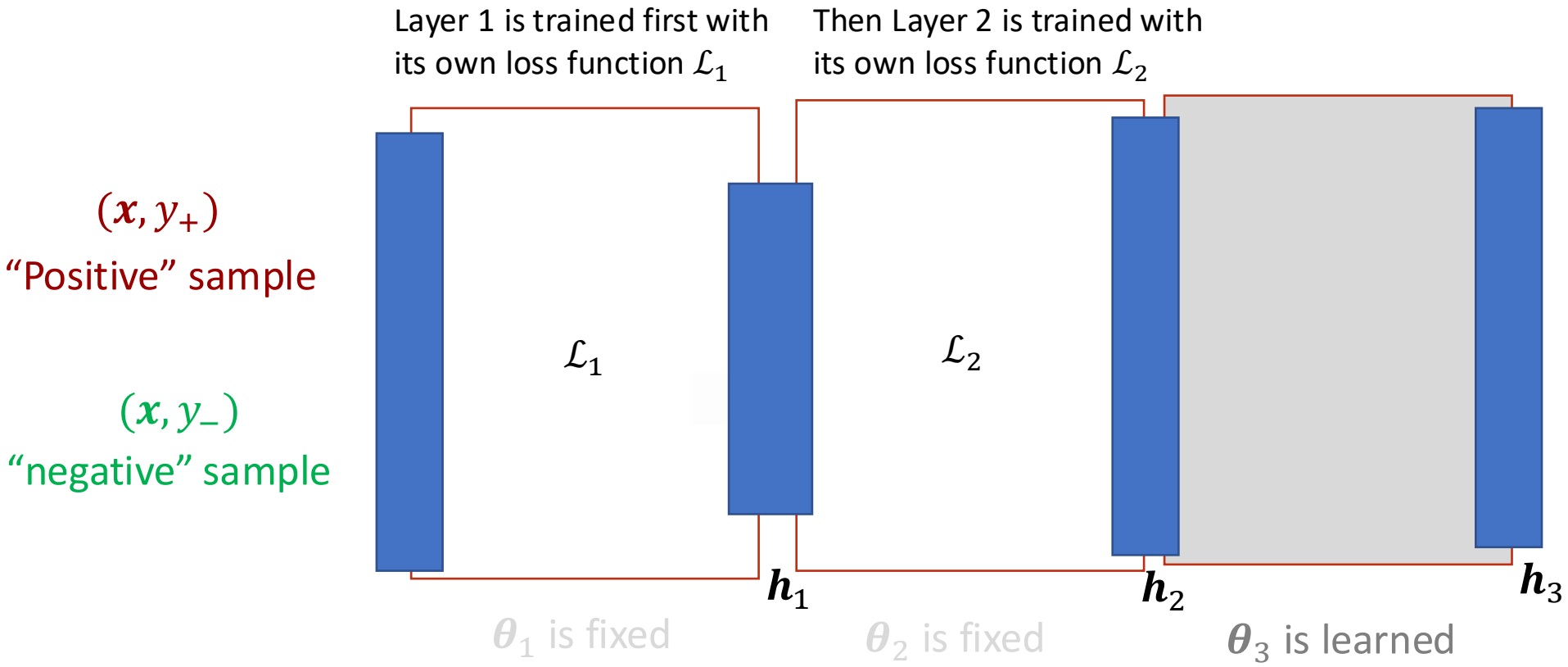


✓ Training is sequential, layer by layer

SAMSUNG

LightTune: Detailed Algorithm

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



✓ Training is sequential, layer by layer

SAMSUNG

LightTune: Proposed Loss Function

1) Backpropagation-free by leveraging the forward-forward (FF) algorithm
   - Softplus loss typically used in the FF algorithm

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

>>

<<

$g_{+,l}[j]$: 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

LightTune: Proposed Loss Function

1) Backpropagation-free by leveraging the forward-forward (FF) algorithm
   - Softplus loss typically used in the FF algorithm

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

>>

<<

Needs exponentiations & divisions to compute derivatives !!

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

LightTune: Proposed Loss Function

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

- Softplus loss typically used in the FF algorithm

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

- To avoid the exponentiations and divisions, we propose an on-device friendly loss

$g_{+,l}[j]$: 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

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

SAMSUNG

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

$$\nabla_{\boldsymbol{W}_l} \mathcal{L}_{\text{Prop},l} = [(2(\boldsymbol{g}_{+,l} - T) - 4) \odot 2\boldsymbol{h}_{+,l} \odot \mathbf{1}(\boldsymbol{x}_{+,l} > 0)]\, \boldsymbol{x}_{+,l}^\top + [(2(\boldsymbol{g}_{-,l} - T) + 4) \odot 2\boldsymbol{h}_{-,l} \odot \mathbf{1}(\boldsymbol{x}_{-,l} > 0)]\, \boldsymbol{x}_{-,l}^\top,$$
$$\nabla_{\boldsymbol{b}_l} \mathcal{L}_{\text{Prop},l} = (2(\boldsymbol{g}_{+,l} - T) - 4) \odot 2\boldsymbol{h}_{+,l} \odot \mathbf{1}(\boldsymbol{x}_{+,l} > 0) + (2(\boldsymbol{g}_{-,l} - T) + 4) \odot 2\boldsymbol{h}_{-,l} \odot \mathbf{1}(\boldsymbol{x}_{-,l} > 0).$$

SAMSUNG

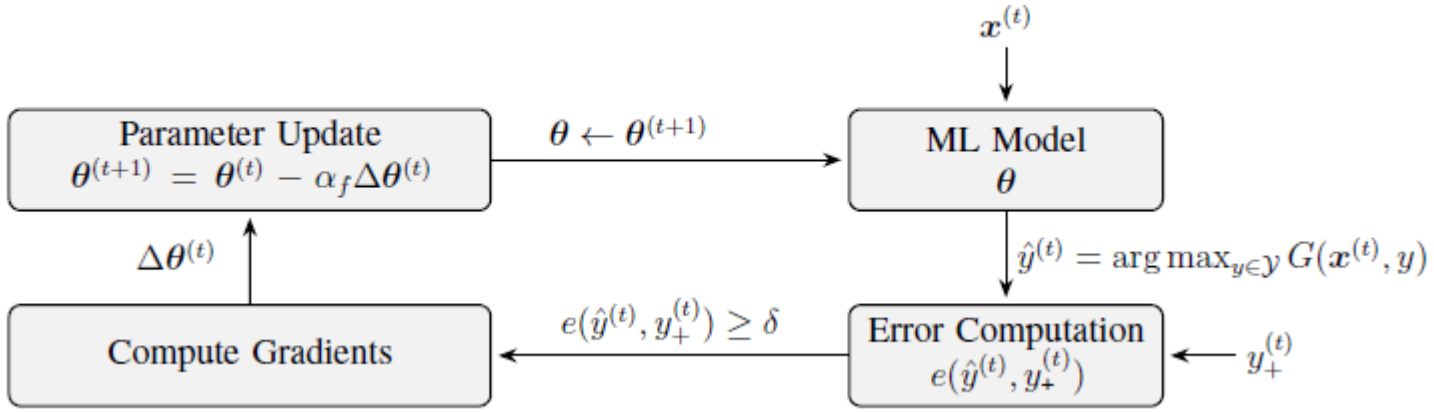
LightTune: Detailed Algorithm

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)



SAMSUNG

LightTune: Convergence

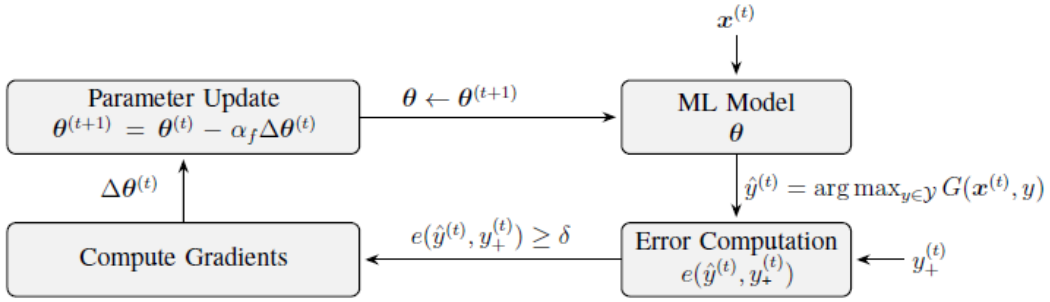
Theorem: Assuming

(1) boundedness of the data and the model,

(2) non-degeneracy,

(3) stationarity,

(4) uniform negative sampling, LightTune ensures that for ReLU MLPs, we have



$$\lim_{t \to \infty} \Pr\left(\left|\hat{y}^{(t)} - y_+^{(t)}\right| \geq \delta\right) = 0$$

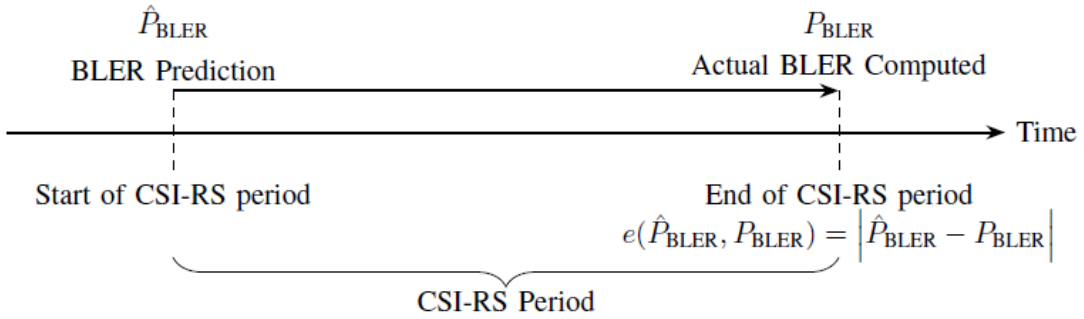
Prediction error

Pre-defined threshold

SAMSUNG

Applications of LightTune: Block Error Rate (BLER) Prediction

- BLER prediction has many application in 5G/6G including link adaptation, PMI selection, …
- Ground-truth can be computed after a delay (i.e., end of the CSI-RS period)



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

SAMSUNG

Applications of LightTune: Block Error Rate (BLER) Prediction

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

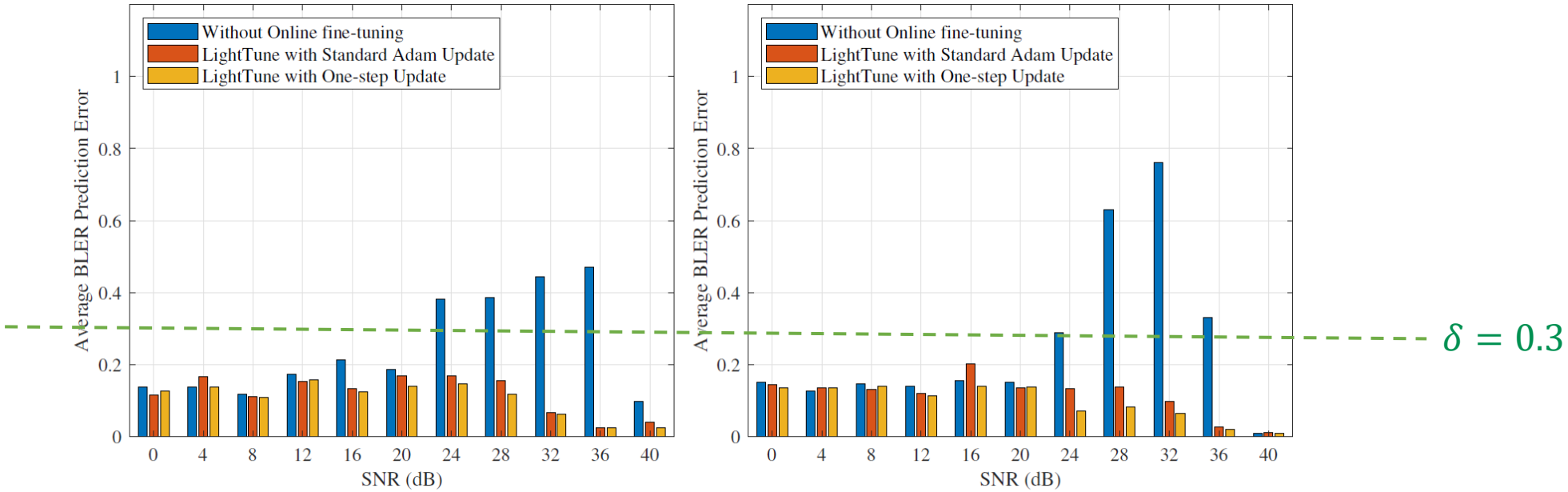
Parameter	Training	Test
Channels	TDL-A30	TDL-A30, TDL-B50, TDL-C200
SNR	Low (0–12 dB)	Low/Medium/High (0–40 dB)
Delay Profile	Low Delay	Low/High Delay
Doppler Frequency	Low (10 Hz)	Low/Medium (10–50 Hz)

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
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	$\lambda = 0.1$

SAMSUNG

Applications of LightTune: Block Error Rate (BLER) Prediction

- Online fine-tuning is a must!



(a) TDL-A30 (Doppler freq. = 10 Hz) (b) TDL-B50 (Doppler freq. = 30 Hz)

SAMSUNG

Applications of LightTune: Link Adaptation in 5G/6G

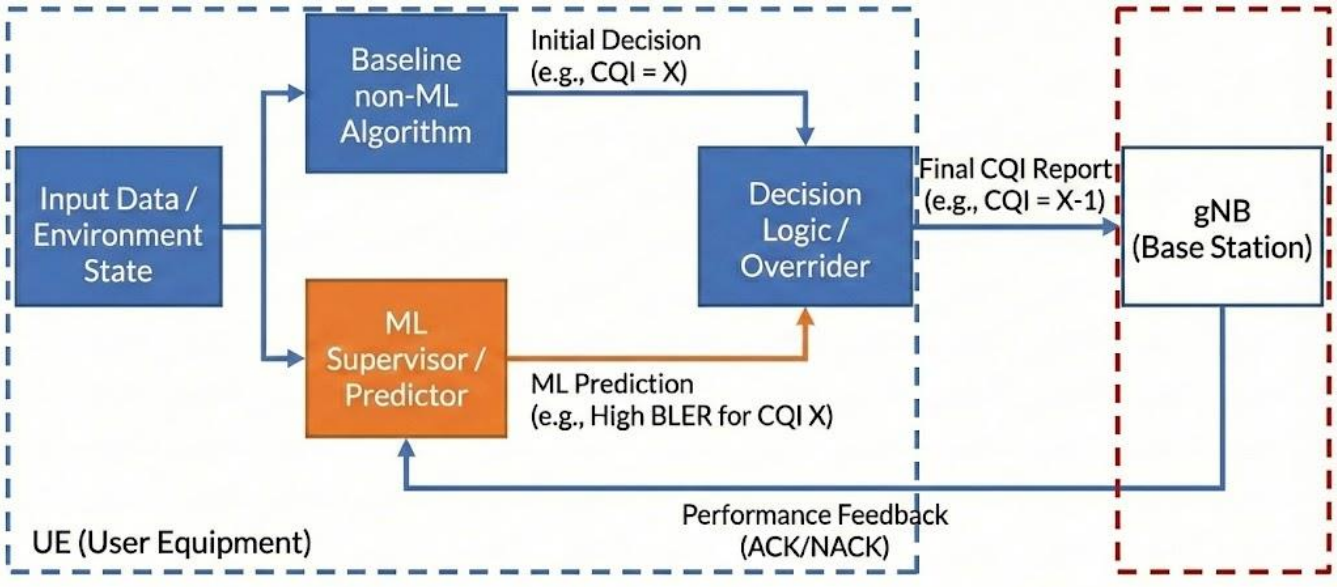
- Conventional table-based link adaption algorithms [e.g., Peralta‘22] are slow to converge
  - Initially, they may select CQIs with high BLER

E. Peralta et al., “Outer loop link adaptation enhancements for ultra reliable low latency communications in 5G,” in IEEE VTC, 2022.

SAMSUNG

Applications of LightTune: Link Adaptation in 5G/6G

- Conventional table-based link adaption algorithms [e.g., Peralta‘22] are slow to converge
  - Initially, they may select CQIs with high BLER
- To mitigate this high BLER, we use our BLER prediction algorithm as a backoff mechanism.

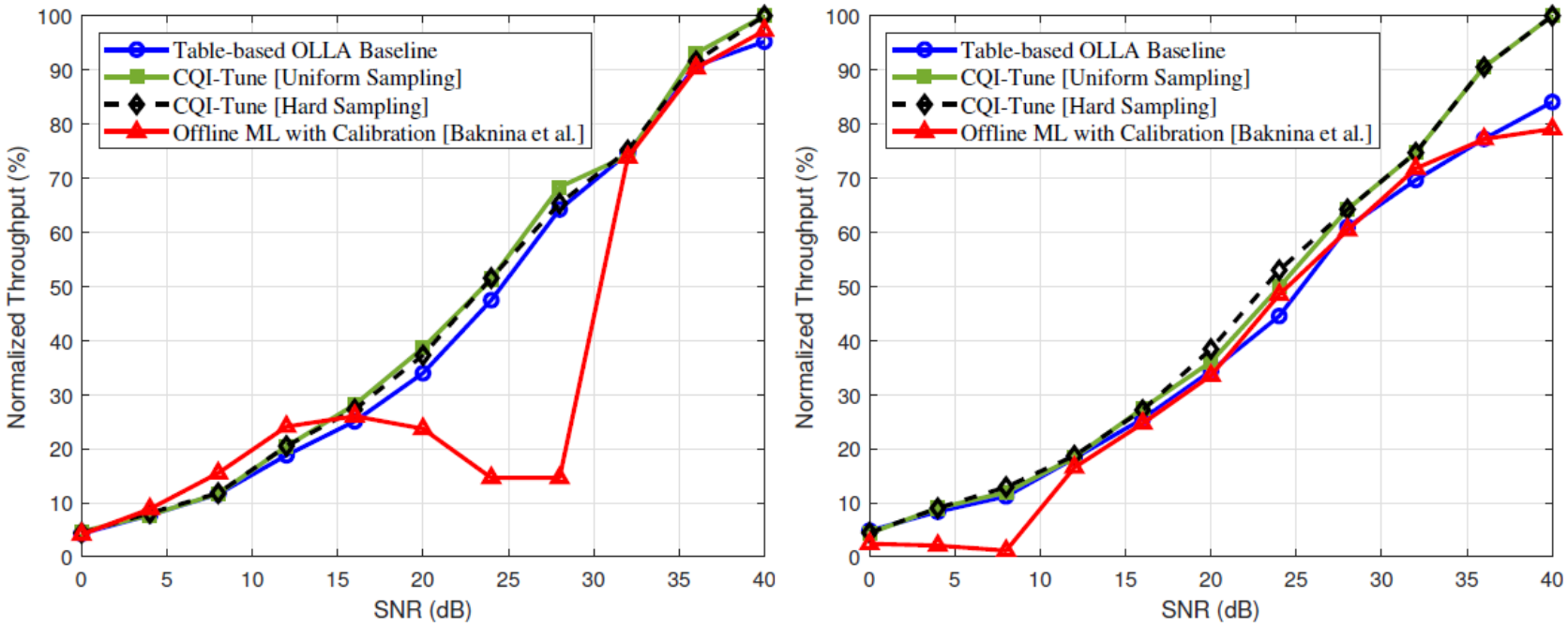


E. Peralta et al., “Outer loop link adaptation enhancements for ultra reliable low latency communications in 5G,” in IEEE VTC, 2022.

SAMSUNG

Applications of LightTune: Link Adaptation in 5G/6G

- Based on the BLER prediction algorithm, we propose a CQI selection algorithm.



(a) TDL-B50 (Dop. freq. = 30 Hz). (b) TDL-C200 (Dop. freq. = 50 Hz)

Up to 9.5% throughput improvement

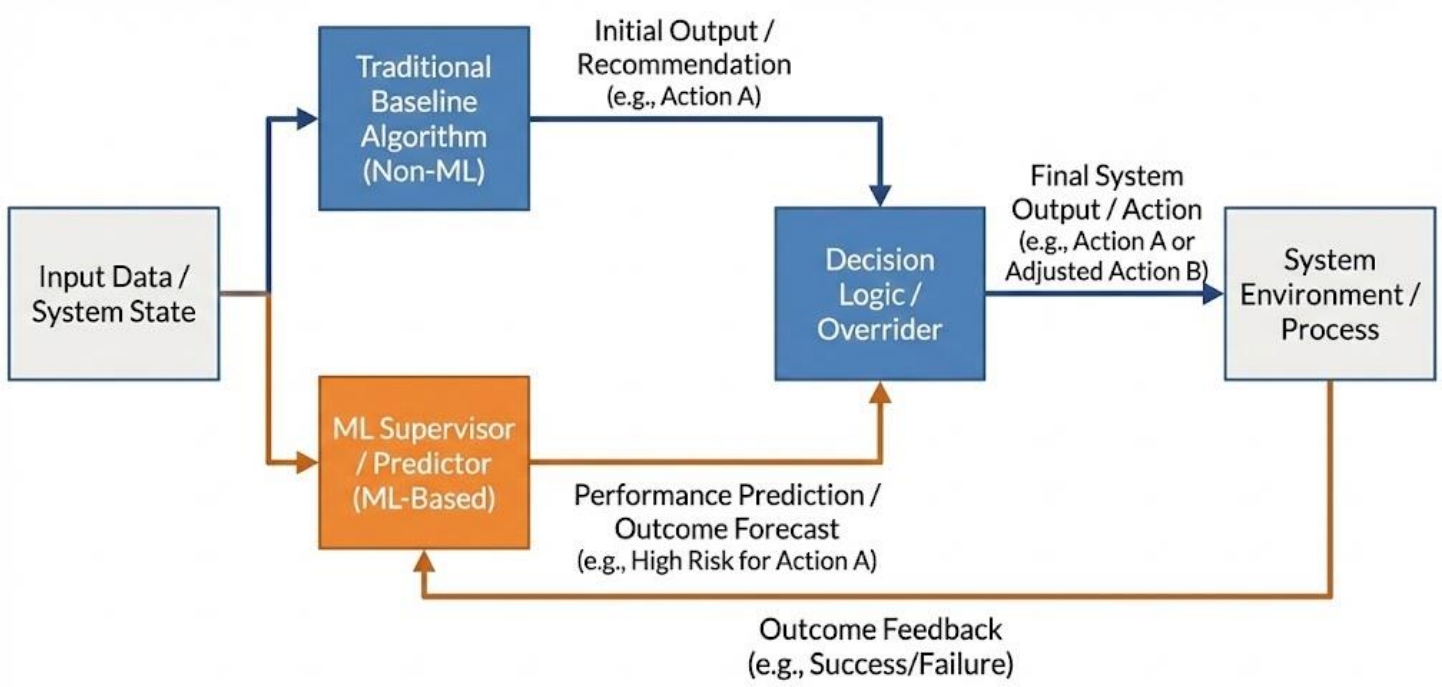
SAMSUNG

Takeaways & Future Work

1. Online fine-tuning is essential for ML-based wireless algorithms
2. We showed case for CQI selection, but LightTune can be applied to other problems
   - Rank Selection
   - PMI Selection
   - …..

Takeaways & Future Work

3. 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)



SAMSUNG

Thank you
Questions?