Stepping Forward on the Last Mile

Chen Feng, Shaojie Zhuo, Xiaopeng Zhang,

Ramchalam Kinattinkara Ramakrishnan, Zhaocong Yuan, Andrew Zou Li

Qualcomm AI Research[1], *Qualcomm Canada ULC*

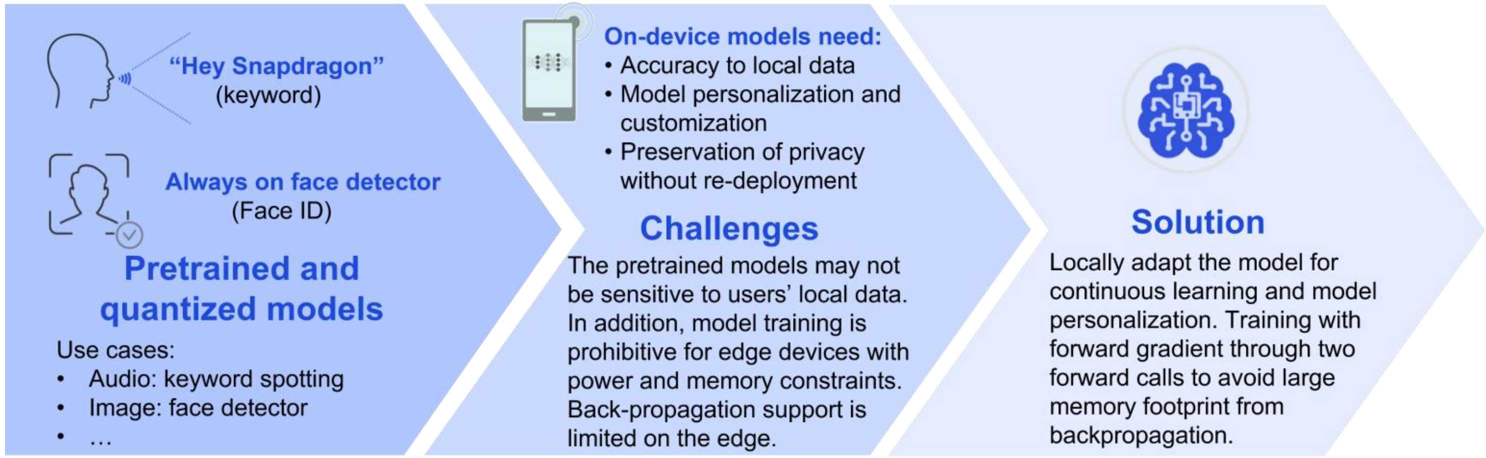
[1] *Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.*

Qualcomm

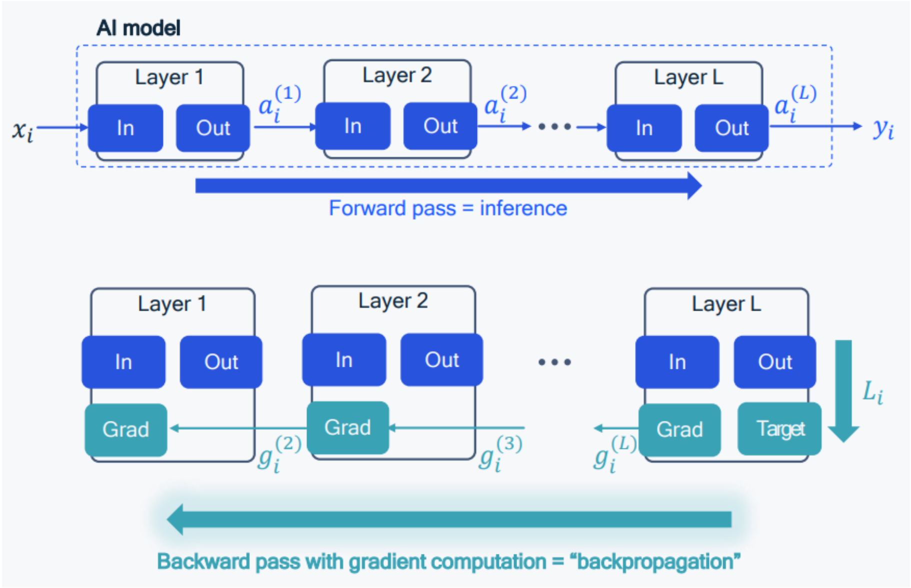
Stepping Forward on the Last Mile

- Motivation
- Methodology
- Quantized Training
- Experimental Results
- Conclusion

Motivation

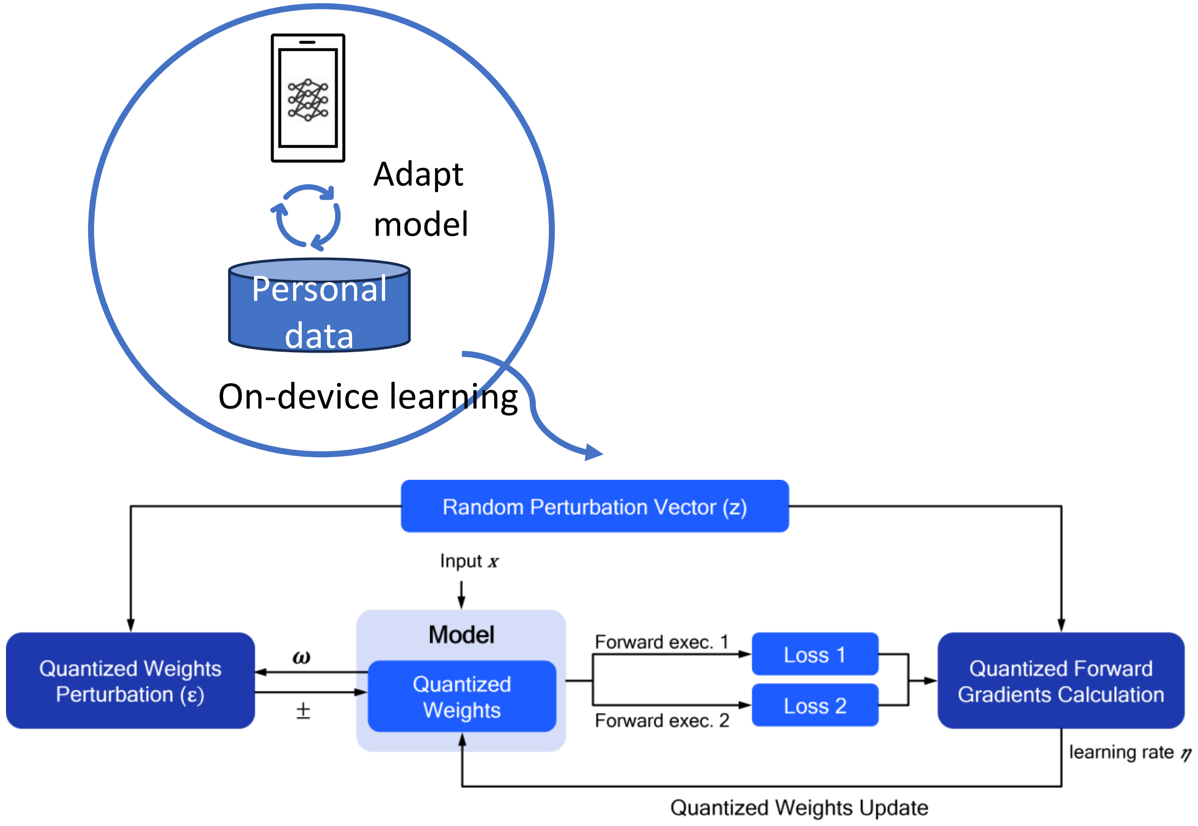


"Hey Snapdragon"
(keyword)
Always on face detector
(Face ID)
Pretrained and quantized models
Use cases:
• Audio: keyword spotting
• Image: face detector
• …
On-device models need:
• Accuracy to local data
• Model personalization and customization
• Preservation of privacy without re-deployment
Challenges
The pretrained models may not be sensitive to users' local data. In addition, model training is prohibitive for edge devices with power and memory constraints. Back-propagation support is limited on the edge.
Solution
Locally adapt the model for continuous learning and model personalization. Training with forward gradient through two forward calls to avoid large memory footprint from backpropagation.



AI model
Layer 1
Layer 2
Layer L
In
Out
x_i
$a_i^{(1)}$
$a_i^{(2)}$
$a_i^{(L)}$
y_i
Forward pass = inference
Grad
Target
$g_i^{(2)}$
$g_i^{(3)}$
$g_i^{(L)}$
L_i
Backward pass with gradient computation = "backpropagation"

Methodology



Model adaptation through fixed-point forward-forward (FF) gradient learning. Forward gradients are estimated through **forward calls only**, without the need of backpropagation.

Training without back-propagation

Definition: Given a machine learning function $f(w): \mathcal{R}^n \to \mathcal{R}$ and model parameters $w \in \mathcal{R}^n$, with perturbation vector $z \in \mathcal{R}^n$, the ***forward gradient*** $g: \mathcal{R}^n \to \mathcal{R}^n$ is defined as a directional derivative of f at point w in direction z :

$$g(w) = (\nabla f(w) \cdot z)z \tag{1}$$

Definition (SPSA): Given a model f with parameters $w \in \mathcal{R}^n$ and a loss function $\mathrm{L}(w)$, SPSA estimates the gradient as:

$$\hat{g}(w) = \frac{L(w+\varepsilon z) - L(w-\varepsilon z)}{2\varepsilon} z \tag{2}$$

where $z \sim N(0, I_n)$ is a weighted vector over all parameter dimensions, randomly sampled from normal distribution with zero-mean and standard deviation.

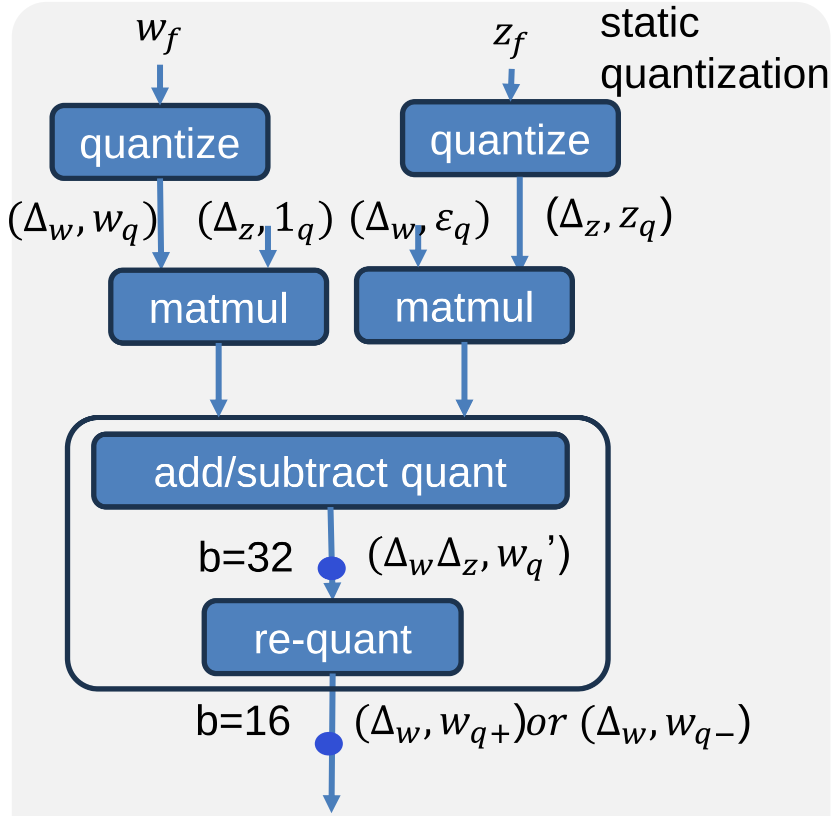
Definition (Sign-m-SPSA):

$$\hat{g}(w) = \frac{1}{m}\sum_{i=1}^{m} sign(\,L(w+\varepsilon z) - L(w-\varepsilon z))\, z_i \tag{3}$$

Definition (Sign-m-SPSA-SGD): With $\hat{g}(w)$ as the estimated forward gradient, an optimizer such as SGD with learning rate η can be used to update model parameters:

$$w_{t+1} = w_t - \eta\, \hat{g}(w) \tag{4}$$

Quantized Training



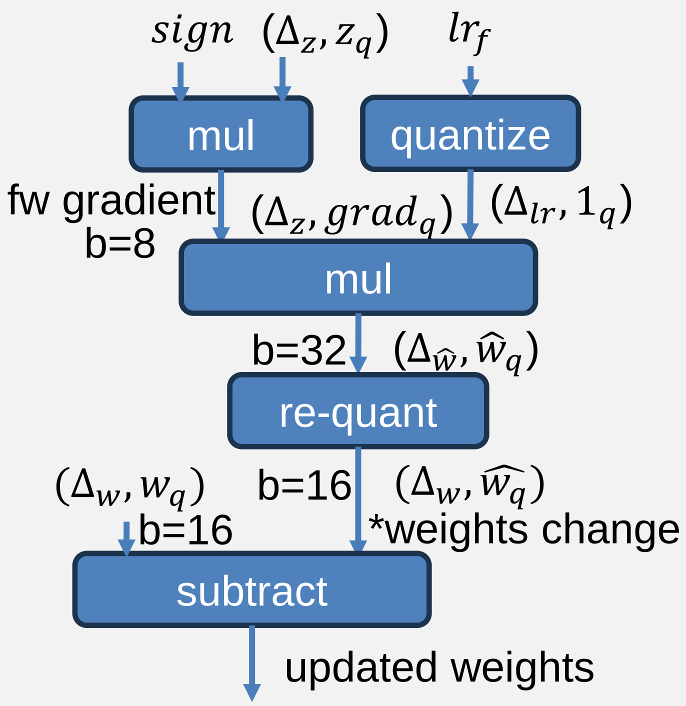
1. weights perturbation

Quantized Perturbation: the quantized weights perturbation can be defined and calculated as:

$$\begin{aligned} w \pm \epsilon z &= w \cdot 1.0 \pm \epsilon z \\ &\approx \Delta_w w_q \cdot \Delta_z \mathbf{1}_q \pm \Delta_w \epsilon_q \cdot \Delta_z z_q \\ &= \Delta_w \Delta_z (w_q \cdot \mathbf{1}_q \pm \epsilon_q \cdot z_q) \overset{re-quant}{\Longrightarrow} \Delta_w \cdot w_{q\pm} \end{aligned} \quad (5)$$

where $\mathbf{1}_q = \lfloor \frac{1.0}{\Delta_z} \rceil$, represents for the quantized value of floating point 1.0 with Δ_z as its scaling factor. Similarly, $\epsilon_q = \lfloor \frac{\epsilon}{\Delta_w} \rceil$, represents for the quantized value of ε with Δ_w as its scaling factor.

Quantized Training



Quantized forward gradients and quantized weight update

Quantized Forward Gradients: the quantized forward gradient, estimated from sign-m-SPSA can be calculated as:

$$
\begin{aligned}
\hat{g}_f &= \frac{1}{m}\sum_{i=1}^{m} sign(\mathbb{L}(w+\epsilon z_i) - \mathbb{L}(w-\epsilon z_i))z_i \\
&\approx \frac{1}{m}\sum_{i=1}^{m} sign(\mathbb{L}(w_{q^+}) - \mathbb{L}(w_{q^-}))\Delta_z z_q \\
&= \Delta_z g_q
\end{aligned} \tag{6}
$$

where g_q represents for the quantized gradients, and it is using the same quantization scaling factor and bit-width as perturbation vector z.

Quantized Weights Update: we can further quantize the learning rate η to a quantized value of 1 , and the change of weights can be derived in the quantized space, with Δ_w as the re-quantized scaling factor.

$$
\begin{aligned}
w_{t+1} &= w_t - \eta \hat{g}_f \\
&\approx \Delta_w w_q - \Delta_\eta 1 \Delta_z g_q \\
&\approx \Delta_w w_q - \Delta_w \lfloor \frac{\Delta_\eta \Delta_z}{\Delta_w} g_q \rceil \\
&= \Delta_w (w_q - \bar{w}_q)
\end{aligned} \tag{7}
$$

QZO-FF enhancement

Algorithm 1 QZO-FF: Quantized Zero-order Forward Gradient Learning(quantized, fp16)

Require: quantized model parameters $w_q \in I^n$, loss $L : I^n \rightarrow R$, perturbation scale ϵ, training steps T, batch size B, learning rate schedule $\{\eta_t\}$

- Given a pre-defined z_{max} of perturbation z, calculate $\Delta_z = z_{max}/(2^{b-1} - 1)$ with b-bit.
- Quantize 1.0 to 1_q with Δ_z.
- Get the quantization scaling factor, Δ_{w^i}, of quantized weights of each layer.

for $t = 1, ..., T$ **do**
  for $m=1, ..., M$ **do**
    Sample random seed s, and batch B
    Generate perturbation vector $z \sim N(0, I_n)$, and quantize the values to (Δ_z, z_q), $z_q \in I^n$
    $w_{q^+} \leftarrow PerturbParameters(w_q, z_q, \epsilon_q)$ ▷Perturb in positive direction
    $l_+ \leftarrow L(w_{q^+}; B)$
    $w_- \leftarrow PerturbParameters(w_q, z_q, -2\epsilon_q)$ ▷Perturb in negative direction
    $l_- \leftarrow L(w_{q^-}; B)$
    $g_q^a \mathrel{+}= sign(l_+ - l_-) \cdot z_q$ ▷Quantized gradient accumulation
    $w_q \leftarrow PerturbParameters(w_q, z_q, \epsilon_q)$ ▷Reset weights to original position
  end for
  $g_q = g_q^a / M$ ▷Quantized gradient averaging
  for $w_q^i \in w_q$ **do** ▷Update weights of each layer
    $\bar{w}_q^i = \lfloor \frac{\Delta_\eta \Delta_z}{\Delta_{w^i}} g_q \rceil$ ▷Re-quantization (see Append.A for fixed-point approximation)
    $w_q^i \leftarrow w_q^i - \bar{w}_q^i$
  end for
end for

Subroutine: $PerturbParameters(w_q, z_q, \epsilon_q)$
for $w_q^i \in w_q$ **do**
  $w_q^i \leftarrow \lfloor \Delta_z (w_q^i \cdot 1_q + \epsilon_q \cdot z_q) \rceil$, where $\epsilon_q = \lfloor \epsilon / \Delta_{w^i} \rceil$ ▷per-tensor Δ_{w^i}
end for

- Momentum Guided Sampling
- Sharpness-aware Perturbation
- Sparse Update
- Kernel-wise Normalization

Experimental Results: Few-shot Learning

Vision tasks:

- 5 datasets
- 3 network architectures
- 5-way 5-shot setting

Table 1: Vision tasks: few-shot learning accuracy (%) with Forward (FF) and Backward (BP) gradients. The averaged accuracy over 100 testing tasks is reported. FT: full fine-tuning; LP: linear probing; Quant: 16w8a with symmetric quantization. FF outperforms zero-shot across the board, and achieves comparable performance (accuracy within 5%) to BP on 26 out of 30 tasks.

Backbone	Training	CUB	Omniglot	Cifar100_fs	miniImageNet	tieredImageNet
Resnet12	Zero-shot	68.46	92.00	60.44	84.44	80.92
	BP, FT	**85.32**	**99.62**	**82.32**	87.34	**82.54**
	BP, LP	84.14	98.64	72.42	**87.46**	81.96
	FF, FT	80.58 (-4.74)	97.44 (-2.18)	71.24 (-11.08)	87.36 (+0.02)	82.12 (-0.42)
	FF, LP	79.02 (-5.12)	96.62 (-2.02)	70.30 (-2.12)	87.30 (-0.16)	82.22 (+0.26)
	FF, LP, Quant	77.42	96.08	68.54	87.00	81.64
Resnet18	Zero-shot	59.96	86.68	74.60	82.58	80.44
	BP, FT	**79.28**	**98.54**	**86.34**	86.96	**86.78**
	BP, LP	78.92	96.48	84.88	87.42	84.68
	FF, FT	76.34 (-5.64)	94.70 (-3.84)	82.20 (-4.14)	**87.66** (+0.70)	85.88 (-0.90)
	FF, LP	73.64 (-5.28)	95.56 (-0.92)	82.32 (-2.56)	87.14 (+0.32)	83.02 (-1.66)
	FF, LP, Quant	70.54	95.86	74.92	85.74	81.00
ViT tiny	Zero-shot	90.60	90.96	82.28	98.78	94.30
	BP, FT	93.08	**99.88**	**90.88**	98.46	**96.04**
	BP, LP	93.90	95.78	84.42	98.40	95.32
	FF, FT	**93.58** (+0.50)	96.96 (-2.92)	88.66 (-2.22)	**99.08** (+0.62)	95.50 (-0.54)
	FF, LP	92.26 (-1.64)	95.00 (-0.78)	84.48 (+0.06)	99.02 (+0.62)	95.18 (-0.14)
	FF, LP, Quant	92.24	95.04	84.40	99.00	95.18

Audio tasks:

- 2 datasets
- 2 network architectures
- 5-way 1-shot setting

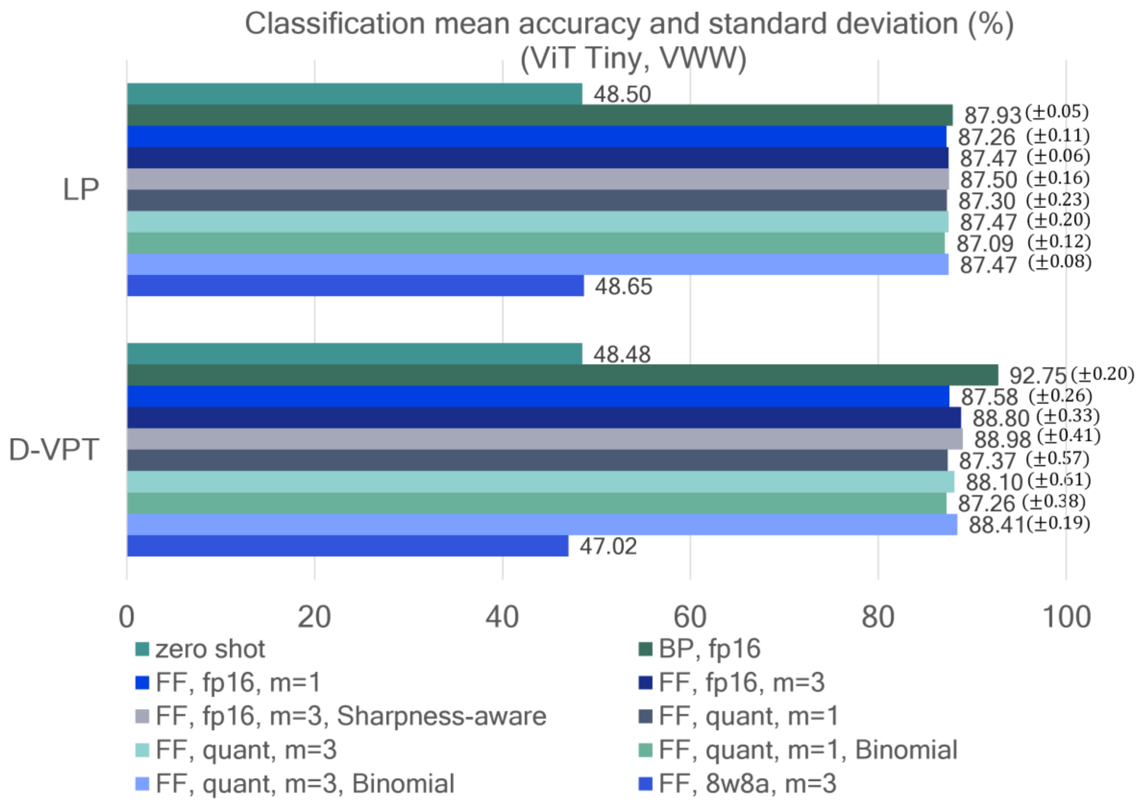
Table 2: Audio tasks: few-shot learning accuracy (%) with Forward (FF) and Backward (BP) gradients. FF achieves comparable (accuracy within 5%) or better performance to BP on 11 out of 16 tasks.

Backbone	Training	ESC- 50		FSDKaggle18	
		SimpleShot	ProtoNet	SimpleShot	ProtoNet
CRNN	BP, FT	66.34	**73.82**	**38.89**	33.11
	BP, LP	**72.11**	71.30	36.88	32.67
	FF, FT	67.20 (+0.86)	64.30 (-11.39)	36.04 (-2.85)	35.52 (+2.41)
	FF, LP	67.38 (-4.73)	61.62 (-9.68)	37.53 (+0.65)	34.67 (+2.00)
	FF, LP, Quant	67.05	63.43	36.90	**35.55**
AST	BP, FT	68.04	**75.85**	38.12	**46.12**
	BP, LP	75.98	70.16	42.86	42.64
	FF, FT	**79.70** (+11.66)	66.98 (-8.87)	**42.92** (+4.80)	40.50 (-5.62)
	FF, LP	76.07 (+0.09)	63.96 (-6.20)	42.72 (-0.14)	38.18 (-4.46)
	FF, LP, Quant	76.13	61.86	42.90	38.10

Experimental Results: Cross-domain Adaptation

Cross-domain Adaptation

- Adapted dataset largely differs from pre-trained dataset
- ViT tiny backbone
- Ablation studies on
    - Two training methods (LP, D-VPT)
    - Effectiveness of quantized FF
    - Gradient averaging in FF
    - Quantization bit-width
    - Perturbation sampling
    - QZO-FF enhancement



Experimental Results: In-domain OOD Adaptation

In-domain OOD Adaptation

- Adapted dataset is similar to the pre-trained dataset, but with data out of distribution (OOD)
- ViT tiny backbone
- Ablation studies on
  - Two training methods (LP, D-VPT)
  - Effectiveness of quantized FF
  - Effectiveness of sparse FF

Table 3: Accuracy (%) of model adaptation to in-domain OOD dataset with Forward (FF) and Backward (BP) gradients. 1 LN: 1 linear layer of decoder; 3 LN: 3 linear layer of decoder. Quant: 16w8a, Sparse: 90% weights pruned. The accuracy numbers (with standard deviation) are averaged over 5 runs.

Backbone	Training	Cifar10-C (easy)	Cifar10-C (median)	Cifar10-C (hard)
LP 1 LN	Zero-shot	82.48	74.59	62.40
	BP	83.75 (± 0.67)	77.88 (± 0.85)	70.03 (± 1.20)
	FF	83.37 (± 0.60)	77.04 (± 0.66)	68.65 (± 0.70)
	FF, Sparse	83.34 (± 0.59)	77.11 (± 0.68)	68.63 (± 0.95)
	FF, Quant	83.23 (± 0.57)	76.73 (± 0.75)	68.28 (± 0.87)
LP 3 LN	Zero-shot	85.83	77.77	62.25
	BP	86.99 (± 0.41)	81.57 (± 0.78)	74.76 (± 0.90)
	FF	86.11 (± 0.59)	79.17 (± 0.70)	67.78 (± 0.72)
	FF, Sparse	86.10 (± 0.58)	79.24 (± 0.63)	68.06 (± 1.11)
	FF, Quant	85.77 (± 0.55)	78.67 (± 0.63)	67.25 (± 0.42)
D-VPT	Zero-shot	89.52	82.24	68.95
	BP	91.66 (± 0.50)	88.90 (± 0.46)	84.54 (± 0.42)
	FF	90.58 (± 0.53)	86.21 (± 0.49)	78.38 (± 0.80)
	FF, Sparse	90.56 (± 0.48)	86.18 (± 0.51)	78.24 (± 0.81)
	FF, Quant	90.41 (± 0.49)	85.77 (± 0.43)	77.45 (± 0.64)

Conclusion

- Continuously updating pre-trained models to local data on the edge is the last mile for model adaptation and customization.
- To overcome the memory limitation of most existing low power devices, forward gradients can be used for model fine-tuning.
- Through comprehensive experiments, we have shown that quantized forward gradient learning with 16w8a can effectively adapt most typical model architectures (e.g., Resnet, ViT-tiny, CRNN, AST) and scales.
- With minimum accuracy reduction, fixed-point forward gradients allows model adaptation using the same memory footprint and operation support as inference, as opposed to backpropagation.
- Therefore, it has the potential to enable model fine-tuning on existing edge devices with limited memory and backpropagation support, without requiring additional hardware adaptation.

Thank You