A Appendix

The organization of the Appendix is as follows: Appendix sections B to F provide detailed proofs of the lemmas and theorems presented in the main text. This is followed by additional insights, including operational specifics of working with Adam and details on how AsyncFGD can be extended for parallel processing on recursive networks, covered in Appendix G. Lastly, Appendix H and I provide a comprehensive view of the training process and additional experimental results, respectively.

B Proof of Lemma 3.1

According to the update rule of Eq.(3), (4), we have

$$
\begin{aligned}
h_1 =& F_1(h_0, w_1) = F_1(x, w_1) \\
o_1 =& J_{F_1}(h_0, w_1)[o_0^\intercal, u_{w_1}^\intercal]^\intercal = J_{F_1}(h_0)o_0 + J_{F_1}(w_1)u_{w_1} = J_{F_1}(w_1)u_{w_1} \\
h_2 =& F_2(h_1, w_2) \\
o_2 =& J_{F_2}(h_1, w_2)[o_1^\intercal, u_{w_2}^\intercal]^\intercal = J_{F_2}(h_1)o_1 + J_{F_2}(w_2)u_{w_2} = J_{F_2}(h_1)J_{F_1}(w_1)u_{w_1} + J_{F_2}(w_2)u_{w_2} \\
=& J_{F_2}(w_1)u_{w_1} + J_{F_2}(w_2)u_{w_2} \\
& \cdots \\
o_L =& \sum_{l=1}^{L} J_{F_L}(w_l)u_{w_l}
\end{aligned}
$$

Then for any $l \in \{1, 2, \ldots, L\}$, take expectation with respect to u_{w_l}, we have

$$
\mathbb{E}_{u_{w_l}}(o_L \cdot u_{w_l}) = \mathbb{E}_{u_{w_l}}\left[(J_{F_L}(w_l)u_{w_l}) \cdot u_{w_l} + \sum_{k \neq l}(J_{F_L}(w_k)u_{w_k}) \cdot u_{w_l}\right]
$$

Note that

$$
\begin{aligned}
\mathbb{E}_{u_{w_l}}\left[(J_{F_L}(w_l)u_{w_l}) \cdot u_{w_l}\right] =& \mathbb{E}_{u_{w_l}}\left(\begin{bmatrix} \frac{\partial J_{F_L,1}}{\partial w_{l,1}} & \frac{\partial J_{F_L,1}}{\partial w_{l,2}} & \cdots & \frac{\partial J_{F_L,1}}{\partial w_{l,d_l}} \\ \frac{\partial J_{F_L,2}}{\partial w_{l,1}} & \frac{\partial J_{F_L,2}}{\partial w_{l,2}} & \cdots & \frac{\partial J_{F_L,2}}{\partial w_{l,d_l}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial J_{F_L,d_{h_L}}}{\partial w_{l,1}} & \frac{\partial J_{F_L,d_{h_L}}}{\partial w_{l,2}} & \cdots & \frac{\partial J_{F_L,d_{h_L}}}{\partial w_{l,d_l}} \end{bmatrix} \begin{bmatrix} u_{w_l,1} \\ u_{w_l,2} \\ \vdots \\ u_{w_l,d_l} \end{bmatrix} \cdot \begin{bmatrix} u_{w_l,1} & u_{w_l,2} & \cdots u_{w_l,d_l} \end{bmatrix}\right) \\
=& \mathbb{E}_{u_{w_l}}\left(\begin{bmatrix} \sum_{i=1}^{d_l} \frac{\partial J_{F_L,1}}{\partial w_{l,i}} \cdot u_{w_{l,i}} \\ \sum_{i=1}^{d_l} \frac{\partial J_{F_L,2}}{\partial w_{l,i}} \cdot u_{w_{l,i}} \\ \vdots \\ \sum_{i=1}^{d_l} \frac{\partial J_{F_L,d_{h_L}}}{\partial w_{l,i}} \cdot u_{w_{l,i}} \end{bmatrix} \cdot \begin{bmatrix} u_{w_l,1} & u_{w_l,2} & \cdots u_{w_l,d_l} \end{bmatrix}\right) \\
=& \mathbb{E}_{u_{w_l}}(D),
\end{aligned}
$$

where

$$
D_{m,n} = \left(\sum_{i=1}^{d_l} \frac{\partial J_{F_L,m}}{\partial w_{l,i}} \cdot u_{w_{l,i}}\right) u_{w_{l,n}} = \frac{\partial J_{F_L,m}}{\partial w_{l,n}} u_{w_{l,n}}^2 + \sum_{k \neq n} \frac{\partial J_{F_L,m}}{\partial w_{l,k}} u_{w_{l,k}} u_{w_{l,n}}
$$

with $m \in \{1, 2, \ldots, d_{h_L}\}, n \in \{1, 2, \ldots, d_l\}$. Since each $u_{w_l} \sim \mathbb{N}(0, I)$, we have

$$
\mathbb{E}_{u_{w_l}}(D_{m,n}) = \frac{\partial J_{F_L,m}}{\partial w_{l,n}}, \quad \mathbb{E}_{u_{w_l}}(D) = J_{F_L}(w_l).
$$

Similarly, we can prove that $\mathbb{E}_{u_{w_l}}\left[\sum_{k \neq l}(J_{F_L}(w_k)u_{w_k}) \cdot u_{w_l}\right] = 0 \in \mathbb{R}^{d_{h_L} \times d_l}$. So we have,

$$
\mathbb{E}\left(\frac{\partial f}{\partial F_L} \cdot o_L \cdot u_{w_l}\right) = \frac{\partial f}{\partial F_L} J_{F_L}(w_l) = \nabla_{l,x} f(w) \in \mathbb{R}^{d_l}
$$

C Proof of Lemma 5.3

Proof of Mean.

$$\begin{aligned}
\hat{h}_1^{t-K+1} &= F_1(h_0^{t-K+1}, w_1^{t-2K+2}) = F_1(x_{i(t-K+1)}, w_1^{t-2K+2}) \\
\hat{o}_1^{t-K+1} &= J_{F_1}(h_0^{t-K+1}, w_1^{t-2K+2})[o_0^\intercal, u_{w_1}^{t-K+1\intercal}]^\intercal = J_{F_1}(h_0^{t-K+1})o_0 + J_{F_1}(w_1^{t-2K+2})u_{w_1}^{t-K+1} \\
&= J_{F_1}(w_1^{t-2K+2})u_{w_1}^{t-K+1} \\
\hat{h}_2^{t-K+1} &= F_2(\hat{h}_1^{t-K+1}, w_2^{t-2K+2}) \\
\hat{o}_2^{t-K+1} &= J_{F_2}(h_1^{t-K+1}, w_2^{t-K+1})[o_1^{t-K+1\intercal}, u_{w_2}^{t-K+1\intercal}]^\intercal = J_{F_2}(h_1^{t-K+1})o_1^{t-K+1} + J_{F_2}(w_2^{t-2K+2})u_{w_2}^{t-K+1} \\
&= J_{F_2}(h_1^{t-K+1})J_{F_1}(w_1^{t-2K+2})u_{w_1}^{t-K+1} + J_{F_2}(w_2)u_{w_2^{t-2K+2}}^{t-K+1} \\
&\cdots\cdots \\
\hat{o}_{L_0}^{t-K+1} &= \sum_{l=1}^{L_0} J_{f_0}(w_l^{t-2K+2})u_{w_l}^{t-K+1} = \sum_{l\in\mathcal{G}(0)} J_{f_0}(w_l^{t-2K+2})u_{w_l}^{t-K+1} \\
&\cdots\cdots \\
\hat{o}_{L_t}^{t-K+1} &= \sum_{l\in\mathcal{G}(0)} J_{f_t}(w_l^{t-2K+2})u_{w_l}^{t-K+1} + \sum_{l\in\mathcal{G}(1)} J_{f_t}(w_l^{t-2K+3})u_{w_l}^{t-K+1} + \cdots + \sum_{l\in\mathcal{G}(t)} J_{f_t}(w_l^{t-2K+t+2})u_{w_l}^{t-K+1} \\
&= \sum_{k=0}^{t}\sum_{l\in\mathcal{G}(k)} J_{f_t}(w_l^{t-2K+k+2})u_{w_l}^{t-K+1} \\
&\cdots\cdots \\
\hat{o}_{L_{K-1}}^{t-K+1} &= \sum_{k=0}^{K-1}\sum_{l\in\mathcal{G}(k)} J_{f_{K-1}}(w_l^{t-2K+k+2}) = \sum_{k=0}^{K-1}\sum_{l\in\mathcal{G}(k)} J_f(w_l^{t-2K+k+2})u_{w_l}^{t-K+1} \\
&= \sum_{k=0}^{K-1}\sum_{l\in\mathcal{G}(k)} \nabla f_{l,x_{i(t-K+1)}}(w^{t-2K+k+2})^\intercal u_{w_l}^{t-K+1}
\end{aligned}$$

Take expectation with respect to $u_{w_l}^{t-K+1}$, where $l \in \mathcal{G}(k)$, we have

$$\mathbb{E}_{u_{w_l}^{t-K+1}}(\hat{o}_{L_{K-1}}^{t-K+1} \cdot u_{w_l}^{t-K+1}) = \nabla f_{l,x_{i(t-K+1)}}(w^{t-2K+k+2})$$

So we have,

$$\mathbb{E}_{u_{w_{\mathcal{G}(k)}}^{t-K+1}}(\hat{o}_{L_{K-1}}^{t-K+1} \cdot u_{w_l}^{t-K+1}) = \nabla f_{\mathcal{G}(k),x_{i(t-K+1)}}(w^{t-2K+k+2})$$

□

Lemma C.1 ([25], Theorem 3)**.** *Let $g_u(x) = \langle \nabla f(x), u\rangle u$, where $u \in \mathbb{R}^d$ is a normally distributed Gaussian vector, then we have*

$$\mathbb{E}_u\|g_u(x)\|^2 \le (d+4)\|\nabla f(x)\|^2$$

Lemma C.2. *Let $g_{u_1,u_2}(x) = \langle \nabla f(x), u_1\rangle u_2$, where $u_1 \in \mathbb{R}^{d_1}, u_2 \in \mathbb{R}^{d_2}$ are two **i.i.d.** normally distributed Gaussian vectors, then we have*

$$\mathbb{E}_{u_1,u_2}\|g_{u_1,u_2}(x)\|^2 \le d_2\|\nabla f(x)\|^2$$

Proof.

$$\begin{aligned}
\mathbb{E}_{u_1,u_2}\|g_{u_1,u_2}(x)\|^2 =& \mathbb{E}_{u_1,u_2}\|\langle \nabla f(x), u_1\rangle u_2\|^2 = \mathbb{E}_{u_1,u_2}\left(\langle \nabla f(x), u_1\rangle^2 \|u_2\|^2\right) \\
=& \mathbb{E}_{u_1}\left(\langle \nabla f(x), u_1\rangle^2\right)\cdot \mathbb{E}_{u_2}\left(\|u_2\|^2\right) \\
\le& d_2\mathbb{E}_{u_1}\left(\langle \nabla f(x), u_1\rangle^2\right) = d_2\mathbb{E}_{u_1}\left(\sum_{i=1}^{d_1}\nabla_i f(x)u_{1,i}\right)^2
\end{aligned}$$

$$
\begin{aligned}
=&d_2\mathbb{E}_{u_1}\left(\sum_{i=1}^{d_1}\nabla_i^2 f(x)u_{1,i}^2+\sum_{i\neq j}\nabla_i f(x)\nabla_j f(x)u_{1,i}u_{1,j}\right)\\
=&d_2\|\nabla f(x)\|^2
\end{aligned}
$$

where the first inequality is due to Lemma 1 in [25].

□

Proof of Variance.

$$
\begin{aligned}
&\mathbb{E}_{u_w^{t'}}\left\|\sum_{k=0}^{K-1}\mathbf{I}_k\cdot\hat{o}_{L_{K-1}}^{t'}\cdot u_{w_{\mathcal{G}(k)}}^{t'}\right\|^2\\
=&\mathbb{E}_{u_w^{t'}}\sum_{k=0}^{K-1}\left\|\hat{o}_{L_{K-1}}^{t'}\cdot u_{\mathcal{G}(k)}^{t'}\right\|^2=\mathbb{E}_{u_w^{t'}}\sum_{k=0}^{K-1}\left\|\left(\sum_{j=0}^{K-1}\left\langle\nabla f_{\mathcal{G}(j),x_{i(t')}}(w^{t'-K+j+1}),u_{w_{\mathcal{G}(j)}}^{t'}\right\rangle\right)\cdot u_{w_{\mathcal{G}(k)}}^{t'}\right\|^2\\
=&\mathbb{E}_{u_w^{t'}}\sum_{k=0}^{K-1}\left[\left(\sum_{j=0}^{K-1}\left\langle\nabla f_{\mathcal{G}(j),x_{i(t')}}(w^{t'-K+j+1}),u_{w_{\mathcal{G}(j)}}^{t'}\right\rangle\right)^2\cdot\left\|u_{\mathcal{G}(k)}^{t'}\right\|^2\right]\\
=&\mathbb{E}_{u_w^{t'}}\sum_{k=0}^{K-1}\left[\left(\left\langle\nabla f_{\mathcal{G}(k),x_{i(t')}}(w^{t'-K+k+1}),u_{w_{\mathcal{G}(k)}}^{t'}\right\rangle^2+\sum_{j\neq k}\left\langle\nabla f_{\mathcal{G}(j),x_{i(t')}}(w^{t'-K+j+1}),u_{w_{\mathcal{G}(j)}}^{t'}\right\rangle^2\right.\right.\\
&\left.\left.+\sum_{m\neq n}\left\langle\nabla f_{\mathcal{G}(m),x_{i(t')}}(w^{t'-K+m+1}),u_{w_{\mathcal{G}(m)}}^{t'}\right\rangle\cdot\left\langle\nabla f_{\mathcal{G}(n),x_{i(t')}}(w^{t'-K+n+1}),u_{w_{\mathcal{G}(n)}}^{t'}\right\rangle\right)\cdot\left\|u_{\mathcal{G}(k)}^{t'}\right\|^2\right]\\
=&\mathbb{E}_{u_w^{t'}}\sum_{k=0}^{K-1}\left[\left(\left\langle\nabla f_{\mathcal{G}(k),x_{i(t')}}(w^{t'-K+k+1}),u_{w_{\mathcal{G}(k)}}^{t'}\right\rangle^2+\sum_{j\neq k}\left\langle\nabla f_{\mathcal{G}(j),x_{i(t')}}(w^{t'-K+j+1}),u_{w_{\mathcal{G}(j)}}^{t'}\right\rangle^2\right)\cdot\left\|u_{\mathcal{G}(k)}^{t'}\right\|^2\right]\\
\leq&\sum_{k=0}^{K-1}\left[(d_k+4)\left\|\nabla f_{\mathcal{G}(k),x_{i(t')}}(w^{t'-K+k+1})\right\|^2+\sum_{j\neq k}\left(d_k\left\|\nabla f_{\mathcal{G}(j),x_{i(t')}}(w^{t'-K+j+1})\right\|^2\right)\right]\\
=&\sum_{k=0}^{K-1}\left[d_k\sum_{j=0}^{K-1}\left\|\nabla f_{\mathcal{G}(j),x_{i(t')}}(w^{t'-K+j+1})\right\|^2+4\left\|\nabla f_{\mathcal{G}(k),x_{i(t')}}(w^{t'-K+k+1})\right\|^2\right]\\
=&\left(\sum_{k=0}^{K-1}d_k\right)\sum_{j=0}^{K-1}\left\|\nabla f_{\mathcal{G}(j),x_{i(t')}}(w^{t'-K+j+1})\right\|^2+4\sum_{k=0}^{K-1}\left\|\nabla f_{\mathcal{G}(k),x_{i(t')}}(w^{t'-K+k+1})\right\|^2\\
=&(d+4)\sum_{k=0}^{K-1}\left\|\nabla f_{\mathcal{G}(k),x_{i(t')}}(w^{t'-K+k+1})\right\|^2=(d+4)\left\|\sum_{k=0}^{K-1}\mathbf{I}_k\nabla f_{\mathcal{G}(k),x_{i(t')}}\left(w^{t'-K+k+1}\right)\right\|^2,
\end{aligned}
$$

where the inequality is due to Lemma C.1 and C.2. □

D proof of Lemma 5.6

Proof. For the convenience of analysis, we denote $t'=t-K+1$, then the update rule of algorithm 1 can be rewritten as

$$
w_{\mathcal{G}(k)}^{t'+1}=w_{\mathcal{G}(k)}^{t'}-\gamma_{t'}\left(\hat{o}_L^{t'}\cdot u_{w_{\mathcal{G}(k)}}^{t'}\right)
$$

Take expectation with respect to $u_{w_{\mathcal{G}(k)}}^{t'}$, we have

$$
w_{\mathcal{G}(k)}^{t'+1}=w_{\mathcal{G}(k)}^{t'}-\gamma_{t'}\mathbb{E}_{u_{w_{\mathcal{G}(k)}}^{t'}}\left(\hat{o}_L^{t'}\cdot u_{w_{\mathcal{G}(k)}}^{t'}\right)=w_{\mathcal{G}(k)}^{t'}-\nabla f_{\mathcal{G}(k),x_{i(t')}}\left(w_{\mathcal{G}(k)}^{t'-K+k+1}\right)
$$

Define diagonal matrices $\mathbf{I}_0, \cdots, \mathbf{I}_k, \cdots, \mathbf{I}_{K-1} \in \mathbb{R}^{d\times d}$ such that all the principle diagonal elements of $\mathbf{I}_k$ in $\mathcal{G}(k)$ are 1, and all the principle diagonal elements of I_k in other than $\mathcal{G}(k)$ are 0. Then we have

$$\hat{o}_L^{t'} \cdot u_w^{t'} = \sum_{k=0}^{K-1} \mathbf{I}_k \cdot \hat{o}_L^{t'} \cdot u_{w_{\mathcal{G}(k)}}^{t'}$$

$$\nabla f_{x_{i(t')}}\left(w^{t'-K+k+1}\right) = \sum_{k=0}^{K-1} \mathbf{I}_k \nabla f_{\mathcal{G}(k),x_{i(t')}}\left(w_{\mathcal{G}(k)}^{t'-K+k+1}\right)$$

Since $f(\cdot)$ is L-smooth, the following inequality holds that:

$$f(w^{t'+1}) \leq f(w^{t'}) + \left\langle \nabla f(w^{t'}), w^{t'+1} - w^{t'} \right\rangle + \frac{L}{2}\|w^{t'+1} - w^{t'}\|^2$$

From the update rule of Algorithm 1, we take expectation with respect to all random variables on both sides and obtain:

$$\begin{aligned}
\mathbb{E}[f(w^{t'+1})] \leq & f(w^{t'}) - \gamma_{t'}\mathbb{E}\left[\nabla f(w^{t'})^\intercal \left(\sum_{k=0}^{K-1} \mathbf{I}_k \cdot \hat{o}_L^{t'} \cdot u_{w_{\mathcal{G}(k)}}^{t'}\right)\right] + \frac{L\gamma_{t'}^2}{2}\mathbb{E}\left\|\sum_{k=0}^{K-1} \mathbf{I}_k \cdot \hat{o}_{L_{K-1}}^{t'} u_{w_{\mathcal{G}(k)}}^{t'}\right\|^2 \\
= & f(w^{t'}) - \gamma_{t'} \sum_{k=0}^{K-1} \nabla f(w^{t'})^\intercal \mathbf{I}_k \left(\nabla f_{\mathcal{G}(k)}\left(w^{t'-K+k+1}\right) - \nabla f_{\mathcal{G}(k)}\left(w^{t'}\right) + \nabla f_{\mathcal{G}(k)}\left(w^{t'}\right)\right) \\
& + \frac{L\gamma_{t'}^2}{2}\mathbb{E}\left\|\sum_{k=0}^{K-1} \mathbf{I}_k \cdot \hat{o}_{L_{K-1}}^{t'} u_{w_{\mathcal{G}(k)}}^{t'} - \nabla f(w^{t'}) + \nabla f(w^{t'})\right\|^2 \\
= & f(w^{t'}) - \gamma_{t'}\left\|\nabla f(w^{t'})\right\|^2 - \gamma_{t'} \sum_{k=0}^{K-1} \nabla f(w^{t'})^\intercal \mathbf{I}_k \left(\nabla f_{\mathcal{G}(k)}\left(w^{t'-K+k+1}\right) - \nabla f_{\mathcal{G}(k)}\left(w^{t'}\right)\right) \\
& + \frac{L\gamma_{t'}^2}{2}\left\|\nabla f(w^{t'})\right\|^2 + \frac{L\gamma_{t'}^2}{2}\mathbb{E}\left\|\sum_{k=0}^{K-1} \mathbf{I}_k \cdot \hat{o}_{L_{K-1}}^{t'} \cdot u_{w_{\mathcal{G}(k)}}^{t'} - \nabla f(w^{t'})\right\|^2 \\
& + L\gamma_{t'}^2 \sum_{k=0}^{K-1} \nabla f(w^{t'})^\intercal \mathbf{I}_k \left(\nabla f_{\mathcal{G}(k)}\left(w^{t'-K+k+1}\right) - \nabla f_{\mathcal{G}(k)}\left(w^{t'}\right)\right) \\
= & f(w^{t'}) - \left(\gamma_{t'} - \frac{L\gamma_{t'}^2}{2}\right)\left\|\nabla f(w^{t'})\right\|^2 + \underbrace{\frac{L\gamma_{t'}^2}{2}\mathbb{E}\left\|\sum_{k=0}^{K-1} \mathbf{I}_k \cdot \hat{o}_{L_{K-1}}^{t'} \cdot u_{w_{\mathcal{G}(k)}}^{t'} - \nabla f(w^{t'})\right\|^2}_{Q_1} \\
& \underbrace{-(\gamma_{t'} - L\gamma_{t'}^2) \sum_{k=0}^{K-1} \nabla f(w^{t'})^\intercal \mathbf{I}_k \left(\nabla f_{\mathcal{G}(k)}\left(w^{t'-K+k+1}\right) - \nabla f_{\mathcal{G}(k)}\left(w^{t'}\right)\right)}_{Q_2},
\end{aligned}$$

Using the fact that $\|x+y\|^2 \leq 2\|x\|^2 + 2\|y\|^2$ and $xy \leq \frac{1}{2}\|x\|^2 + \frac{1}{2}\|y\|^2$, we have

$$\begin{aligned}
Q_1 = & \frac{L\gamma_{t'}^2}{2}\mathbb{E}\left\|\sum_{k=0}^{K-1} \mathbf{I}_k \cdot \hat{o}_{L_{K-1}}^{t'} \cdot u_{w_{\mathcal{G}(k)}}^{t'} - \nabla f(w^{t'}) - \sum_{k=0}^{K-1} \mathbf{I}_k \nabla f_{\mathcal{G}(k)}\left(w^{t'-K+k+1}\right)\right. \\
& \left. + \sum_{k=0}^{K-1} \mathbf{I}_k \nabla f_{\mathcal{G}(k)}\left(w^{t'-K+k+1}\right)\right\|^2 \\
\leq & L\gamma_{t'}^2 \underbrace{\mathbb{E}\left\|\sum_{k=0}^{K-1} \mathbf{I}_k \cdot \hat{o}_{L_{K-1}}^{t'} \cdot u_{w_{\mathcal{G}(k)}}^{t'} - \sum_{k=0}^{K-1} \mathbf{I}_k \nabla f_{\mathcal{G}(k)}\left(w^{t'-K+k+1}\right)\right\|^2}_{Q_3} +
\end{aligned}$$

$$+ L\gamma_{t'}^2 \underbrace{\left\| \sum_{k=0}^{K-1} \mathbf{I}_k \nabla f_{\mathcal{G}(k)} \left(w^{t'-K+k+1} \right) - \nabla f(w^{t'}) \right\|^2}_{Q_4}$$

$$\begin{aligned} Q_2 =& - (\gamma_{t'} - L\gamma_{t'}^2) \sum_{k=0}^{K-1} \nabla f(w^{t'})^\intercal \mathbf{I}_k \left(\nabla f_{\mathcal{G}(k)} \left(w^{t'-K+k+1} \right) - \nabla f_{\mathcal{G}(k)} \left(w^{t'} \right) \right) \\ \leq & \frac{\gamma_{t'} - L\gamma_{t'}^2}{2} \left\| \nabla f(w^{t'}) \right\|^2 + \frac{\gamma_{t'} - L\gamma_{t'}^2}{2} \left\| \sum_{k=0}^{K-1} \mathbf{I}_k \nabla_{\mathcal{G}(k)} f(w^{t'-K+k+1}) - \nabla f(w^{t'}) \right\|^2 \end{aligned}$$

Using $\mathbb{E}\|\xi - \mathbb{E}[\xi]\|^2 \leq \mathbb{E}\|\xi\|^2$, we have

$$\begin{aligned} Q_3 =& \mathbb{E} \left\| \sum_{k=0}^{K-1} \mathbf{I}_k \cdot \hat{o}_{L_{K-1}}^{t'} \cdot u_{w_{\mathcal{G}(k)}}^{t'} - \sum_{k=0}^{K-1} \mathbf{I}_k \nabla f_{\mathcal{G}(k)} \left(w^{t'-K+k+1} \right) \right\|^2 \\ \leq & \mathbb{E} \left\| \sum_{k=0}^{K-1} \mathbf{I}_k \cdot \hat{o}_{L_{K-1}}^{t'} \cdot u_{w_{\mathcal{G}(k)}}^{t'} \right\|^2 \\ \leq & (d+4) \left\| \sum_{k=0}^{K-1} \mathbf{I}_k \nabla f_{\mathcal{G}(k), x_{i(t')}} \left(w^{t'-K+k+1} \right) \right\|^2 \\ = & (d+4) \sum_{k=0}^{K-1} \left\| \nabla f_{\mathcal{G}(k), x_{i(t')}} (w^{t'-K+k+1}) \right\|^2 \\ \leq & (d+4) KM, \end{aligned}$$

where the second inequality is due to Lemma 5.3. Then we bound Q_4,

$$\begin{aligned} Q_4 =& \left\| \sum_{k=0}^{K-1} \mathbf{I}_k \nabla f_{\mathcal{G}(k)} \left(w^{t'-K+k+1} \right) - \nabla f(w^{t'}) \right\|^2 = \sum_{k=0}^{K-1} \left\| \nabla f_{\mathcal{G}(k)} (w^{t'-K+k+1}) - \nabla f_{\mathcal{G}(k)} (w^{t'}) \right\|^2 \\ \leq & \sum_{k=0}^{K-1} \left\| \nabla f(w^{t'-K+k+1}) - \nabla f(w^{t'}) \right\|^2 \\ \leq & L^2 \sum_{k=0}^{K-1} \left\| w^{t'} - w^{t'-K+k+1} \right\|^2 \\ = & L^2 \sum_{k=0}^{K-1} \left\| \sum_{j=\max\{0,t'-K+k+1\}}^{t'-1} (w^{j+1} - w^j) \right\|^2 = L^2 \sum_{k=0}^{K-1} \left\| \sum_{j=\max\{0,t'-K+k+1\}}^{t'-1} \gamma_j (\hat{o}_{L_{K-1}}^j \cdot u_w^j) \right\|^2 \\ \leq & L^2 \gamma_{\max\{0,t'-K+1\}}^2 \sum_{k=0}^{K-1} K \sum_{j=\max\{0,t'-K+k+1\}}^{t'-1} (d+4) \left\| \sum_{k=0}^{K-1} \nabla f_{\mathcal{G}(k), x_{(j)}} (w^{t'-K+k+1}) \right\|^2 \\ \leq & (d+4) KL\gamma_{t'} \frac{\gamma_{\max\{0,t'-K+1\}}}{\gamma_{t'}} \sum_{k=0}^{K-1} \sum_{j=\max\{0,t'-K+k+1\}}^{t'-1} \left\| \sum_{k=0}^{K-1} \nabla f_{\mathcal{G}(k), x_{(j)}} (w^{t'-K+k+1}) \right\|^2 \\ \leq & (d+4) L\gamma_{t'} \sigma K^4 M, \end{aligned}$$

where the second inequality is from Assumption 5.1, the third inequality is due to Lemma 5.3, the fourth inequality follows from $L\gamma_{t'} < 1$, the last inequality follows from the inequality of arithmetic and geometric means, Assumption 5.2 and $\sigma := \max_{t'} \frac{\gamma_{\max\{0,t'-K+1\}}}{\gamma_{t'}}$. Integrating the upper bound together, we have

$$\mathbb{E}\left[f(w^{t'+1}) - f(w') \right] \leq - \frac{\gamma_{t'}}{2} \left\| \nabla f(w^{t'}) \right\|^2 + (d+4) L\gamma_{t'}^2 KM + \frac{\gamma_{t'} + L\gamma_{t'}^2}{2} (d+4) L\gamma_{t'} \sigma K^4 M$$

$$
\begin{aligned}
&\leq -\frac{\gamma_{t'}}{2}\left\|\nabla f(w^{t'})\right\|^2 + 2(d+4)L\gamma_{t'}^2(KM+\sigma K^4 M) \\
&= -\frac{\gamma_{t'}}{2}\left\|\nabla f(w^{t'})\right\|^2 + 2(d+4)L\gamma_{t'}^2 M_K,
\end{aligned}
$$

where we let $M_K = KM + \sigma K^4 M$.

□

E Proof of Theorem 5.6

Proof. Let $\gamma_t = \gamma$ be a constant, taking total expectation in Lemma 5.5, we have

$$
\mathbb{E}\left[f(w^{t'+1})\right] - \mathbb{E}\left[f(w^{t'})\right] \leq -\frac{\gamma}{2}\mathbb{E}\|\nabla f(w^{t'})\|^2 + 2(d+4)L\gamma^2 M_K,
$$

where $\sigma = 1$ and $M_k = KM + K^4 M$. summing the above inequality from $t' = 0$ to $T-1$ we have

$$
\mathbb{E}[f(w^T)] - f(w^0) \leq -\frac{\gamma}{2}\sum_{t'=0}^{T-1}\mathbb{E}\|\nabla f(w^{t'})\|^2 + 2T(d+4)\gamma^2 L M_K
$$

Then we have

$$
\frac{1}{T}\sum_{t'=0}^{T-1}\mathbb{E}\|\nabla f(w^{t'})\|^2 \leq \frac{2(f(w^0) - f(w^*))}{\gamma T} + 4(d+4)L\gamma M_K.
$$

□

F Proof of Theorem 5.7

Proof. Let $\{\gamma_{t'}\}$ be a diminishing sequence and $\gamma_{t'} = \frac{\gamma_0}{t'+1}$, such that $\sigma < K$ and $M_K = KM + K^5 M$. Taking expectation in Lemma 5.5 and summing it from $t' = 0$ to $T-1$, we have

$$
\mathbb{E}[f(w^T)] - f(w^0) \leq -\frac{1}{2}\sum_{t'=0}^{T-1}\gamma_t\mathbb{E}\|\nabla f(w^{t'})\|^2 + \sum_{t'=0}^{T-1}2(d+4)\gamma_{t'}^2 L M_K.
$$

Letting $\Gamma_T = \sum_{t'=0}^{T-1}\gamma_{t'}$, then we have

$$
\frac{1}{\Gamma_T}\sum_{t'=0}^{T-1}\gamma_{t'}\mathbb{E}\|\nabla f(w^{t'})\|^2 \leq \frac{2\left(f(w^0) - f(w^*)\right)}{\Gamma_T} + \frac{\sum_{t'=0}^{T-1}4(d+4)\gamma_{t'}^2 L M_K}{\Gamma_T}
$$

□

G Details of AsyncFGD

G.1 Working with Adam

We provide example for AsyncFGD working with Adam in Algorithm 2. Minimal changes are made on Adam by substituting the gradient the estimator using Forward Gradient.

G.2 Execution Details

Details are presented in Figure 5. By pipelining over time dimension, we can preserve buffer for input in only one timestamp and still achieve parallel computation.

Algorithm 2 AsyncFGD-Adam

Initialize: Stepsize sequence $\{\gamma_t\}_{t=K-1}^{T-1}$, weight $w^0 = \left[w^0_{\mathcal{G}(0)}, \cdots, w^0_{\mathcal{G}(K-1)}\right] \in \mathbb{R}^d, m_{\mathcal{G}(k)} = 0, v_{\mathcal{G}(k)} = 0, \beta_1 = 0.9, \beta_2 = 0.999, \eta = 1e-8$

for $t = 0, 1, \cdots, T-1$ **do**
  for $k = 0, 1, \cdots, K-1$ **in parallel do**
    Read $\hat{h}^{t-k}_{L_{k-1}}, \hat{o}^{t-k}_{L_{k-1}}$ from storage if $k \neq 0$
    Compute $\hat{h}^{t-k}_{L_k}, \hat{o}^{t-k}_{L_k}$
    Send $\hat{h}^{t-k}_{L_k}, \hat{o}^{t-k}_{L_k}$ to next worker's storage if $k \neq K-1$
  end for
  Broadcast $\hat{o}^{t-K+1}_{L_{K-1}}$
  for $k = 0, 1, \cdots, K-1$ **in parallel do**
    Compute $\Delta w^{t-K+1}_{\mathcal{G}(k)} = \hat{o}^{t-K+1}_{L_{K-1}} u^{t-K+1}_{w_{\mathcal{G}(k)}}$
    Update $m_{\mathcal{G}(k)} = \beta_1 \Delta w^{t-K+1}_{\mathcal{G}(k)} + (1-\beta_1) m_{\mathcal{G}(k)}$
    Update $v_{\mathcal{G}(k)} = \beta_2 \Delta w^{t-K+1}_{\mathcal{G}(k)} + (1-\beta_2) v_{\mathcal{G}(k)}$
    Compute $\hat{m}_{\mathcal{G}(k)} = m_{\mathcal{G}(k)} / \beta_1^t$
    Compute $\hat{v}_{\mathcal{G}(k)} = v_{\mathcal{G}(k)} / \beta_2^t$
    Update $w^{t-K+2}_{\mathcal{G}(k)} = w^{t-K+1}_{\mathcal{G}(k)} - \gamma_{t-K+1} \hat{m}_{\mathcal{G}(k)} / \hat{v}_{\mathcal{G}(k)}$
  end for
end for

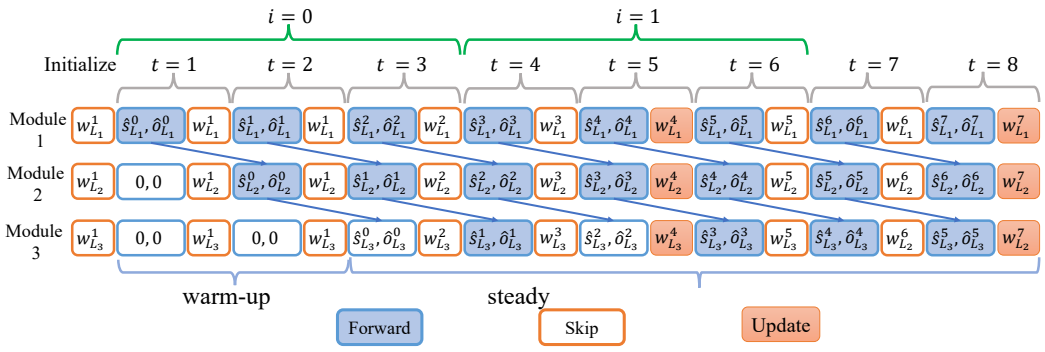


Figure 5: Details in executon of AsyncFGD-RNN with 3 modules. In the skip stage, only the host accumulate loss and its jvp value and other workers will jump right into the next state.

G.3 Extension: AsyncFGD-Recursive

In this section, we extend the potential of AsyncFGD by exploring the parallelization of sequential inputs in RNNs with reduced memory footprint, necessitating the preservation of input for only a single timestamp.

We adopt a one-to-many RNN network for ease of illustration and denote the equal length of each sequence as n. We begin by refactoring the original loss for RNNs in terms of cumulative loss and new activation. Here, s_l^t signifies the hidden state at timestamp t on layer l. At timestamp t, each layer ingests (s_{l-1}^t, s_l^{t-1}) as input, generating $s_l^t = F_l(s_{l-1}^t, s_l^{t-1}, w_l)$. We represent the stacked latent states passed from $t-1$ as $s^{t-1} = [s_1^{t-1}, s_2^{t-1}, \ldots, s_L^{t-1}]$ and the output as $y_t = F(s_0^t, s^{t-1}; w)$, where s_0^t symbolizes the input data x_t. The cumulative loss from timestamp $1 \sim T$ is given by:

$$\sum_{t=1}^{T} f(F(x_t, s^{t-1}; w), y_t) \tag{15}$$

We next refactor equation 2 for the i_{th} sequential input in iteration $i, i \geq 0$ as:

$$w_l^{i+1} = w_l^i - \gamma_i \frac{\partial \mathcal{L}_i}{\partial w_l^i}, \quad \forall l \in 1, 2, \ldots, L \tag{16}$$

where $\mathcal{L}i := \sum t = in + 1^{(i+1)n} f(F(x_t, s^{t-1}; w), y_t)$ represents the loss for the i_{th} sequence. We break the dependency between timestamps and iterations by employing dynamic staleness in AsynFGD. Specifically, the computation in module $k \in 1, 2, \cdots, K$ at timestamp t is defined as follows:

$$\hat{s}L_k{}^{t-k} = f_k \left(\hat{s}L_{k-1}{}^{t-k}, \hat{s}\mathcal{G}(k)^{t-k-1}; w\mathcal{G}(k)^{t-2K+k+2} \right) \tag{17}$$

$$\hat{o}L_k{}^{t-k} = Jf_k \left(\hat{s}Lk - 1^{t-k}, \hat{s}\mathcal{G}(k)^{t-k-1}; w\mathcal{G}(k)^{t-2K+k+2} \right) u_{\mathcal{G}(k)}^{t-k}, \tag{18}$$

$$\text{where } u_{\mathcal{G}(k)}^{t-k} = [\hat{o}Lk - 1^{t-k\intercal}, \hat{o}\mathcal{G}(k)^{t-k-1\intercal}, uw_{\mathcal{G}(k)}{}^{t-k\intercal}]^\intercal$$

Given that tasks belonging to the same iteration use identical parameters, we use $\delta(k, t, i) = t - ni - k - 1, t \in [in+1, (i+1)n]$ to quantify this difference for the i_{th} sequential. If $\delta(k, t, i) \leq 0$, then module k uses stale parameters from iteration $i - 1$ at timestamp t. AsyncFGD-RNN only updates the parameter upon the completion of the last computation in the sequence. Specifically, we use:

$$w^{t-K+2} = \begin{cases} w^{t-K+1}, & \text{if } \dfrac{t-K+1}{n} \notin \mathbb{N}^* \\ w^{t-K+1} - \gamma_{\lfloor \frac{t-K}{n} \rfloor} \mathbb{E} u w^t \left(\left(\dfrac{\partial \mathcal{L} \lfloor \frac{t-K}{n} \rfloor}{\partial s L_K{}^{t-K}} o_{L_K}^{t-K} \right) u_w^{(t-K)} \right), & \text{otherwise} \end{cases}$$

Refer to figure 5 for detailed execution instructions. By combining training and prediction, we can process data from different timestamps of sequential input, maintain a buffer for just a single timestamp, and still achieve parallelization among various workers.

H Training Details

In this section, we explain some details in the trainning process.

H.1 Random Seed

The seeds for all experiments are fixed to 0.

H.2 Description of Network Architecture

H.2.1 Models Deployed in Section 6.2.

The network structures of $ConvS, ConvL, FCS$ and FCL are enumerated in Tables 8, 11, 9, and 10 correspondingly. In these designations, the subscripts $ReLU$ and $Tanh$ signify the particular activation function used in the model. Moreover, the larger models, denoted as counterparts, incorporate batch normalization layers for enhanced performance.

H.2.2 Models used in Section 6.3.

We delve into the specific models employed in Section 6.3. For MobileNet, we utilize the structure of MobileNet_V3_Small. The ResNet-18 structure is used when implementing ResNet. The model EfficientNet_B0 signifies the EfficientNet architecture. The MNASNet0_5 is used for MNasNet. Lastly, we adopt ShuffleNet_V2_X0_5 for ShuffleNet.

H.3 Model Splitting

In this section, we provide details for how to split the model into consective moduels and distribute them in different worker, we will first provide how to split models in Section 6.2, then we provide how to split model in 6.3.

H.3.1 Model Splitting in Section 6.2

In Section 6.2, all models are split with $K = 3$. Details for how to split the ConvS, ConvL, FCS, FCL are repented in Table 5 Table 6, Table3 and Table 4, respectively.

Table 3: Details for model splitting for ConvS, definition for layers can be found in Table 8

K	Layers
1	conv1, act
2	pool1, fc1
3	act2, fc2

Table 4: Details for model splitting for ConvL, definition for layers can be found in Table 11

K	Layers
1	conv1, bn1, act1, pool1, conv2, bn2, act2, pool2
2	conv3, bn3, act3, pool3,conv4, bn4, act4, pool4
3	conv5, bn5, act5, pool5, fc1

Table 5: Details for model splitting for FCS, definition for layers can be found in Table 9

K	Layers
1	fc1,ac1
2	fc2,ac2
3	fc3,ac3

Table 6: Details for model splitting for FSL, definition for layers can be found in Table10

K	Layers
1	fc1, bn1, ac1, fc2, bn2, ac2
2	fc3, bn3, ac3 ,fc4, bn4, ac4
3	fc5,bn5,ac5,fc5

H.3.2 Model Splitting in Section 6.3

In section 6.3, all models are divided into four parts ($K = 4$). Detailed descriptions of how each model is split are provided below. Note that 'head' and 'tail' refer to the layers before and after the main blocks of each architecture, respectively, which are assigned to the first and the last worker:

- **ResNet-18:** The core of ResNet-18 consists of 4 Residual Blocks, each distributed to one of the four workers.
- **EfficientNet:** The core of EfficientNet consists of 7 Mobile Inverted Bottlenecks (MBConv). The first worker handles MBConv 1, the second handles MBConv 2 to 3, the third manages MBConv 4 to 6, and the last one manages MBConv 7.
- **MoblieNet:** The core of MoblieNetV3-small includes 13 layers of bottlenecks. The first worker handles layers 1 to 3, the second manages layers 4 to 7, the third manages layers 8 to 11, and the last worker handles layers 12 to 13.
- **MnasNet:** The core of MnasNet consists of 6 blocks of inverted residuals. Blocks 1 to 2, 3 to 5, and 6 are assigned to workers 2, 3, and 4 respectively, while the first worker only handles the head.
- **ShuffleNet:** The core of ShuffleNet consists of 3 stages, each assigned to workers 2, 3, and 4, respectively.

I Additional Experimental Results

I.1 Ablation study in α

We have incremented the value of α_{bias} gradually, with a step size of 0.0075, over 20 epochs. This process can be generalized using the following equations:

$$\alpha_{bias} = \begin{cases} t \times \frac{\alpha^*_{bias}}{20}, & t <= 20 \\ \alpha^*_{bias}, & \text{otherwise} \end{cases}$$

Here, α^*_{bias} control the rate of increase and the maximum attainable value of α_{bias}, respectively. The ablation study with respect to α^*_{bias} is presented in Table 7.

We observe that reducing α^*_{bias} to 0, which corresponds to only updating the classifier, still results in performance gains compared to updating the full model. This improvement can be attributed to reduced variance. As α^*_{bias} increases, we generally see better results, since the norm of the gradient approximation increases. However, when α^*_{bias} exceeds 0.25, we sometimes observe a performance drop due to the corresponding increase in variance.

Table 7: Ablation study on different value of α^*_{bias}

Dataset	Model	α^*_{bias}							
		0.00	0.03	0.06	0.09	0.12	0.15	0.20	0.25
CIFAR10	Res-18	0.838	0.838	0.845	0.855	0.865	0.878	0.872	0.885
	Mobile	0.898	0.912	0.911	0.910	0.913	0.911	0.914	0.909
	Efficient	0.892	0.900	0.902	0.903	0.902	0.902	0.887	0.895
	Shuffle	0.788	0.805	0.808	0.812	0.820	0.820	0.822	0.825
	Mnas	0.782	0.790	0.788	0.789	0.788	0.789	0.777	0.782
FMNIST	Res-18	0.866	0.869	0.871	0.873	0.875	0.880	0.882	0.884
	Mobile	0.890	0.908	0.906	0.906	0.906	0.906	0.899	0.901
	Efficient	0.889	0.904	0.906	0.902	0.905	0.904	0.908	0.897
	Shuffle	0.849	0.854	0.857	0.860	0.864	0.870	0.870	0.877
	Mnas	0.854	0.868	0.870	0.870	0.870	0.870	0.864	0.864

I.2 Acceleration across Various Platforms and Architectures

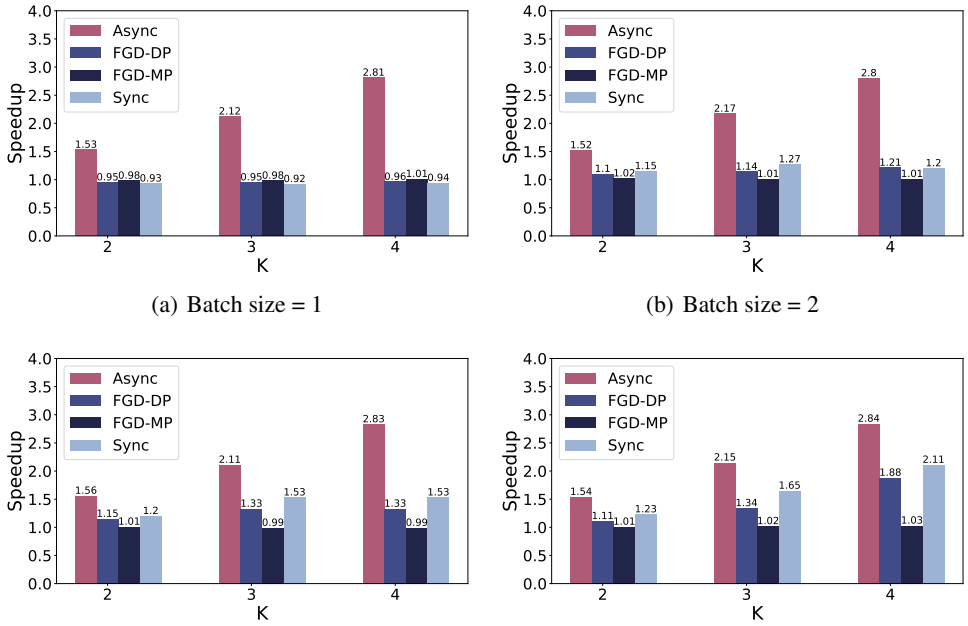
In Section 6.5, we examined the acceleration of AsyncFGD in comparison to vanilla FGD on ResNet-18, using two hardware platforms: 1) NVIDIA AGX Orin, an embedded device, and 2) a cluster of four NVIDIA 1080 Ti GPUs. These platforms were chosen to reflect real-world edge device scenarios and to simulate situations of ample computational power, such as in the case of stacked chips.

In this section, we expand our scope of investigation by incorporating two additional devices: 1) NVIDIA A100, and 2) Intel(R) Xeon(R) CPU E5-2678 v3 @2.50GHZ. These additions allow us to further examine acceleration under various conditions. We also provide supplementary results on acceleration with respect to different batch sizes to reflect variable input streams. Moreover, to emulate streamlined input, the mini-batch size of the synchronized pipeline is set to 1.

The performance of ResNet-18 with different batch sizes on the four NVIDIA 1080Ti GPUs, A100, and AGX Orin platforms is illustrated in Figures 6, 7, and 8, respectively. Results for MobileNetV3-small on AGX Orin are presented in Figure 10. A notable observation is that AsyncFGD performance appears largely insensitive to batch size. In contrast, other algorithms typically exhibit poorer performance with smaller batch sizes. Particularly, when the batch size is reduced to 1, these algorithms offer negligible performance improvements over vanilla FGD. Furthermore, the overall

acceleration on a single device is constrained by computational power. For instance, while AsyncFGD achieves a speedup of 2.84× on a four GPU cluster, it only delivers a 2.11 × speedup on a single AGX Orin. Communication also imposes a limit on the overall acceleration, as demonstrated by the superior performance on the A100 in comparison to the four-GPU cluster. This is attributable to the elimination of communication overhead on a single device, except for the sending and receiving operations of CUDA kernels.

Results for MobileNetV3-small with different batch sizes on CPU are depicted in Figure 9. Due to the inherently sequential execution pattern of CPUs, the acceleration is constrained, resulting in only modest speedup and advantage over other algorithms.



(a) Batch size = 1

(b) Batch size = 2

(c) Batch size = 3

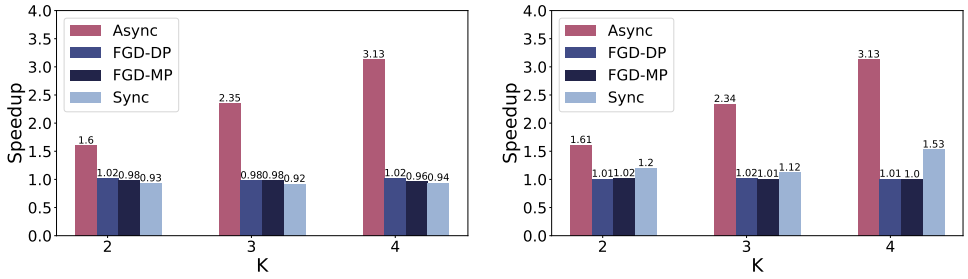
(d) Batch size = 4

Figure 6: Acceleration with differernt batch size on ResNet-18, cluster with 4 Nvidia 1080 Ti

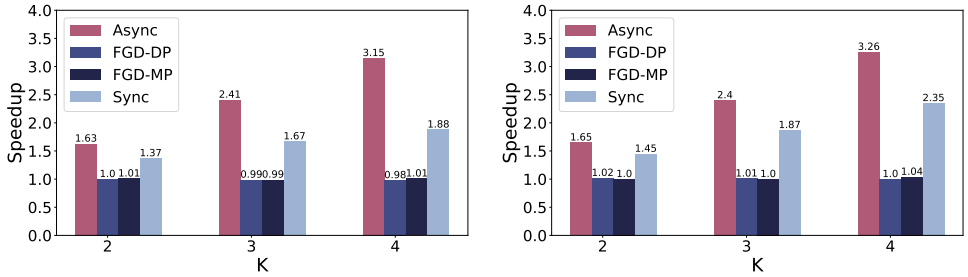
I.3 Memory Profiling on Other Basic Units of Convolutional Neural Networks

This section outlines memory profiling for basic units within a Convolutional Neural Network (CNN). Commonly, a CNN layer is coupled with a batch normalization layer and an activation layer using ReLU, so we've combined these elements for our memory testing. We examine the memory consumption against the number of layers and present the results in Figure 11(a). For further examination, we also assess the memory consumption against the number of output channels and batch size, with results shown in Figures 11(b) and 11(c), respectively.

Our findings reveal that implementing forward gradients can significantly reduce memory consumption. Generally, the majority of memory usage in CNNs stems from intermediate results, since CNNs often operate in a 'broadcast then product' pattern (to be specific, they are referred as 'img2col'). Consequently, the additional memory required by the random tangent in AsyncFGD is minimal. As such, the memory consumption appears to be invariant to the number of layers, mainly because in the forward pass we discard almost all the intermediate variables.

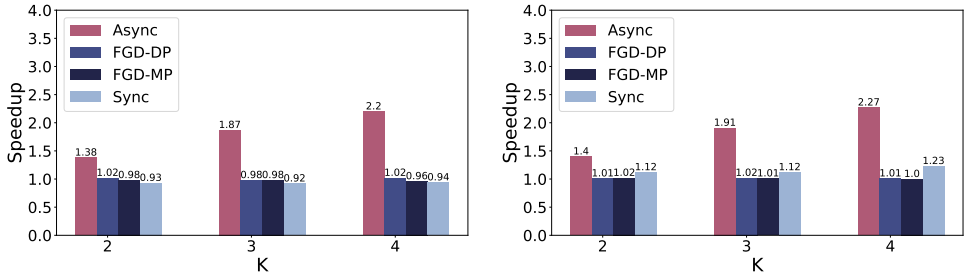


(a) Batch size = 1 (b) Batch size = 2

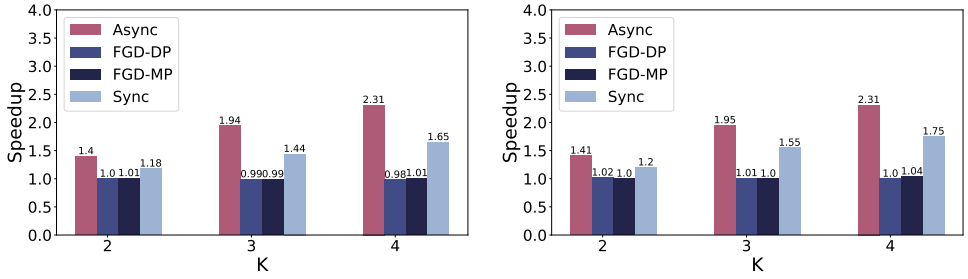


(c) Batch size = 3 (d) Batch size = 4

Figure 7: Acceleration with differernt batch size on ResNet-18, A100

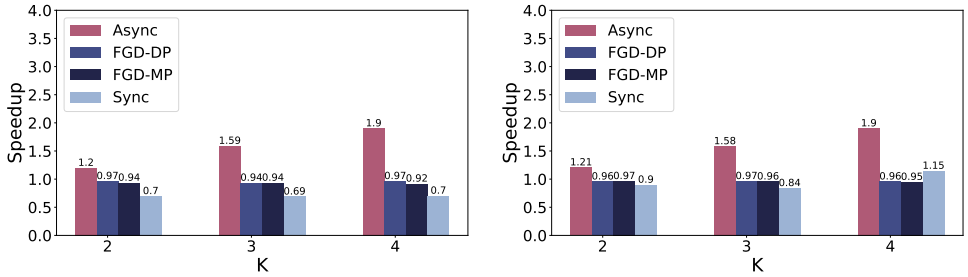


(a) Batch size = 1 (b) Batch size = 2



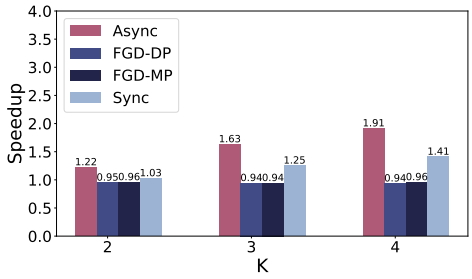
(c) Batch size = 3 (d) Batch size = 4

Figure 8: Acceleration with differernt batch size on ResNet-18, AGX Orin

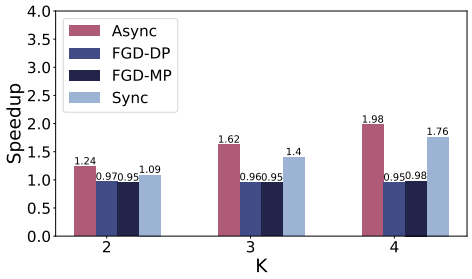


(a) Batch size = 1

(b) Batch size = 2

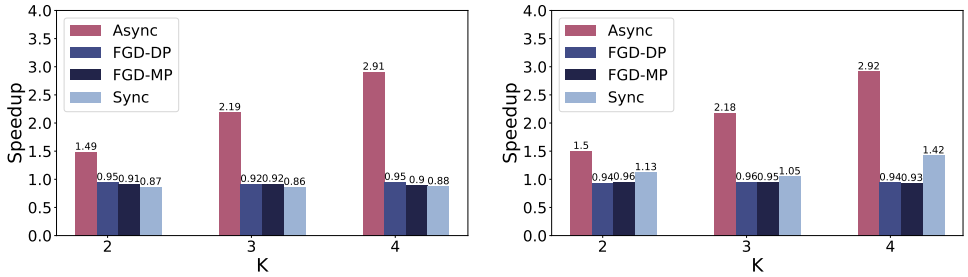


(c) Batch size = 3



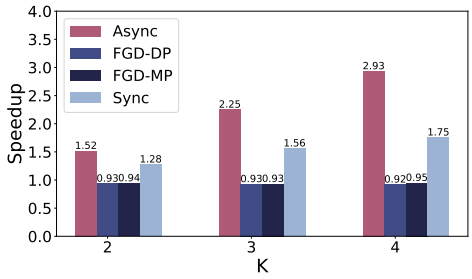
(d) Batch size = 4

Figure 9: Acceleration with differernt batch size on MobileNetV3-small, Intel(R) Xeon(R) CPU E5-2678 v3 @ 2.50GHz

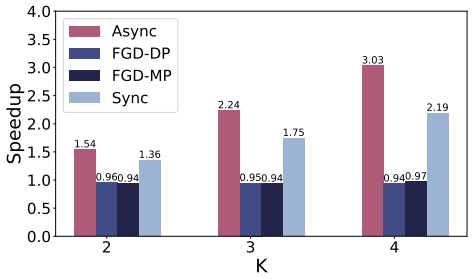


(a) Batch size = 1

(b) Batch size = 2



(c) Batch size = 3



(d) Batch size = 4

Figure 10: Acceleration with differernt batch size on MobileNetV3-small, AGX Orin

Table 8: Network architecture for $ConvS_{ReLU}$ and $ConvS_{Tanh}$. $ConvS_{ReLU}$ denotes using ReLU for the activation functions and $ConvS_{Tanh}$ denotes using Tanh as activation functions

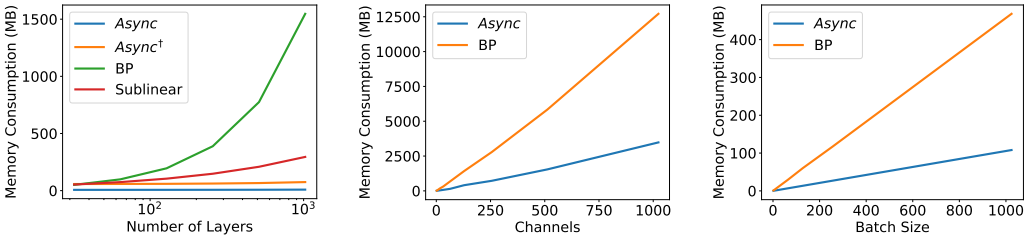
Layer	Type	Params
conv1	Conv2d	out_channels=32, kernel_size=5, stride=1, padding=2
act1	ReLU/Tanh	N/A
pool1	Maxpooll2d	kernel_size=2, stride=2, padding=0, dilation=1
fc1	Linear	out_features=1000
act2	ReLU/Tanh	N/A
fc2	Linear	out_features=10

Table 9: Network architecture for FCS_{ReLU} and FCS_{Tanh}. FCS_{ReLU} denotes using ReLU for the activation functions and FCS_{Tanh} denotes using Tanh as activation functions

Layer	Type	Params
flatten	Flatten	N/A
fc1	Linear	out_features=1024
ac1	ReLU/Tanh	N/A
fc2	Linear	out_features=512
ac2	ReLU/Tanh	N/A
fc3	Linear	out_features=256

Table 10: Network architecture for FCL_{ReLU} and FCL_{Tanh}. FCL_{ReLU} denotes using ReLU for the activation functions and FCL_{Tanh} denotes using Tanh as activation functions

Layer	Type	Params
flatten	Flatten	N/A
fc1	Linear	out_features=1024
bn1	BatchNorm1d	N/A
ac1	ReLU/Tanh	N/A
fc2	Linear	out_features=1024
bn2	BatchNorm1d	N/A
ac2	ReLU/Tanh	N/A
fc3	Linear	out_features=1024
bn3	BatchNorm1d	N/A
ac3	ReLU/Tanh	N/A
fc4	Linear	out_features=1024
bn4	BatchNorm1d	N/A
ac4	ReLU/Tanh	N/A
fc5	Linear	out_features=512
bn5	BatchNorm1d	N/A
ac5	ReLU/Tanh	N/A
fc6	Linear	out_features=10



(a) Different number of layers (b) Different channels (c) Different batch size

Figure 11: Memory consumpition of basic units in convoluntional networks, batch size=64, channels=3 and number of layers=18 unless appears as in the x axis

Table 11: Network architecture for $ConvL_{ReLU}$ and $ConvL_{Tanh}$. $ConvL_{ReLU}$ denotes using ReLU for the activation functions and $ConvL_{Tanh}$ denotes using Tanh as activation functions

Layer	Type	Params
conv1	Conv2d	out_channels=32, kernel_size=3, stride=1, padding=2
bn1	BatchNorm2d	N/A
act1	ReLU/Tanh	N/A
pool1	Maxpooll2d	kernel_size=2, stride=2, padding=0, dilation=1
conv2	Conv2d	out_channels=64, kernel_size=3, stride=1, padding=2
bn2	BatchNorm2d	N/A
act2	ReLU/Tanh	N/A
pool2	Maxpooll2d	kernel_size=2, stride=2, padding=0, dilation=1
conv3	Conv2d	out_channels=128, kernel_size=3, stride=1, padding=2
bn3	BatchNorm2d	N/A
act3	ReLU/Tanh	N/A
pool3	Maxpooll2d	kernel_size=2, stride=2, padding=0, dilation=1
conv4	Conv2d	out_channels=256, kernel_size=3, stride=1, padding=2
bn4	BatchNorm2d	N/A
act4	ReLU/Tanh	N/A
pool4	Maxpooll2d	kernel_size=2, stride=2, padding=0, dilation=1
conv5	Conv2d	out_channels=512, kernel_size=3, stride=1, padding=2
bn5	BatchNorm2d	N/A
act5	ReLU/Tanh	N/A
pool5	Maxpooll2d	kernel_size=2, stride=2, padding=0, dilation=1
flatten	Flatten	N/A
fc1	Linear	out_features=10