

Split Learning of Multi-Modal Medical Image Classification

Bishwamittra Ghosh^{1,2} Yuan Wang¹ Huazhu Fu¹ Qingsong Wei¹ Yong Liu¹ Rick Siow Mong Goh¹

¹*Institute of High Performance Computing (IHPC), Agency for Science, Technology and Research (A*STAR), Singapore*

²*Max Planck Institute for Software Systems, Saarbrücken, Germany*

Abstract—In the past decades, machine learning (ML) has made significant progress in medical image classification. The success can be attributed to two factors: (i) unique patient data collected and processed by clinics/hospitals and (ii) corresponding ML models solving the underlying classification task. In practice, patient data may contain sensitive information unique to patients’ demography; and ML models often require higher computational resources beyond the affordability of an individual hospital.

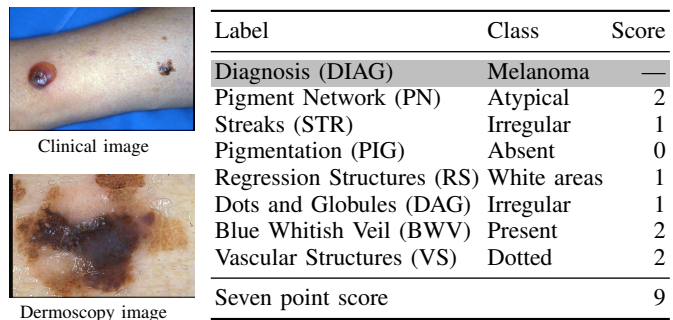
Considering practical concerns, we explore a collaborative ML approach in which the data provider, referred to as the client, aims to leverage the computational resources of a server in jointly training a unified ML model without the need to share any raw data. Specifically, we focus on the *skin lesion* classification problem using a real-world dataset containing multi-modal image inputs and multi-label ground truth.

To enable collaborative yet privacy-preserving skin lesion classification, we develop a learning framework called SplitFusionNet based on *u-shape split learning*. The key idea of SplitFusionNet is to split the ML model into a (client, server) partition of deep neural network layers: the client layers process multi-modal input data and multi-labels, while server layers perform computationally extensive mid-layer computations. Additionally, we apply lossless compression and decompression to improve the communication cost between the client and the server. Experimentally, SplitFusionNet requires less training pipeline time than non-split centralized training while achieving equal predictive performance.

Index Terms—Split-learning, Multi-modal Classification, Multi-label Classification, Privacy-preserving Machine Learning

I. INTRODUCTION

In the past decades, machine learning (ML) has seen advancement as a predictive system, particularly in the healthcare and medical domains [1, 2]. The emergence of deep neural networks (DNNs) such as convolutional neural networks [3], residual networks [4], and vision transformers [5] has demonstrated success in diverse medical data analysis tasks like image classification, object detection, and more. This success incentivizes core institutions like clinics and hospitals to collect and process unique patient data in image and text formats and employ DNNs for various pattern recognition tasks [6, 7]. However, DNN models demand higher computational resources such as GPUs/TPUs (Graphical/Tensor Processing Units) for effective training [8, 9], which is not always affordable by individual hospitals [10]. As such, hospitals may disclose sensitive patient data to a third party server for an efficient training while being susceptible to data privacy leakage. While there are contemporary approaches to tackle



Label	Class	Score
Diagnosis (DIAG)	Melanoma	—
Pigment Network (PN)	Atypical	2
Streaks (STR)	Irregular	1
Pigmentation (PIG)	Absent	0
Regression Structures (RS)	White areas	1
Dots and Globules (DAG)	Irregular	1
Blue Whitish Veil (BWV)	Present	2
Vascular Structures (VS)	Dotted	2
Seven point score		9

Fig. 1: A representative multi-modal image input (left) for multi-label classification problem (right) from Seven point checklist criteria dataset.

privacy preserving [11–15] and efficient DNN training [16–18], we resort to a recent advancement in collaborative ML, called split learning.

Split Learning. Collaborative ML enables multiple parties to share learning resources like data and computational capabilities [19, 20]. Split learning [21–25], a collaborative ML approach, involves a *client* with unique and high-quality data collaborating with a *server* possessing higher computational resources to jointly learn a unified DNN model. Split learning facilitates collaborative yet privacy-preserving learning, with the client sharing abstract representational features with the server instead of raw input data [22]. While split learning has been extensively studied on a single modality input [21, 22], a less explored research area that is our focus is *multi-modal image classification* [26].

Skin Lesion Classification. Skin cancer stands as the fourth leading cause of malignancy in humans [27], emphasizing the critical need for early detection via skin lesion classification [28–30]. Notably, the early detection of melanoma, the most perilous form of skin cancer, can increase the 10-year survival rate from below 39% to over 93% [31].

Our investigation centers on the *seven-point checklist criteria* dataset, a widely studied dataset in DNN-based skin cancer detection problems [32, 33]. This dataset poses a unique classification challenge, demanding a combined approach to multi-modal and multi-label classification. Specifically, in Figure 1 we address two image modalities: clinical images portraying diverse skin views captured by a digital camera and dermoscopy images showcasing sub-surface structures

obtained through skin imaging techniques. The classification task involves eight multi-label classification: one diagnosis label as shown in the gray row and seven checklist criteria labels. To our best knowledge, all prior approaches consider skin lesion classification in a centralized training involving GPUs and disregard the practical scenario where hospitals may have limited computational resources [29, 32, 34–37]. Therefore, the central research question in this paper is: *Can we devise a ML framework for multi-modal and multi-label skin lesion classification that harnesses the computational resource of a powerful server without necessitating the sharing of raw data by the data owner?*

Contribution. We propose SplitFusionNet based on *u-shape split learning* of a DNN model for multi-modal and multi-label classification tasks. The key idea is partitioning the DNN model into (client, server) layers: client layers handle multi-modal images and associated labels, while server layers undertake computationally intensive mid-layer computations. We consider the client and server components residing in separate physical machines – the transfer of representational features and gradients between the server and the client is performed via internet-based socket programming. Thus, the bottleneck in SplitFusionNet is the additional communication cost, which we address via implementing a lossless compression/decompression mechanism [38] for features and gradients on both the client and server sides. Lossless compression significantly enhances communication speed without compromising the prediction performance of split learning.

Experimental Results. In our experiments, we emulate SplitFusionNet by allocating the client on a CPU (Central Processing Unit) and the server on a GPU. Our observations can be summarized into three key points:

- 1) *Equal prediction performance:* SplitFusionNet achieves an equal prediction accuracy to a non-split centralized DNN model [34] addressing the same classification task based on the seven-point criteria checklist dataset.
- 2) *Efficient training:* The training time alone is reduced to 7% in SplitFusionNet compared to the non-split CPU-based implementation. This efficiency is owed to leveraging the GPU on the server via split learning.
- 3) *Reduced training pipeline time:* Through lossless compression/decompression of features and gradients within SplitFusionNet, the total training pipeline time, including communication and compression expenses, is reduced to 49% of the time required by the non-split method.

Related Work

In skin lesion classification, deep multi-modal learning has been applied by utilizing shared cross-modality features in the underlying classification task [29, 32, 34–37]. Notably, [29] used two ResNet50 models to extract clinical and dermoscopy images, while [32] adopted the InceptionV3 model to fuse multi-modality data. Later, [36] introduced a hyper-connected CNN to extract cross-modality features from all layers of the CNN, unlike [29, 32] that focused on the final few layers. Recently, [34] proposed a two-stage procedure to process

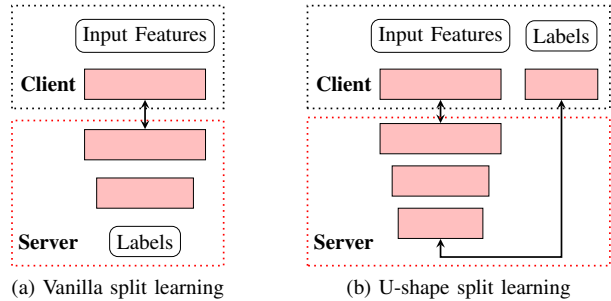


Fig. 2: Split learning with vanilla (left) and u-shape (right) configurations. Unlike vanilla split, the u-shape split retains both input features and labels on the client (top; black dotted rectangle), ensuring higher data privacy. The double arrow (\leftrightarrow) at the cut layer signifies the transfer of the representation features and gradients during the forward and backward passes.

multi-modal image modalities followed by the metadata of patients. Unlike previous methods that considered complementary features from two image modalities, [37] considered both correlated and complementary features in an adversarial setting. To our best knowledge, all earlier methods considered a centralized learning setting for skin lesion classification and none of them focused on collaborative ML training with the goal of achieving faster training while prioritizing data privacy – in this work, we address both via split learning.

II. PRELIMINARIES

Dataset. We consider a multi-modal and multi-label classification dataset represented as $\mathbf{D} = (\mathbf{x}_c, \mathbf{x}_d, \mathbf{y}) \sim \mathcal{D}$, where \mathcal{D} is the joint probability distribution of two images and a multi-label vector. The clinical image $\mathbf{x}_c \in \mathbb{R}^{w \times h \times c}$ and the dermoscopy image $\mathbf{x}_d \in \mathbb{R}^{w \times h \times c}$ have equal dimensions w , h , and c denoting their width, height, and channel number, respectively. The multi-label vector $\mathbf{y} \triangleq [y^{(1)}, \dots, y^{(8)}]$ comprises labels with cardinality $|\mathbf{y}| = 8$. Each label $y^{(i)} \subseteq \mathbb{Z}$ is drawn from a set of integers, where the number of classes per label ranges between 2 to 5.

Split Learning. Split learning [22] is a collaborative and privacy-preserving training procedure that facilitates the training of low-end edge devices through collaboration with a powerful central server. Inspired by message-passing, the core concept of split learning involves dividing the neural network into a shallow network on the client side and a deep network on the server side so that computationally intensive training occurs on the server without sharing the raw data of the client. Various configurations of split learning have been proposed in the literature (Figure 2). In vanilla split learning (Figure 2a), the client trains a partial neural network up to the *cut layer* and transmits the output of this cut layer to the server. Upon receiving the client’s output, the server completes the forward pass through the remaining layers. During the backward pass, the server’s gradients are computed from its last layer to the cut layer. Then, these cut layer gradients are sent back to the client, who performs the rest of the backward pass. In the

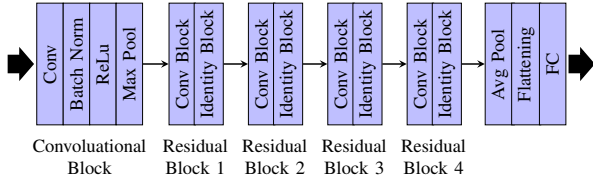


Fig. 3: A ResNet model architecture. Sequential computational blocks make ResNet suitable for split learning.

vanilla configuration, the client shares the ground-truth labels only, which might still pose a risk to the data privacy.

In u-shape split learning (Figure 2b), both input features and labels remain on the client side. Hence, the network is partitioned into three sub-networks: *client-head*, *server-body*, and *client-tail*. The server-body contains the majority of the layers following client-head network, wherein the client-tail network has, in the simplest form, a linear classification layer to compare logits with labels. In the forward pass, the output of the cut layer at the client-head is sent to the server-body network, and the output of the cut layer at the server-body is sent to the client-tail network. In backward pass, gradients are sent from the client-tail to the server-body and from the server-body to the client-head. Therefore, u-shape split learning not only safeguards sensitive input features (clinical and dermoscopy images) but also protects sensitive label information (diagnosis and seven-point criteria labels).

Residual Network. A Residual Neural Network or ResNet [4] is a convolutional neural network designed to learn residual functions concerning the input across a series of stacked layers (Figure 3). We specifically focus on ResNet50, a convolutional neural network comprising 4 residual blocks preceded by a convolutional block, resulting in a total of 50 layers. The sequential computational blocks in ResNet renders it suitable for split learning, which we explore in this work.

Lossless Compression and Decompression. Lossless compression is a data compression method to reduce the size of original data through compression while ensuring a perfect decompression from the compressed data without any loss of information [39, 39]. Most lossless compression algorithms leverage the statistical redundancy within the target data. In contrast, lossy compression [40] generally achieves a higher compression ratio (i.e., the ratio of the size of original and compressed data), but it can only approximate the original data during decompression. In our study, we implement compression and decompression on representational features and gradients – an approximation of these values might necessitate additional training epochs and prolongs the overall training pipeline time. Henceforth, we consider lossless compression and decompression in the paper.

III. METHODOLOGY

We discuss the main contribution of the paper, SplitFusionNet, a u-shape split learning of fusion DNN model operating on two image modalities. We first discuss

the non-split baseline model in Section III-A and propose an extension to the u-shape split model in Section III-B.

A. FusionNet

We discuss the baseline non-split model, referred to as FusionNet [34], which combines two image modalities for multi-label classification (refer to Figure 4, disregarding the client-server partition, which we explain shortly). FusionNet utilizes two pre-trained ResNet-50 backbones to extract features from the two modalities. Formally, we denote a ResNet model as $C : \mathbb{R}^{w \times h \times c} \rightarrow \mathbb{R}^N$, converting an image \mathbf{x} to a representational feature \mathbf{b} . The ResNet for the clinical image, $C_c(\mathbf{x}_c) = \mathbf{b}_c$, extracts a representational feature vector \mathbf{b}_c from the clinical image \mathbf{x}_c . Similarly, the dermoscopy ResNet, $C_d(\mathbf{x}_d) = \mathbf{b}_d$, outputs feature vector \mathbf{b}_d corresponding to the dermoscopy image \mathbf{x}_d . Both \mathbf{b}_c and \mathbf{b}_d have equal dimensions, denoted as N . Next, two representation features \mathbf{b}_c and \mathbf{b}_d , are element-wise added to construct a shared feature vector $\mathbf{b}_f \in \mathbb{R}^N$. Subsequently, features \mathbf{b}_c , \mathbf{b}_d , and \mathbf{b}_f corresponding to three branches – two single-modality branches and one cross-modality branch – are fed into three fully connected MLP (multi-layer perceptron) models M_c , M_d , and M_f to obtain predictions, $\hat{\mathbf{y}}_c$, $\hat{\mathbf{y}}_d$, and $\hat{\mathbf{y}}_f$, respectively. Thus, M_c and M_d enable learning on separate single modalities, while M_f learns cross-modality features representation. Finally, FusionNet is trained end-to-end to minimize the overall loss of the three sets of predictions.

$$\begin{aligned}
 L_c &= \sum_{i=1}^{|\mathcal{Y}|} \text{CE}(\hat{\mathbf{y}}_c^{(i)}, \mathbf{y}^{(i)}) & L_d &= \sum_{i=1}^{|\mathcal{Y}|} \text{CE}(\hat{\mathbf{y}}_d^{(i)}, \mathbf{y}^{(i)}) \\
 L_f &= \sum_{i=1}^{|\mathcal{Y}|} \text{CE}(\hat{\mathbf{y}}_f^{(i)}, \mathbf{y}^{(i)}) & L &= L_c + L_d + L_f
 \end{aligned}$$

Specifically, within each branch, we compare the predictions of the MLP model with the ground truth for each of the 8 different classification tasks using the cross-entropy loss function, denoted as CE. The final loss is obtained as a linear sum of individual branch losses with uniform weights. The gradient of the loss is then backward propagated through all MLPs and ResNets, enabling end-to-end training of FusionNet.

B. SplitFusionNet: A U-shape Split of FusionNet

There are multiple ways to split a DNN model. In SplitFusionNet, we split the FusionNet model into three sequential sub-networks: client-head, server-body, and client-tail (Figure 4) – the purpose being keeping both input images and labels on the client side.

Split Configuration. The client-head contains the convolutional blocks of two ResNets C_c and C_d , corresponding to two image modalities. The client-tail comprises the linear classification layer of three MLPs M_c , M_d , and M_f , corresponding to three branches. The server-body contains the remaining intermediate layers of FusionNet. Our choice for this split configuration is motivated by the following two reasons.

- **Reduction of Computational Load in Client.** We prioritize minimizing the computational load on the client while

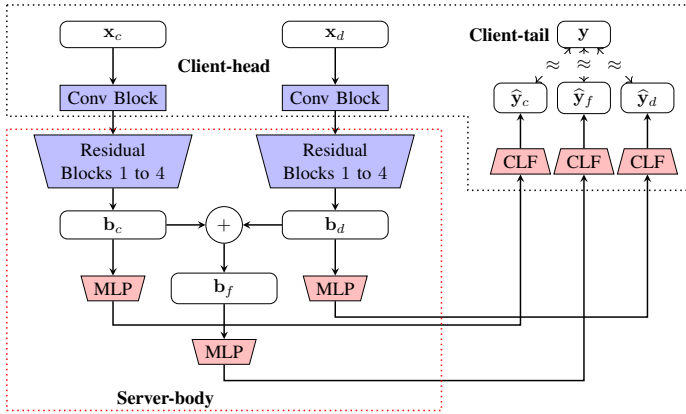


Fig. 4: The model architecture of SplitFusionNet, which is partitioned between a client (black dotted region) and a server (red dotted region). The client-head network comprises the convolutional blocks of two ResNet50 models, each corresponding to one of the two image modalities. The client-tail network consists of the linear classification layer of three MLP branches: two for the input image modalities and one fusion branch from the cross-modality. The server-body network includes the residual blocks of two ResNet50 models and three MLP networks (without the classification layer) corresponding to three branches.

ensuring data privacy through a small set of DNN layers. As a result, the client-head hosts the convolutional block of ResNets, comprising less than 0.04% of the total trainable parameters in SplitFusionNet. The convolutional layer might still be susceptible to model inversion attacks [41, 42] – we do not focus such attacks in the paper. Similarly, the client-tail consists of the classification layer of MLPs, representing less than 0.02% of trainable parameters.

- **Reduction in Transferred Data.** In SplitFusionNet, we leverage the higher computational resources of the server. As such, SplitFusionNet encounters a reduced efficiency when large data (representation features and gradients) are transferred between the client and server. To reduce transferred data, the cut layer between the client-head and server-body (the output of the convolutional block) produces the smallest size representation features compared to splitting at other layers within the ResNet model. Likewise, between the server-body and client-tail, the classification layer generates the smallest size representation features.

We additionally experimented with alternative splitting configurations: placing the first residual block of ResNets in the client-head and/or transferring all MLP layers to the client-tail. However, none of these configurations yielded superior performance in the training pipeline, as discussed earlier.

Efficient Transfer of Representation Features and Gradients. We employ lossless compression and subsequent decompression on representation features and gradients before transmitting them over the internet – the goal is to minimize transferred data without affecting training. Typically, a lossless compression algorithm offers a hyper-parameter to determine the trade-off between compression ratio and computational time for conversion, which we empirically study in Section IV.

IV. EMPIRICAL PERFORMANCE ANALYSIS

We conduct an empirical evaluation of SplitFusionNet. Specifically, we delve into the experimental setup, the objectives of the experiments, and the obtained experimental results.

Experimental Setup. We develop a prototype of SplitFusionNet using Python 3.7. In the client-server split learning setup, the client-side network operates on an AMD EPYC™ 7742 CPU with 512 GB memory, while the server-side network utilizes an RTX 3090 GPU with 24 GB GDDR6X

TABLE I: The count of trainable parameters shared between the client and server differs in non-split and split implementations. In the non-split implementation, the entire model resides within the client. Here, ‘H’, ‘T’, and ‘B’ stand as abbreviations for head, tail, and body, respectively.

Method	Client (H)	Server (B)	Client (T)	Total
FusionNet	50373448	—	—	50373448
SplitFusionNet	19072	50345088	9288	50373448

memory, hosted on two separate physical machines¹. Table I details the parameter distribution between the client and server, where the server manages over 99% of the trainable parameters. Communication between the client and server is established using Python’s Socket API. Within SplitFusionNet, we integrate the Python binding of the Zstandard (or zstd) lossless compression algorithm developed by [38]. We configure the compression level hyper-parameter in Zstandard to ‘fast’ (level = 1). The dataset for the seven-point checklist criteria comprises 1011 multi-modal images, split into training, validation, and test datasets containing 413, 203, and 395 samples, respectively. Each competing method is executed for 100 training epochs, repeated three times with random model initialization, and we report the corresponding mean and standard deviation. Subsequently, we discuss the objectives of our empirical study.

- 1) **RQ1 (Predictive Performance):** How does SplitFusionNet compare with FusionNet in predictive performance?
- 2) **RQ2 (Training Efficiency):** How does SplitFusionNet compare with FusionNet in total training pipeline time including training time, data preparation/compression time, and communication time?

Summary of Experimental Results: In experiments, SplitFusionNet achieves an equal prediction accuracy to FusionNet in the multi-modal and multi-label skin lesion classification task. Moreover, SplitFusionNet utilizing lossless compression and decompression requires only 49% of the training pipeline time taken by FusionNet. *Therefore*, SplitFusionNet

¹In a parallel implementation, we consider the SambaNova RDU [43] as a server in place of the GPU. The GPU implementation provides a superior performance than the RDU and we report results for GPU only.

TABLE II: Results regarding test accuracy across 8 classification tasks indicate that, on average, SplitFusionNet (and SplitFusionNet* with lossless compression and decompression) achieves an equal prediction accuracy to FusionNet. The slight variance in accuracy can be attributed to the randomization inherent in DNN training.

Method	DIAG	PN	STR	PIG	RS	DAG	BWV	VS	Average
FusionNet	0.64 ± 0.00	0.58 ± 0.02	0.67 ± 0.01	0.64 ± 0.01	0.73 ± 0.01	0.54 ± 0.02	0.84 ± 0.01	0.79 ± 0.00	0.68 ± 0.00
SplitFusionNet	0.64 ± 0.01	0.60 ± 0.01	0.68 ± 0.01	0.62 ± 0.01	0.75 ± 0.01	0.55 ± 0.00	0.84 ± 0.01	0.79 ± 0.01	0.68 ± 0.00
SplitFusionNet*	0.65 ± 0.00	0.60 ± 0.00	0.66 ± 0.01	0.61 ± 0.02	0.75 ± 0.01	0.54 ± 0.02	0.84 ± 0.01	0.79 ± 0.01	0.68 ± 0.00

TABLE III: The average per-epoch training pipeline time for FusionNet and SplitFusionNet is in seconds (and in percentage). Compared to FusionNet, SplitFusionNet demands lower training time. However, SplitFusionNet results in an increased overall training pipeline time due to higher communication costs. By adopting lossless compression before data transfer and subsequent decompression after transfer, SplitFusionNet* exhibits the lowest training pipeline time among the considered methods.

Method	Training Client	Compression Client	Communication Client to Server	Training Server	Compression Server	Communication Server to Client	Training Pipeline Time
FusionNet	76.37 (100%)	0.00 (0%)	0.00 (0%)	0.00 (0%)	0.00 (0%)	0.00 (0%)	76.37 (100%)
SplitFusionNet	2.93 (3%)	0.12 (0%)	73.28 (80%)	1.97 (2%)	0.50 (1%)	11.94 (13%)	91.58 (100%)
SplitFusionNet*	2.89 (8%)	2.50 (7%)	16.69 (44%)	1.90 (5%)	2.50 (7%)	10.50 (28%)	37.71 (100%)

TABLE IV: The average per-epoch size of transferred data is measured in Megabytes (MB). Compared to SplitFusionNet, SplitFusionNet*, equipped with lossless compression, reduces the size of transferred data, leading to more efficient communication. Column 6 (resp. column 7) denotes the sum of columns 2 and 5 (resp. columns 3 and 4).

Method	Client-head Representation	Client-head Gradient	Server-body Representation	Server-body Gradient	Transferred Data Client to Server	Transferred Data Server to Client	Total Transferred Data
SplitFusionNet	632.41	632.41	0.61	0.41	632.82	633.02	1265.84
SplitFusionNet*	420.60	594.36	0.45	0.38	420.98	594.81	1015.79

demonstrates an equal prediction performance while requiring less training pipeline time. We next discuss our experimental results in details.

A. Multi-label and Multi-modal Prediction Performance

In Table II, we present the average test accuracy across 8 classification tasks for various models. Both SplitFusionNet and SplitFusionNet* (SplitFusionNet implementing loss-less compression and decompression) train an identical DNN model as FusionNet, except the trained model is shared between two physical machines through the client and the server. Consequently, the prediction accuracy of all three models is expected to be similar, as evident in our experiments. Moreover, all models share the same hyperparameters, and we refrain from performing hyperparameter tuning—our focus is on demonstrating collaborative privacy-preserving training for multi-modal medical image classification rather than enhancing accuracy through hyperparameter tuning or neural architecture search. In fact, SplitFusionNet can potentially benefit from the future improvement of the non-split model FusionNet. *Therefore, SplitFusionNet achieves an equal predictive performance with FusionNet.*

B. Efficiency in Training

Training Pipeline Time. In Table III, we present the average training pipeline time per epoch between FusionNet and SplitFusionNet. FusionNet requires approximately 76 seconds to complete a training epoch on a CPU. On the other hand, SplitFusionNet delegates the computational workload to the

server on a GPU, reducing the training time to around 4.9 seconds, accounting for 7% of FusionNet’s training time. However, SplitFusionNet incurs communication costs for transferring representation features and gradients between the client and server, resulting in approximately 85 seconds, summing up to a total training pipeline time of 92 seconds—significantly higher than FusionNet. Conversely, SplitFusionNet* integrates lossless compression before transferring representation features and gradients, resulting in a total training pipeline time of around 38 seconds, which constitutes 41% and 49% of the total training pipeline time of SplitFusionNet and FusionNet, respectively. *Hence, SplitFusionNet with lossless compression facilitates efficient and private training for multi-modal medical image classification compared to the non-split method.*

Transferred Data. In Table IV, we illustrate the size of transferred data between the server and the client. In comparison to SplitFusionNet, SplitFusionNet* with lossless compression effectively reduces the size of both representation features and gradients exchanged between the client and the server. Particularly, representation features are compressed more than corresponding gradients, for instance, 420.60 MB versus 594.36 MB for client-head, using the same hyperparameter for compression – possibly due to higher redundancy in representation features than gradients. Consequently, the achieved compression ratio stands at approximately $\frac{1265.84}{1015.79} = 1.25$, resulting in a reduction of communication time from 93% in SplitFusionNet to 72% in SplitFusionNet* (see Table III). *Therefore, lossless compression emerges as an effec-*

tive approach to decrease transferred data in split learning, thereby diminishing the server-client communication cost.

V. CONCLUSION

We study multi-modal and multi-label skin lesion classification within a split learning framework, aiming to enable collaborative yet privacy preservation and efficient training. Our proposed framework, SplitFusionNet, operates on a u-shape split learning paradigm, wherein sensitive patients' image data and classification labels are retained on the client side, while the computationally intensive mid-layer training is offloaded to a powerful server. SplitFusionNet integrates lossless compression to mitigate communication costs between the client and the server. In our experimental validation, SplitFusionNet achieves equivalent predictive performance compared to the non-split centralized learning framework, while requiring only 49% of the overall training pipeline time. In future work, we plan to extend SplitFusionNet to a multi-client setting involving multiple data providers and further explore enhanced communication protocols between the client and server.

Acknowledgments. The work is supported by RIE2025 Industry Alignment Fund – Industry Collaboration Project (IAF-ICP) (Award No: I2301E0020), administered by A*STAR.

REFERENCES

- [1] V. Jain and J. M. Chatterjee, "Machine learning with health care perspective," *Cham: Springer*, pp. 1–415, 2020.
- [2] J. Futoma, M. Simons, T. Panch, F. Doshi-Velez, and L. A. Celi, "The myth of generalisability in clinical research and machine learning in health care," *The Lancet Digital Health*, vol. 2, no. 9, 2020.
- [3] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *TNNLS*, 2021.
- [4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, pp. 770–778, 2016.
- [5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.
- [6] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annual review of biomedical engineering*, vol. 19, 2017.
- [7] J. S. Duncan and N. Ayache, "Medical image analysis: Progress over two decades and the challenges ahead," *TPAMI*, vol. 22, no. 1, 2000.
- [8] D. Steinkraus, I. Buck, and P. Y. Simard, "Using GPUs for machine learning algorithms," in *Proc. ICDAR*, pp. 1115–1120, IEEE, 2005.
- [9] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. ISCA*, 2017.
- [10] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha, "Efficient machine learning for big data: A review," *Big Data Research*, vol. 2, no. 3, pp. 87–93, 2015.
- [11] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *IEEE SP*, pp. 19–38, IEEE, 2017.
- [12] E. Hesamifard, H. Takabi, M. Ghasemi, and R. N. Wright, "Privacy-preserving machine learning as a service," *PETS*, 2018.
- [13] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *ACM CCS*, 2017.
- [14] M. Al-Rubaie and J. M. Chang, "Privacy-preserving machine learning: Threats and solutions," *IEEE SP*, vol. 17, no. 2, pp. 49–58, 2019.
- [15] Y. Wang, H. Fu, R. Kanagavelu, Q. Wei, Y. Liu, and R. S. M. Goh, "An aggregation-free federated learning for tackling data heterogeneity," in *CVPR*, 2024.
- [16] B. Sliwa, N. Piatkowski, and C. Wietfeld, "Limits: Lightweight machine learning for iot systems with resource limitations," in *IEEE ICC*, 2020.
- [17] A. Harlap, D. Narayanan, A. Phanishayee, V. Seshadri, N. Devanur, G. Ganger, and P. Gibbons, "Pipedream: Fast and efficient pipeline parallel dnn training," *arXiv:1806.03377*, 2018.
- [18] C.-C. Chen, C.-L. Yang, and H.-Y. Cheng, "Efficient and robust parallel dnn training through model parallelism on multi-gpu platform," *arXiv:1809.02839*, 2018.
- [19] Y. Hu, D. Niu, J. Yang, and S. Zhou, "FDML: A collaborative machine learning framework for distributed features," in *Proc. KDD*, 2019.
- [20] Y. Liu, Y. Kang, X. Zhang, L. Li, Y. Cheng, T. Chen, M. Hong, and Q. Yang, "A communication efficient collaborative learning framework for distributed features," *arXiv preprint arXiv:1912.11187*, 2019.
- [21] M. G. Poirot, P. Vepakomma, K. Chang, J. Kalpathy-Cramer, R. Gupta, and R. Raskar, "Split learning for collaborative deep learning in health-care," *arXiv preprint arXiv:1912.12115*, 2019.
- [22] P. Vepakomma, O. Gupta, T. Swedish, and R. Raskar, "Split learning for health: Distributed deep learning without sharing raw patient data," *arXiv preprint arXiv:1812.00564*, 2018.
- [23] S. Abuadba, K. Kim, M. Kim, C. Thapa, S. A. Camtepe, Y. Gao, H. Kim, and S. Nepal, "Can we use split learning on 1d cnn models for privacy preserving training?," in *Proc. ASIACCS*, 2020.
- [24] C. Thapa, M. A. P. Chamikara, and S. A. Camtepe, "Advancements of federated learning towards privacy preservation: from federated learning to split learning," *Federated Learning Systems: Towards Next-Generation AI*, pp. 79–109, 2021.
- [25] J. Kim, S. Shin, Y. Yu, J. Lee, and K. Lee, "Multiple classification with split learning," in *Proc. SMA*, pp. 358–363, 2020.
- [26] H. Hermessi, O. Mourali, and E. Zagrouba, "Multimodal medical image fusion review: Theoretical background and recent advances," *Signal Processing*, vol. 183, p. 108036, 2021.
- [27] M. F. Simoes, J. S. Sousa, and A. C. Pais, "Skin cancer and new treatment perspectives: A review," *Cancer letters*, vol. 357, 2015.
- [28] I. A. Ozkan and M. Koklu, "Skin lesion classification using machine learning algorithms," *IJISAE*, vol. 5, no. 4, pp. 285–289, 2017.
- [29] J. Yap, W. Yolland, and P. Tschandl, "Multimodal skin lesion classification using deep learning," *Experimental dermatology*, vol. 27, no. 11, pp. 1261–1267, 2018.
- [30] M. A. Kassem, K. M. Hosny, R. Damaševičius, and M. M. Eltoukhy, "Machine learning and deep learning methods for skin lesion classification and diagnosis: a systematic review," *Diagnostics*, vol. 11, 2021.
- [31] C. M. Balch, J. E. Gershenwald, S.-j. Soong, J. F. Thompson, M. B. Atkins, D. R. Byrd, A. C. Buzaid, A. J. Cochran, D. G. Coit, S. Ding, et al., "Final version of 2009 AJCC melanoma staging and classification," *Journal of clinical oncology*, vol. 27, no. 36, p. 6199, 2009.
- [32] J. Kawahara, S. Daneshvar, G. Argenziano, and G. Hamarneh, "Seven-point checklist and skin lesion classification using multitask multimodal neural nets," *JBHI*, vol. 23, no. 2, pp. 538–546, 2018.
- [33] G. Argenziano, C. Catrixalà, M. Ardigo, P. Buccini, P. De Simone, L. Eibenschutz, A. Ferrari, G. Mariani, V. Silipo, I. Sperduti, et al., "Seven-point checklist of dermoscopy revisited," *British Journal of Dermatology*, vol. 164, no. 4, pp. 785–790, 2011.
- [34] P. Tang, X. Yan, Y. Nan, S. Xiang, S. Krammer, and T. Lasser, "FusionM4Net: A multi-stage multi-modal learning algorithm for multi-label skin lesion classification," *Medical Image Analysis*, vol. 76, 2022.
- [35] Z. Ge, S. Demyanov, R. Chakravorty, A. Bowling, and R. Garnavi, "Skin disease recognition using deep saliency features and multimodal learning of dermoscopy and clinical images," in *Proc. MICCAI*, 2017.
- [36] L. Bi, D. D. Feng, M. Fulham, and J. Kim, "Multi-label classification of multi-modality skin lesion via hyper-connected convolutional neural network," *Pattern Recognition*, vol. 107, p. 107502, 2020.
- [37] Y. Wang, Y. Feng, L. Zhang, J. T. Zhou, Y. Liu, R. S. M. Goh, and L. Zhen, "Adversarial multimodal fusion with attention mechanism for skin lesion classification using clinical and dermoscopic images," *Medical Image Analysis*, vol. 81, p. 102535, 2022.
- [38] Y. Collet and M. Kucherawy, "Zstandard compression and the application/zstd media type," tech. rep., 2018.
- [39] A. Gupta, A. Bansal, and V. Khanduja, "Modern lossless compression techniques: Review, comparison and analysis," in *Proc. ICECCT*, 2017.
- [40] V. K. Goyal, A. K. Fletcher, and S. Rangan, "Compressive sampling and lossy compression," *IEEE SPM*, vol. 25, 2008.
- [41] O. Li, J. Sun, X. Yang, W. Gao, H. Zhang, J. Xie, V. Smith, and C. Wang, "Label leakage and protection in two-party split learning," *arXiv preprint arXiv:2102.08504*, 2021.
- [42] D. Pasquini, G. Ateniese, and M. Braschi, "Unleashing the tiger: Inference attacks on split learning," in *Proc. CCS*, pp. 2113–2129, 2021.
- [43] R. Prabhakar and S. Jairath, "Sambanova sn10 rdu: Accelerating software 2.0 with dataflow," in *HCS*, pp. 1–37, IEEE, 2021.