Towards Optimization of Privacy and Accuracy for Differential Privacy in Image Classification

Chad Schwenke*, Dilochan Karki†
Colorado State University
Email: *chad.schwenke@colostate.edu, †dilochan.karki@colostate.edu

***Abstract*—Differential privacy(DP) has become a popular approach to providing privacy guarantees and preventing privacy leakage of datasets that are used to train deep learning models. It does so by introducing random sampled noise into the neural networks during training. However, this affects the inference capability of the model on the training dataset and reduces the accuracy at which the model can predict on test data. Less strict implementations of differential privacy can be implemented to obtain private models with higher accuracy. But such methods have shown to provide weaker privacy guarantees. Additionally, training neural networks with DP is more computationally intensive than training normal models. This prompts a discussion on the privacy vs. utility trade-off of implementing DP to train privacy-preserving deep learning models. Our work shows how transfer learning and efficient gradient accumulation mechanisms can help achieve accuracy closer to non-private counterparts. Furthermore, since obtaining pretrained models for many image classification problems can be difficult in practice, we explore the benefits of using publicly available pretraining datasets that are out of distribution with the private training datasets and see if it can be considered a general approach to solving privacy-preserving problems.**

I. Introduction

Despite the impressive utility of neural networks throughout various computation tasks, many researchers have found that they are vulnerable to privacy attacks that can reveal information about the data used during training [1]. These attacks are especially a concern in situations in which the training set contains sensitive data such as personal identifiable information (PII) or proprietary knowledge for example.

The most common of these privacy attacks, the membership inference attack, is where an adversary attempts to determine if a specific training sample was used during the training of the neural network. Moreover, another kind of attack exists called the model inversion attack in which an adversary attempts to reconstruct a training sample using the predictions of a model. Additionally, others have explored attacks that can infer the parameters and hyperparameters of the model as well as other properties of the training data [2].

The most popular method at preventing these attacks is training the model with a privacy guarantee. Differential privacy (DP) was first demonstrated as an effective method for ensuring this privacy guarantee within neural networks in 2016 called deferentially private stochastic gradient descent (DP-SGD) [3]. Since then, much research has been done on the application of DP to neural networks for various tasks [1], [4]. Despite the protections offered by training with DP, model utility will generally suffer as a result. This effect has been called the privacy-utility trade off and is what researchers attempt to minimize throughout DP research in neural networks [1].

In the domain of image classification, DP is especially important as images can contain highly revealing information about an entity or subject. Past works on differentially private image classification have recognized that private learning on a pretrained model can provide substantial benefits to model accuracy and utility while still maintaining the same privacy budget [5]. Various researchers have demonstrated that the use of pretrained models can allow for differentially private training with accuracy that is near state of the art (SOTA) models with non-private training [6], [7], [8].

In more recent works, researchers have demonstrated that the use of pretrained models are suitable for differentially private learning even when the pretraining dataset comes from a different distribution than the training dataset [9]. This discovery is important because public in distribution datasets for a private training task can be difficult to obtain in real world scenarios. This work aims to analyze the effectiveness of using models pretrained on public datasets for private learning on datasets that are out domain of or lack similarity to the public pretraining dataset.

In the following section, we provide a background on differential privacy, deep learning with differential privacy, gradient accumulation and transfer learning. Section 3 provides brief introductions to related works on differentially private learning and ways to achieve highly accurate private models. Section 4 explains our approach and implementation along with our empirical results and compares them with other works. Section 5 describes any future works that can be further done by expanding upon this work. Finally, section 6 concludes with a discussion of our results.

II. Background

A. Differential Privacy

Differential Privacy was first introduced in [10] as a way to provide privacy for statistical databases such that it provides useful information like averages about the samples the database contains without revealing the attributes of the actual samples. An adversary with access to the database shouldn't be able to learn whether an individual sample is present or not through queries without access to the database. In [10], it is formally defined as,

A randomized algorithm A satisfies ϵ-differential privacy if for any neighboring databases D and D', and for any subset B of the range of R, we have:

$$\Pr[A(D) \in B] \leq e^{\epsilon} \cdot \Pr[A(D') \in B]$$

To simplify, it states that for any randomized algorithm A, it provides privacy guarantees for any subset B of the range of A, given that the probability that A(D) outputs a value in S is at most e^{ϵ} times the probability that A(D') outputs a value in S. Neighbouring databases differ from each other by a single individual sample of data. The parameter ϵ is a positive real number that controls the amount of noise that is added to the output of the algorithm A. This means, that with differential privacy, an adversary with access to auxiliary information on the database won't have information on individual data revealed from the output of the algorithm.

Differential Privacy is achieved by adding appropriately sampled random noise to the answer of a query function that summarizes attributes of data entirely or subsets as requested by a user. The magnitude of noise added is computed based on the sensitivity of the query function i.e. the range by which output changes with changes in input of the function. We can achieve ϵ-differential privacy based on the amount of noise added to the query function proportional to its sensitivity. A smaller value of ϵ means a higher level of protection provided by the randomised query function i.e. stronger privacy guarantees. Hence, ϵ can be defined as the privacy budget i.e. the maximum threshold to which information can be disclosed. However, there is a trade-off in implementing differentially private algorithms. Adding sampled noise to the query function output reduces its accuracy which may make it less useful for statistical analysis while making it harder for adversaries to infer information from the database. To increase accuracy, the amount of noise added to the output can be reduced but at the cost of reduced privacy.

The choice of noise distribution plays an important role in balancing this trade-off between privacy and accuracy. A distribution that adds too much noise can result in inaccurate query output, whereas a distribution that adds too little noise can compromise privacy. The choice of distribution also depends on the specific application and the desired level of privacy protection. Commonly used distributions are Laplace and Gaussian distributions. The former adds noise proportional to the sensitivity of the query while the latter adds noise proportional to the square root of the sensitivity of the query. Laplace distribution [11] provides stronger privacy guarantees while allowing for accurate query outputs to a certain extent compared to Gaussian distribution which focuses on accuracy but at the cost of weaker privacy guarantees [4]. A variant with Gaussian noise distribution is called (ϵ,δ)-differential privacy, a relaxation of ϵ-DP. The term δ signifies the privacy protection given by Gaussian distribution. In [12], it is formally defined as,

A randomized algorithm A is called (ϵ, δ)-differentially private if for all neighbouring databases D and $D' \in D_n$. We have, for all X $\subseteq$ B, where B is the set of all possible outputs:

$$\Pr[A(D) \in X] \leq e^{\epsilon} \cdot \Pr[A(D') \in X] + \delta$$

The term δ signifies the upper bound on the probability of privacy leakage below specified ϵ i.e. relaxation of privacy. Its value is inversely proportional to the size of the database. This variant is beneficial in cases where ϵ-DP is too restrictive and affects the accuracy of the output function.

B. Deep Learning with Differential Privacy

Using deep learning with differential privacy was first introduced in [3] as a mechanism to introduce privacy guarantees into machine learning models that are usually trained in publicly accessible data. The primary motivation for this algorithm's implementation is to ensure privacy of the input data used to train the neural network. It controls the influence of training data on the model with differential privacy applied to the gradients that are used to update the weights of a neural network during back-propagation while training.

Normally, deep learning optimization algorithms are implemented as an iterative approach to minimize the error function of a machine learning model while training. The parameters of the model are updated to optimize the error function which is calculated as the difference between predicted output of the model and actual output. At every iteration, a gradient of the error function is computed for a small random subset of training data, known as batch. The gradients for weights of each unit of the model point towards the steepest direction of increase for the error function. Using this, the models are updated in the reverse direction with small steps called the learning rate. For a certain number of epochs or iterations, this is repeated until the model converges i.e. a state with least value of error function is reached. At this state, the model has effectively learned features of the training data and now can be use for prediction on new unseen data.

To achieve differential privacy, random Gaussian noise is sampled and added to the gradients of a multi-layered neural network. However, adding noise to the gradients can cause them to be very large and take larger step sizes while updating the weights of the models. This leads to issues in the convergence of the model and poor performance as well. To prevent this, The gradients need to be clipped i.e. set a limit on the sensitivity of each gradients so as to limit the influence of individual training data on the resulting gradients. Clipping is achieved by defining the size of mini-batch of samples that are used to calculate the gradients. Then, the gradient for each example of the mini-batch is clipped to a maximum Euclidean norm and normalized. This way, gradients are clipped on minimal set of examples to reduce overhead and maximize utility by reducing the negative effect of clipping. When the mini-batch size is same as the batch size, the gradient of the error function is computed for each individual data point in the batch and clipped to the maximum norm. If the norm of the calculated gradient is less than the defined maximum threshold, it is kept the same, else it is scaled down to the defined maximum norm. Clipping gradients computed over the average of a batch used to train can cause information loss and using mini-batches helps alleviate this.

C. Gradient Accumulation and Clipping

Deep Learning for image classification generally involve large datasets of images and are computationally extensive. In the case of differentially private deep learning, computational costs are further increased by the need for larger batch sizes during training. For a given privacy budget, the amount of noise added depends on the sensitivity of gradients which is inversely proportional to the batch size. Previous works such as [13], [14] have also shown a significant increase in accuracy with larger batch sizes to train private deep learning models. Using larger batch sizes with gradient clipping can cause memory overhead as gradients for all of the samples are clipped. This can be alleviated to some extent with gradient accumulation, where the gradients of multiple mini-batches are accumulated and averaged to update the weights. It has been observed that this has been beneficial for efficient large batch training for image classification [14], [8], [13].

To further reduce overhead from gradient clipping in large batches, a memory-efficient strategy called ghost clipping was introduced in [15] for DP training of large language models. It clipped the gradients of a subset of parameters and scaled the gradient vectors for the remaining to compensate for the clipped ones. In addition to that, it applied a single clipping threshold to all of the samples of a mini-batch rather than computing the maximum norm for each gradient while clipping them individually. The clipping threshold is estimated from the maximum norm of the gradients taken from a subset of the samples from a mini-batch This allowed for larger training batch sizes and better performance while training the model to convergence.

In [7], the ghost clipping strategy was modified into mixed-ghost clipping and implemented to efficiently train private deep learning models for image classification. It differs from ghost clipping by using a weighted sum of the maximum norm of gradients in the current and previous mini-batches as the clipping threshold. The ratio of the number of samples present in the current and previous mini-batches is used to calculate the weights. This allows for efficient training as the clipping mechanism adapts to the changing distribution of gradients across various mini-batches, which can be more prominent in larger batch sizes. It is demonstrated through image classification experiments with large convolutional neural networks (CNNs) in [7] that more accurate models can be obtained with this approach. Additionally, the larger training times for private models are also significantly reduced by a factor 2 with accuracy levels closer to their non-private counterparts i.e. training models without DP.

The authors of [7] have open-sourced a privacy engine in PyTorch [16]. that facilitates mixed-ghost clipping for DP training. It has abstractions that allow usage of any optimizer algorithms and calculate the noise multiplier based on hyper-parameter values provided. It has been show to be faster and efficient than the prevalent privacy engine of Opacus library which has been the state-of-the-art for DP training.

D. Transfer Learning with pretrained models

Machine learning models used for tasks such as image classification are complex and generally contain multiple layers of neurons. CNNs are generally used for image-based classification. More advanced implementations of it for coloured images and other complex objects contain highly sophisticated models that can take quite a long time to train. Additionally, finding the right set of hyperparameters for optimal performance of the model for such a complex model is a daunting task as well. Thus, an approach to using pretrained model whose weights are saved and reused for further classification is beneficial. Currently, the notion of using pretrained models is quite popular as it helps enhance the convergence time of the model for the particular dataset. Generally, a pretrained model is a saved model that was trained on a large dataset (usually available in public). Then, the pretrained model is used as it is for classification tasks or its trained again for the dataset we are trying to use it for. With transfer learning, the weights of the pretrained model gives faster convergence time and better accuracy when used with dataset of the given use case. There are generally two common ways of training a pretrained model. With fine-tuning, a few or entire layers of the pretrained model are trained again with the dataset to get a better representation of the features contained in the dataset. The other approach is to simply add another classifier on top of the pretrained model to get a model that can better classify the training dataset.

III. Related Works

Differential privacy (DP) has been the most popular researched method for adding privacy within neural networks. DP for deep learning was first explored within a practical system in [17] where the researchers facilitated collaborative deep learning with selective sharing of model parameters during training. They then incorporated differential privacy into these shared parameters and achieved considerable accuracy and privacy in image classification experiments with the MNIST image dataset.

First introduced in 2016 [3], the Differentially Private SGD (DPSGD) algorithm demonstrated that neural networks could add differential privacy during the training process and still maintain usable accuracy. They introduced DP based noises on gradients before they are used for updates and tracked the privacy spending in a process they titled moments accountant. Experiments were conducted on image classification datasets to validate their implementation in Tensorflow. Expanding on this work in 2017 [18], DPSGD was shown to be applicable in collaborative learning using secure multiparty communication. Another work in 2017 [19] proposed an approach called Private Aggregation of Teacher Ensembles (PATE) in which the moments accountant [3] methods were used to create teacher models using disjointed datasets that taught student models with added noise. It was further scaled up for performance in [20] for real world datasets and uses Gaussian noise instead of Laplace to obtain aggregation mechanisms. These mechanisms provide more accurate consensus in teachers' votes, which is crucial to scale learning tasks for a large number of output

classes. An alternative to the DPSGD algorithm was created called the Adaptive Laplace Mechanism (AdLM). This method is more efficient because it separates the noise insertion from the amount of epochs allowing for more thorough training. Additionally, it allows for more noise insertion to less relevant features of the model allowing for better privacy protection.

Through DP, noise added to the model at each epoch during training causes a significant hindrance to its convergence. DP-based machine learning models have been observed to have lower accuracy and more convergence times compared to their non-DP counterparts [21]. Most of the current literature on differentially private learning is trying to find the optimal privacy budget that gives decent accuracy while guaranteeing privacy protection for any given deep learning task [4]. Researchers are working on finding answers to the privacy and accuracy trade-off with DP in deep learning.

To obtain effective accuracy for private models, a different approach to obtaining them is also being explored. With pretrained models, a significant accuracy gain can be obtained on deep learning models for image classification tasks [5]. They observed that the benchmarks for private models were comparatively less favourable compared to when they were pretrained on public data. Along with empirical results, they contribute with some theoretical demonstrations that pretrained models can be exploited to better handle the accuracy and privacy tradeoffs DP private models come with.

Recent works on image classification for medical images provide compelling evidence that using pretrained models on public data is a practical approach to obtaining accurate models for private medical image datasets [9], [8]. They have also demonstrated that pretrained models generalize well to aid private training even when models from public datasets are used that are out of distribution with the private dataset [9]. Similarly, a new way to train DP models with semi-private learning was introduced in [22] using unlabelled public data and labelled private data. With a mix of principal component analysis and differentially privacy, they were able to obtain (ϵ,δ)-DP under a tight privacy budget even with a difference in the distribution of pretraining and private data. These works provide results that show prospects of using DP effectively on real-world datasets.

Works on improving the efficiency and convergence time of DP models such as [7] have been able to significantly reduce the training time and memory head although, without any effects on accuracy. They introduce the mixed ghost-clipping approach to reduce the sensitivity of the gradient functions during noise insertion and show it to be a much more efficient approach used with pretrained models. There have been similar works that focus on considerations for optimal privacy budgets in the real world. [8] suggests that finding a privacy budget suitable for the real world is a significant task and depends on the kinds of images we are dealing with and the level of privacy guarantee required.

IV. Methodology

A. Experimental Setup

1) Hardware: For our experiments, we used the Falcon HPC Cluster at the Computer Science Department of Colorado State University (CSU). It is a high-performance computing (HPC) cluster designed to support both CPU and GPU jobs. It boasts a powerful hardware configuration, featuring 4 AMD EPYC 74F3 CPUs, 3 AMD Ryzen Threadripper 3960X CPUs, totaling 240 cores, 2.5 TB of RAM, 8 NVIDIA A100 GPUs, and 12 NVIDIA GeForce RTX 3090 GPUs, providing a total GPU memory of 928 GB. Accessible only from within the CSU network or through CSU's VPN off-campus, the cluster's login nodes serve as entry points for users, facilitating job preparation and submission. Slurm, a workload manager software is employed for scheduling and managing computational tasks on the compute nodes, with the CSU Computer Science network file system available to all nodes. The cluster comprises login nodes for user interaction, compute nodes for actual computing tasks, and a dedicated management node for cluster administration. Users work directly on login nodes, where they log in, access files, and submit jobs, while compute nodes handle the execution of these jobs, managed by the scheduler. A management node is reserved for administrative tasks and is not accessible to users. The cluster uses the open-source scheduler Slurm for managing compute jobs. Storage resources include CS network file system (NFS) Space, providing read/write access to NFS shares similar to other CS Linux systems, and Scratch Space, offering local SSD storage for temporary job files, accessible only for the job's duration and automatically purged after completion. The Falcon HPC Cluster serves as a robust and resourceful platform for computational tasks within the Computer Science Department.

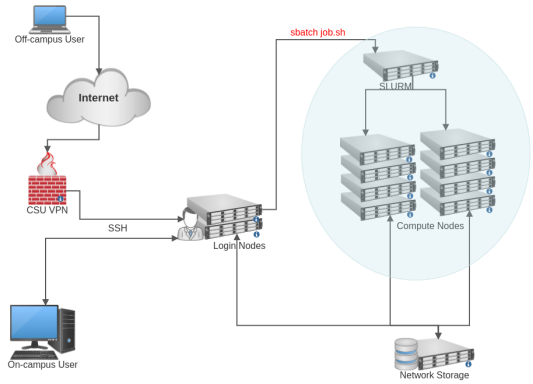

Fig. 1: Architecture of the Falcon Cluster

2) Software: We leveraged the publicly available code base known as *private_vision* for training our neural networks [7]. This library is based on PyTorch [16] and provides support for training neural networks with differential privacy using mixed ghost clipping. In general, we found the mixed ghost clipping to improve training time and reduce memory overhead.

To ensure that a model is compatible with DP training, this code base uses the *ModuleValidator* function from the Opacus library [23]. This function will check to see if each component of a model is compatible with Opacus training and automatically alter it if not. To ensure fairness between our private and non-private results, we chose to call this function in all of our experiments. As a result, all layers that were

compatible with our DP training were trained.

To introduce differential privacy into the models, *private_vision* uses an implementation of Renyi Differential Privacy (RDP) [24]. It is a relaxation of the standard Differential Privacy (DP) framework that provides a more flexible and powerful notion of privacy. It is based on the Renyi divergence, which is a measure of the difference between two probability distributions. In RDP, the privacy guarantee is expressed in terms of the Renyi divergence between the output distributions of a randomized mechanism applied to two adjacent inputs. The parameter controls the order of the Renyi divergence, and the parameter ϵ controls the privacy budget. A mechanism satisfies (α, ϵ)-RDP if the Renyi divergence between its output distributions is bounded by ϵ for all adjacent inputs. RDP is popular in various implementations of machine learning and data analysis applications.

3) Model and Pretraining Dataset: For each of our experiments, we leveraged the BEiT family of neural network models [25]. We selected this model because it demonstrated the best performance in our initial testing and can be easily loaded with pretrained weights from the Hugging Face platform [26]. Specifically, we used the *beit_base_patch16_224* model which has 102.6 million parameters and processes images with a square dimension of 224 pixels. It was noted in [9] that fine-tuning the whole model was sufficient to obtain reduced computational cost while making sure the transfer of weights is effective when using datasets that don't belong to the same distribution. Hence, for our experiments, after fixing the model layers with *ModuleValidator*, all of the layers are fine-tuned with the training dataset.

The pretrained dataset that we used in all experiments was ImageNet22K [27]. This dataset is publicly available and consists of millions of images with roughly 22 thousand classes of natural objects at the time of this writing. The BEiT models we obtained from Hugging Face [26] were first trained on ImageNet22K using DALL-E dVAE as visual tokenizer to perform self supervised masked image modeling. Lastly, the models were fine-tuned again on ImageNet22K.

B. Datasets

In our experiments, we tested three medical imaging datasets that are dissimilar to our pretraining dataset ImageNet [27]. For example ImageNet consists of natural images of everyday objects and has 22 thousand classes. Conversely, our datasets Pediatric Pneumonia Chest X-ray, DermNet, and HAM10000 have 2, 23 and 7 classes respectively. Additionally, we observed that the classes in these datasets had no overlap with any classes in ImageNet22K.

Data augmentations described below for each dataset were applied using the Torchvision [28] library in Python. In all datasets, we applied a random horizontal flip augmentation to all train data as a standard because we found this to generally help with overfitting in all experiments regardless of dataset or configuration. In Pediatric Pneumonia Chest X-ray and DermNet, the samples were normalized using the mean and standard deviation of ImageNet [27]. However, for HAM10000 we saw this decrease in performance so no normalization was performed.

1) Pediatric Pneumonia Chest X-ray: The Pediatric Pneumonia Chest X-ray dataset consists of labeled chest X-rays from 5,856 pediatric patients who are either healthy or have pneumonia [29]. This dataset distinguishes between bacterial and viral pneumonia for a total of 3 classes. However, to compare our results with other research that has tested this dataset [29], [30], we chose to use a version that treats bacterial and viral pneumonia as the same to provide a binary classification task.

Fig. 2: Pediatric Pneumonia Dataset

For both our private and non-private training cases, the only data augmentation that we added for training was a random resize crop to 224 square pixels. This augmentation simply takes a random crop of the image in any position and then resizes it to the desired dimensions [28]. Unlike [30], we found that adding random affine transformations decreased accuracy in both our private and non-private cases, so we chose not to use them.

Since this dataset consists of X-rays, we opted to augment our test data by cropping out the edges of each image. This was motivated by the fact that the relevant information in an X-ray is generally in the center where the lungs are located and the edges do not provide any useful information. This was achieved by first resizing the smaller side of each image to 256 pixels while maintaining its aspect ratio. Finally, we took a center crop of the image with a size of 224 square pixels before running it through our test function. We found this to improve the test accuracy in our experiments for this dataset.

Class	Number of Samples
Normal	1349
Pneumonia	3883

Fig. 3: Sample Distribution of Pneumonia Training Dataset

Class	Number of Samples
Normal	234
Pneumonia	390

Fig. 4: Sample Distribution of Pneumonia Test Dataset

2) DermNet: DermNet is a dataset consisting of various labeled images of skin diseases with 23 total classes [31]. The total number of samples within this dataset is 19,559.

In the private training case, the data augmentations we added were random affine transformations as we found them to

be sufficient at preventing overfitting without the need for additional augmentations. These augmentations applied to each image consisted of a random rotation from -10 to 10 degrees, a translation from 0% to 10%, a re-scaling down from 0% to 10%, and shears of 0% to 5% degrees in each axis. For our

Fig. 5: DermNet Dataset

non-private training, overfitting was an issue when using only the random affine transformations. To better help our model generalize, we added the random resize crop to 224 square pixels augmentation used in our Pneumonia datasets. The combined effect of these augmentations prevented overfitting and allowed us to achieve our highest accuracy in non-private training. No data augmentations were applied to our test data in any case other than simply resizing the images to 224 square pixels to fit our BEiT model structure.

Class	Number of Samples
Acne and Rosacea Photos	840
Actinic Keratosis Basal Cell Carcinoma and oth...	1149
Atopic Dermatitis Photos	489
Bullous Disease Photos	448
Cellulitis Impetigo and other Bacterial Infect...	288
Eczema Photos	1235
Exanthems and Drug Eruptions	404
Hair Loss Photos Alopecia and other Hair Diseases	239
Herpes HPV and other STDs Photos	405
Light Diseases and Disorders of Pigmentation	568
Lupus and other Connective Tissue diseases	420
Melanoma Skin Cancer Nevi and Moles	463
Nail Fungus and other Nail Disease	1040
Poison Ivy Photos and other Contact Dermatitis	260
Psoriasis pictures Lichen Planus and related d...	1405
Scabies Lyme Disease and other Infestations an...	431
Seborrheic Keratoses and other Benign Tumors	1371
Systemic Disease	606
Tinea Ringworm Candidiasis and other Fungal In...	1300
Urticaria Hives	212
Vascular Tumors	482
Vasculitis Photos	416
Warts Molluscum and other Viral Infections	1086

Fig. 6: Sample Distribution of DermNet Training Dataset

3) HAM10000: HAM10000 is a collection of 10000 dermatoscopic images [32] used in the research of image classification for skin lesion detection. It contains images of pigmented skin lesions labeled with 7 different categories of diagnosis: Melanocytic nevi, Melanoma, Benign Keratosis-like lesions, Basal cell carcinoma, Actinic Keratoses, Vascular Lesions and Dermatofibroma. The authors of [32] acquired dermatoscopic images from various populations along with any corrections required from expert dermatologists. This dataset has been instrumental in advancing research on automated skin lesion classification. Upon inspection of the sample distribution for each of the categories, it was found that there was an uneven distribution, with a large number of samples belonging to the Melanocytic nevi and Melanoma categories. These majority classes cause the dataset to be

Class	Number of Samples
Acne and Rosacea Photos	312
Actinic Keratosis Basal Cell Carcinoma and oth...	288
Atopic Dermatitis Photos	123
Bullous Disease Photos	113
Cellulitis Impetigo and other Bacterial Infect...	73
Eczema Photos	309
Exanthems and Drug Eruptions	101
Hair Loss Photos Alopecia and other Hair Diseases	60
Herpes HPV and other STDs Photos	102
Light Diseases and Disorders of Pigmentation	143
Lupus and other Connective Tissue diseases	105
Melanoma Skin Cancer Nevi and Moles	116
Nail Fungus and other Nail Disease	261
Poison Ivy Photos and other Contact Dermatitis	65
Psoriasis pictures Lichen Planus and related d...	352
Scabies Lyme Disease and other Infestations an...	108
Seborrheic Keratoses and other Benign Tumors	343
Systemic Disease	152
Tinea Ringworm Candidiasis and other Fungal In...	325
Urticaria Hives	53
Vascular Tumors	121
Vasculitis Photos	105
Warts Molluscum and other Viral Infections	272

Fig. 7: Sample Distribution of DermNet Test Dataset

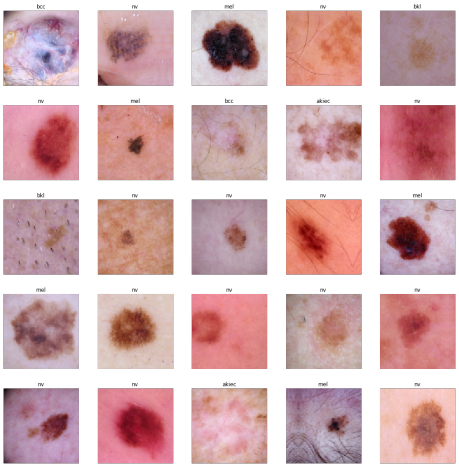

Fig. 8: HAM10000 Dataset

imbalanced resulting in the model not learning enough from other minority classes. Hence, for DP experiments, the data

cell_type	
Melanocytic nevi	5822
dermatofibroma	1067
Benign keratosis-like lesions	1011
Basal cell carcinoma	479
Actinic keratoses	297
Vascular lesions	129
Dermatofibroma	107

Fig. 9: Sample Distribution of HAM10000 Training Dataset

was augmented to produce a balanced dataset with an even sample distribution for each category. The samples belonging to minority classes were replicated based on a rate determined from the proportions of the classes. Lower or no rates were assigned to majority classes whereas higher rates were assigned to minority classes. Then, using a PyTorch *dataloader* [16], the training data was randomly sampled with transformations applied that augmented data with random horizontal and

Melanocytic nevi	883
Benign keratosis-like lesions	88
dermatofibroma	46
Basal cell carcinoma	35
Actinic keratoses	30
Vascular lesions	13
Dermatofibroma	8

Fig. 10: Sample Distribution of HAM10000 Test Dataset

vertical flips, random rotations by 20 degrees, and color jitters that changed the brightness, contrast and hue by 0.1 each. Finally, training and test dataset with a balanced distribution of the samples was obtained.

Melanocytic nevi	5822
Dermatofibroma	5350
Melanoma	5335
Vascular lesions	5160
Benign keratosis-like lesions	5055
Basal cell carcinoma	4790
Actinic keratoses	4455

Fig. 11: Sample Distribution of HAM10000 Augmented Training Dataset

C. Empirical Results

In each of our datasets, we train for the best accuracy with DP using Epsilons of 0.5, 1, and 2 for different values for various hyperparameters such as epochs, batch size and maximum gradient norm. The maximum gradient norm hyperparameter was found to not have a strong influence on the performance of the optimizer wrapped with the privacy engine. Other works have found this to be true as well [9]. Thus, the maximum gradient norm was set to 0.5 for all of our experiments with and without the pretrained model. Next, we use the optimal hyperparameters for each value of the privacy budget to rerun the experiments without using the pretrained weights from ImageNet [27]. Finally, we ran an experiment without DP and the pretrained weights from ImageNet. For private experiments, the learning rate was set to 1e-3 whereas, non-private training required learning rate to be set to 1e-4 to get acceptable training and test accuracy. This data allows us to compare the impact of using ImageNet as the pretrained weights and also determine if they are suitable for competing with a non-private baseline.

1) Pediatric Pneumonia Chest X-ray: For experiments on the Pediatric Pneumonia dataset, we found that the optimal performance could be obtained with batch size (logical batch size) the same as mini-batch size (physical batch size). This might suggest that gradient accumulation during training did not have a positive impact on improving the accuracy and performance of the model. Multiple experiments were ran with batch sizes varying from 75, 100, 175, and 250 and the number of epochs varying from 5 and 10. It was observed that for the same privacy budget, a larger value of mini-batch size required more epochs trained to get similar accuracy.

From various experiments on private training, the best test accuracy for models trained with 0.5, 1 and 2 privacy budgets was 82.53%, 84.14% and 83.81% respectively. It was observed that looser privacy budgets had a positive impact on the accuracy. Additionally, we observe that the Epsilon of 1 gave better accuracy than an Epsilon of 2 in this case, suggesting that the extra noise may have been preventing overfitting.

Similarly, experiments done on private training but without a pretrained model resulted in a significant drop in test accuracy. As illustrated in Figure 12, the test accuracy for privacy budgets 0.5, 1 and 2 was 62.34% for all instances. This suggests that without the pretrained weights, the models didn't converge and test results suggest random guesses rather than generalization on the training data. The large difference in accuracy for models with and without pretrained weights demonstrates that transfer learning provides a significant boost in performance and accuracy while training models with differential privacy.

The model achieved a test accuracy of 93.43% during training without incorporating differential privacy. This established a baseline for comparison with the outcomes of private training, highlighting that differential privacy training, when combined with transfer learning and mixed-ghost clipping, yields differentially private models that closely resemble their non-private counterparts.

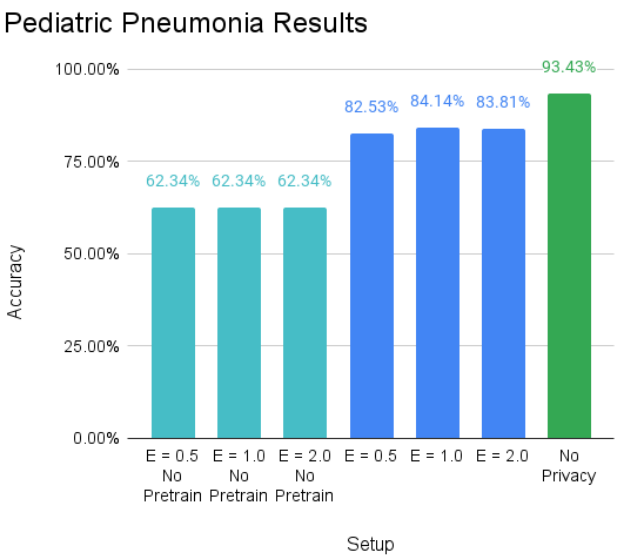

Fig. 12: Accuracy results for Pediatric Pneumonia Dataset

2) DermNet: With the DermNet dataset, we found that the optimal performance could be obtained when batch size (logical batch size) varied from 100, 500 and 1000 while mini-batch size (physical batch size) was kept constant at 100 for all instances. This suggests gradient accumulation during training had a positive impact on improving the accuracy and performance of the model. However, varying the mini-batch size (physical batch size) to larger amounts didn't help with the performance of the model. Multiple experiments were ran with batch size varying from 100, 500, and 1000 and the number of epochs varying from 5 and 10.

From various experiments on private training, the best test

accuracy for models trained with 0.5, 1 and 2 privacy budgets was 21.014%, 27.186% and 32.059% respectively. It was observed that looser privacy budgets had a significant positive impact on the accuracy of models for this dataset.

Similarly, experiments done on private training but without a pretrained model resulted in a significant drop in test accuracy. As illustrated in 13 the test accuracy for privacy budgets 0.5, 1 and 2 were 11.169%, 11.719% and 12.969% respectively. This suggests that without the pretrained weights, the models did not converge well and resulted in worse training and test accuracy. The large difference in accuracy for models with and without pretrained weights demonstrates that transfer learning provides a significant boost in performance and accuracy while training models with differential privacy.

During non-private training, the model attained a test accuracy of 67.291% without integrating differential privacy. This set a benchmark for assessing the results of private training. It emphasized that the inclusion of differential privacy training, along with transfer learning and mixed-ghost clipping, does not necessarily produce differentially private models that closely mirror their non-private equivalents and this behaviour should not be generally expected for all image datasets.

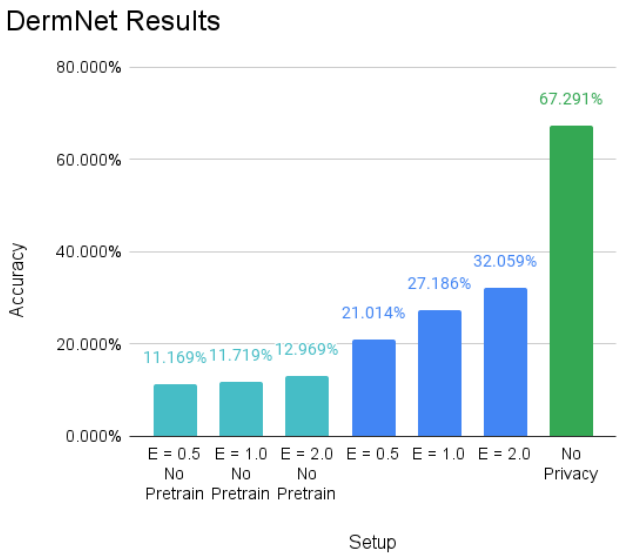

Fig. 13: Accuracy results for DermNet Dataset

3) HAM10000: For experiments with private training on the HAM10000 dataset, we found that the optimal performance could be obtained with a batch size (logical batch size) much larger than the mini-batch size (physical batch size). This might be because gradient accumulation before updating the weights while backpropagating during training helped improve the training and test accuracy. Multiple experiments with batch sizes of 800 and 1600, mini-batch sizes of 10 and 20, and numbers of epochs of 5 and 10 were carried out. It was observed that for the same privacy budget, a larger value of mini-batch size required more epochs trained to get similar accuracy.

From various experiments on private training, the best test accuracy for models trained with 0.5, 1 and 2 privacy budgets was 64.33%, 69.67% and 70.62% respectively. As expected, the test accuracy decreased with stricter privacy budgets. These values were obtained for a batch size of 1600 trained for 10 epochs.

Similarly, experiments done on private training but without a pretrained model resulted in a significant drop in test accuracy. As illustrated in 14 the test accuracy for privacy budgets 0.5, 1 and 2 was 29.03%, 30.62% and 32.04%. The large difference in accuracy for models with and without pretrained weights demonstrates that transfer learning provides a significant boost in performance and accuracy while training models with differential privacy.

Finally, a test accuracy of 78.75% was obtained while training the model without differential privacy. This provided a baseline to compare the results of private training with non-private and shows that DP training with the combination of transfer learning and mixed-ghost clipping provides differentially private models that are closer to their non-private counterparts.

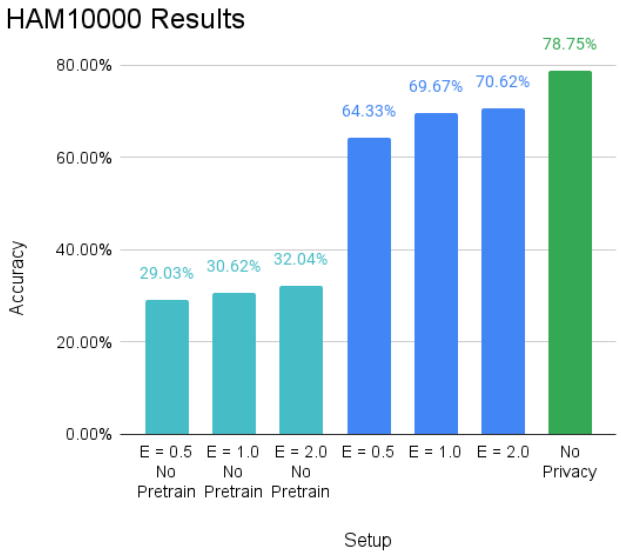

Fig. 14: Accuracy results for HAM1000 Dataset

D. Comparison

1) Pediatric Pneumonia Chest X-ray: For the Pediatric Pneumonia Chest X-ray dataset, other research has trained a VGG11 model with batch normalization using DP for the same binary classification task [30]. Although not mentioned in their paper, we were able to determine from their source code that their models were pretrained on a smaller subset of ImageNet known as ImageNet1K [27]. In their strict privacy case of E = 0.64 using RDP accounting, they achieved a receiver-operator characteristic area-under-the-curve (ROC-AUC) of 0.848. In our experiments using a larger pretrained dataset with the BEiT model, we were able to achieve a ROC-AUC of 0.876 (+.028) with a stricter privacy budget of E = 0.5 using RDP accounting.

We obtained our ROC-AUC results using the *roc_auc_score* and *roc_curve* functions from scikit-learn [33].

Additionally, we found more research that tested this dataset using a novel algorithm that uses semi-private learning called PILLAR [22]. In their experiments, they treat 10% of the data set as public and use it to compute principle components. Also, they use ImageNet1K as their pretraining dataset in each of their models. In their strict case of E = 0.1 using RDP accounting, they achieved an accuracy of 83% compared to our 82.53% with E = 0.5 using RDP accounting. Although our accuracy and privacy are not as optimal, we did not have to treat any of the images as public. We are curious to see if membership inference attacks are possible against the 10% of data that was public future work.

Lastly, the original researchers that published this dataset were able to achieve an accuracy of 92.8% using a convolutional neural network pretrained on ImageNet1K for the binary classification task. In our non-private baseline we achieved an accuracy of 93.43% (+0.63%) with our BEiT pretrained on ImageNet22K.

2) DermNet: The only research that we were able to find that applied DP to DermNet using a single neural network was PILLAR [22]. In their strict case of E = 0.1 using RDP accounting, they achieved an accuracy of 19% compared to our 21.014% with E = 0.5 using RDP accounting. We were able to achieve slightly better accuracy with a looser privacy budget. Although our privacy budget is looser, we do not treat any samples as public during training.

3) HAM10000: [34] has results that are state of the art for non-private training for skin lesion classification and uses capsule networks for dermatoscopic image classification. FixCaps employs a high-performance large-kernel in the lower convolution layer to mitigate the spatial information losses resulting from convolution and pooling. Group convolution is implemented in the capsule layer to prevent underfitting in the model. In comparison to various existing methods, the network enhances accuracy while significantly decreasing computational overhead. Experimental findings indicate that FixCaps outperforms IRv2-SA in skin cancer diagnosis, achieving an accuracy of 96.49% on the HAM10000 dataset. In comparison, our non-private experiments resulted in the best accuracy of only 78.75%. It would be interesting to implement private training on FixCaps to gather data on how it would perform with differential privacy and see if we could further improve test accuracy for private training. This would require developing implementations of the privacy engine that can support layers used in FixCaps.

However, there are not any baselines in the current literature to compare our DP training results with. Hence, further experiments can be done on recent works that have very high accuracy without differential privacy and develop implementations with DP training to observe their utility vs privacy tradeoff with differential privacy incorporated.

V. Future Enhancements

Following up on this work, we first think it would be useful to run additional experiments with other medical and non-medical datasets. We think it is important to continue experimenting to see if out of domain transfer learning is a valid method for training with differential privacy. Also, we think it would be interesting to test datasets that have some overlapping classes with the pretraining dataset.

Next, we think that it is important to run attacks against our models such as the membership inference and model inversion attacks. We think it would be useful to demonstrate the relationship between various privacy budgets and success on the attackers side. Also, experiments with much tighter privacy budgets than our Epsilon of 0.5 case should be conducted for these tests. We also hypothesize that different datasets will perform differently in these attacks despite similar privacy budgets.

Finally, we hypothesize that using data augmentations to artificially increase the sample size, and then the batch size, will allow for more accurate training on our smaller datasets. For example, in [9] the researchers used a batch size of 4,096 for both of their chest X-ray datasets which consisted of 224 thousand and 337 thousand samples. They note that the use of much larger batch sizes can significantly improve accuracy in private training. Conversely, our chest X-ray dataset only had 5,856 samples, which severely limited how far we could increase the batch size. We suggest artificially increasing sample sizes using similar methods that we did in HAM10000 to see if this is a suitable method at improving accuracy.

VI. Conclusion

In this work, we demonstrated that the use of pretrained models trained on large publicly available datasets can increase accuracy significantly when performing transfer learning with differentially private learning. We showed that this is true for our datasets that are out of distribution with the pretraining dataset and have no overlapping classes. In all of our experiments, we see that using randomly initialized weights for our private learning gives very poor accuracy in all cases, and the use of the pretrained weights always provides a significant advantage.

Additionally, in the Pediatric Pneumonia Chest X-ray dataset we demonstrated that using ImageNet22K instead of ImageNet1K improved accuracy substantially. This shows that the size of the pretraining dataset can have an impact on how well differentially private training will perform.

Despite the pretrained models consistently providing an advantage, we see in the DermNet experiment that the advantage may not always be enough. The accuracy obtained in each of the three private trials was less than half of the accuracy obtained in the non-private trial. This result demonstrates that large publicly available datasets for pretraining may not be a general solution for differentially private training of all image types.

Our experiments further show the complexity of achieving a universal solution that provides an ideal balance between privacy and utility. To attain DP models with optimal privacy guarantees suitable for real-world applications, additional in-depth experiments that explore a range of hyperparameters and pretraining datasets are necessary across different image classification tasks.

REFERENCES

[1] N. Ponomareva, H. Hazimeh, A. Kurakin, Z. Xu, C. Denison, H. B. McMahan, S. Vassilvitskii, S. Chien, and A. G. Thakurta, "How to DP-fy ML: A Practical Guide to Machine Learning with Differential Privacy," *Journal of Artificial Intelligence Research*, vol. 77, pp. 1113–1201, Jul. 2023. [Online]. Available: https://www.jair.org/index.php/jair/article/view/14649

[2] F. Mirshghallah, M. Taram, P. Vepakomma, A. Singh, R. Raskar, and H. Esmaeilzadeh, "Privacy in Deep Learning: A Survey," *ArXiv*, 2020.

[3] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep Learning with Differential Privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 308–318. [Online]. Available: https://dl.acm.org/doi/10.1145/2976749.2978318

[4] A. E. Ouadrhiri and A. Abdelhadi, "Differential Privacy for Deep and Federated Learning: A Survey," *IEEE Access*, vol. 10, pp. 22 359–22 380, 2022, conference Name: IEEE Access.

[5] A. Ganesh, M. Haghifam, M. Nasr, S. Oh, T. Steinke, O. Thakkar, A. G. Thakurta, and L. Wang, "Why Is Public Pretraining Necessary for Private Model Training?" in *Proceedings of the 40th International Conference on Machine Learning*. PMLR, Jul. 2023, pp. 10 611–10 627, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v202/ganesh23a.html

[6] H. Mehta, W. Krichene, A. Thakurta, A. Kurakin, and A. Cutkosky, "Differentially Private Image Classification from Features," Nov. 2022, arXiv:2211.13403 [cs]. [Online]. Available: http://arxiv.org/abs/2211.13403

[7] Z. Bu, J. Mao, and S. Xu, "Scalable and Efficient Training of Large Convolutional Neural Networks with Differential Privacy," Nov. 2022, arXiv:2205.10683 [cs]. [Online]. Available: http://arxiv.org/abs/2205.10683

[8] S. De, L. Berrada, J. Hayes, S. L. Smith, and B. Balle, "Unlocking High-Accuracy Differentially Private Image Classification through Scale," Jun. 2022, arXiv:2204.13650 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2204.13650

[9] L. Berrada, S. De, J. H. Shen, J. Hayes, R. Stanforth, D. Stutz, P. Kohli, S. L. Smith, and B. Balle, "Unlocking Accuracy and Fairness in Differentially Private Image Classification," Aug. 2023, arXiv:2308.10888 [cs]. [Online]. Available: http://arxiv.org/abs/2308.10888

[10] C. Dwork, "Differential Privacy," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Berlin, Heidelberg: Springer, 2006, pp. 1–12.

[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating Noise to Sensitivity in Private Data Analysis," *Journal of Privacy and Confidentiality*, vol. 7, no. 3, pp. 17–51, 2016, number: 3. [Online]. Available: https://journalprivacyconfidentiality.org/index.php/jpc/article/view/405

[12] J. Dong, A. Roth, and W. J. Su, "Gaussian Differential Privacy," May 2019, arXiv:1905.02383 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1905.02383

[13] H. Mehta, A. Thakurta, A. Kurakin, and A. Cutkosky, "Large Scale Transfer Learning for Differentially Private Image Classification," May 2022, arXiv:2205.02973 [cs]. [Online]. Available: http://arxiv.org/abs/2205.02973

[14] A. Kurakin, S. Song, S. Chien, R. Geambasu, A. Terzis, and A. Thakurta, "Toward Training at ImageNet Scale with Differential Privacy," Feb. 2022, arXiv:2201.12328 [cs]. [Online]. Available: http://arxiv.org/abs/2201.12328

[15] X. Li, F. Tramèr, P. Liang, and T. Hashimoto, "Large Language Models Can Be Strong Differentially Private Learners," Nov. 2022, arXiv:2110.05679 [cs]. [Online]. Available: http://arxiv.org/abs/2110.05679

[16] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[17] R. Shokri and V. Shmatikov, "Privacy-Preserving Deep Learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, Oct. 2015, pp. 1310–1321. [Online]. Available: https://dl.acm.org/doi/10.1145/2810103.2813687

[18] M. Chase, R. Gilad-Bachrach, K. Laine, K. Lauter, and P. Rindal, "Private Collaborative Neural Network Learning," 2017, publication info: Preprint. MINOR revision. [Online]. Available: https://eprint.iacr.org/2017/762

[19] N. Papernot, M. Abadi, Erlingsson, I. Goodfellow, and K. Talwar, "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data," Mar. 2017, arXiv:1610.05755 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1610.05755

[20] N. Phan, X. Wu, H. Hu, and D. Dou, "Adaptive Laplace Mechanism: Differential Privacy Preservation in Deep Learning," in *2017 IEEE International Conference on Data Mining (ICDM)*, Nov. 2017, pp. 385–394, iSSN: 2374-8486.

[21] A. Blanco-Justicia, D. Sánchez, J. Domingo-Ferrer, and K. Muralidhar, "A Critical Review on the Use (and Misuse) of Differential Privacy in Machine Learning," *ACM Computing Surveys*, vol. 55, no. 8, pp. 160:1–160:16, Dec. 2022. [Online]. Available: https://dl.acm.org/doi/10.1145/3547139

[22] F. Pinto, Y. Hu, F. Yang, and A. Sanyal, "PILLAR: How to make semi-private learning more effective," Jun. 2023, arXiv:2306.03962 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2306.03962

[23] A. Yousefpour, I. Shilov, A. Sablayrolles, D. Testuggine, K. Prasad, M. Malek, J. Nguyen, S. Ghosh, A. Bharadwaj, J. Zhao, G. Cormode, and I. Mironov, "Opacus: User-friendly differential privacy library in PyTorch," version: 4. [Online]. Available: http://arxiv.org/abs/2109.12298

[24] I. Mironov, K. Talwar, and L. Zhang, "R\'enyi differential privacy of the sampled gaussian mechanism." [Online]. Available: http://arxiv.org/abs/1908.10530

[25] H. Bao, L. Dong, S. Piao, and F. Wei, "BEiT: BERT pre-training of image transformers." [Online]. Available: http://arxiv.org/abs/2106.08254

[26] "huggingface/pytorch-image-models," original-date: 2019-02-02T05:51:12Z. [Online]. Available: https://github.com/huggingface/pytorch-image-models

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255, iSSN: 1063-6919. [Online]. Available: https://ieeexplore.ieee.org/document/5206848

[28] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," in *Proceedings of the 18th ACM international conference on Multimedia*, ser. MM '10. Association for Computing Machinery, pp. 1485–1488. [Online]. Available: https://dl.acm.org/doi/10.1145/1873951.1874254

[29] D. S. Kermany, M. Goldbaum, W. Cai, C. C. S. Valentim, H. Liang, S. L. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, J. Dong, M. K. Prasadha, J. Pei, M. Y. L. Ting, J. Zhu, C. Li, S. Hewett, J. Dong, I. Ziyar, A. Shi, R. Zhang, L. Zheng, R. Hou, W. Shi, X. Fu, Y. Duan, V. A. N. Huu, C. Wen, E. D. Zhang, C. L. Zhang, O. Li, X. Wang, M. A. Singer, X. Sun, J. Xu, A. Tafreshi, M. A. Lewis, H. Xia, and K. Zhang, "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning," *Cell*, vol. 172, no. 5, pp. 1122–1131.e9, Feb. 2018, publisher: Elsevier. [Online]. Available: https://www.cell.com/cell/abstract/S0092-8674(18)30154-5

[30] A. Ziller, D. Usynin, R. Braren, M. Makowski, D. Rueckert, and G. Kaissis, "Medical imaging deep learning with differential privacy," vol. 11, no. 1, p. 13524, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-021-93030-0

[31] Dermnet. [Online]. Available: https://www.kaggle.com/datasets/shubhamgoel27/dermnet

[32] P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Scientific Data*, vol. 5, no. 1, p. 180161, Aug. 2018, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/sdata2018161

[33] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[34] Z. Lan, S. Cai, X. He, and X. Wen, "Fixcaps: An improved capsules network for diagnosis of skin cancer," *IEEE Access*, vol. 10, pp. 76 261–76 267, 2022.

VII. Appendix

A. Dataset Experiment Results

Here we provide tables containing the results of all our experiments for different values of hyperparameters such as batch size, epochs and epsilon and the corresponding test accuracy that was obtained.

TABLE I: Pediatric Pneumonia Results

Batch Size	Epochs	Epsilon	Accuracy
250	5	1	84.135
75	3	1	83.814
100	3	2	83.814
175	5	2	83.814
250	3	2	83.494
250	10	2	83.494
75	3	2	83.333
75	5	2	83.333
250	5	2	83.333
175	3	1	83.013
100	5	2	82.853
75	5	0.5	82.532
75	5	1	82.532
250	3	1	82.532
75	10	2	82.532
100	5	0.5	82.372
75	10	1	82.372
250	10	1	82.212
100	3	1	82.051
100	5	1	82.051
175	10	1	81.731
175	3	2	81.731
100	10	1	81.571
100	10	2	81.41
175	5	1	80.449
175	10	2	80.449
75	10	0.5	79.968
175	5	0.5	79.327
175	10	0.5	79.167
250	10	0.5	77.564
75	3	0.5	76.923
250	5	0.5	75
100	3	0.5	73.878
100	10	0.5	70.513
175	3	0.5	62.66
250	3	0.5	62.66

TABLE II: DermNet Results

Batch Size	Epochs	Epsilon	Accuracy
1000	10	2	32.059
1000	5	2	30.785
500	5	2	30.36
500	10	2	29.41
500	3	2	28.711
1000	3	2	28.086
1000	5	1	27.186
500	3	1	26.637
1000	10	1	26.212
500	5	1	25.962
1000	3	1	25.562
500	10	1	24.313
100	3	2	22.514
1000	3	0.5	21.014
500	3	0.5	18.691
500	5	0.5	18.516
1000	5	0.5	18.391
1000	10	0.5	18.291
100	5	2	18.266
100	3	1	17.091
100	10	2	14.718
100	5	1	14.343
500	10	0.5	13.968
100	3	0.5	13.043
100	10	1	12.569
100	10	0.5	12.419
100	5	0.5	10.595

TABLE III: HAM10000 Results

Batch Size	Epsilon	Epochs	Mini Batch Size	Test Accuracy
1600	2	10	20	70.537
1600	2	10	10	69.015
1600	2	5	20	69.600
1600	2	5	10	70.620
1600	1	10	20	66.237
1600	1	10	10	70.688
1600	1	5	20	69.667
1600	1	5	10	70.437
1600	0.5	10	20	63.962
1600	0.5	10	10	57.872
1600	0.5	5	20	62.757
1600	0.5	5	10	64.330
800	2	10	20	68.379
800	2	10	10	67.643
800	2	5	20	70.018
800	2	5	10	67.810
800	1	10	20	62.138
800	1	10	10	59.796
800	1	5	20	66.923
800	1	5	10	66.589
800	0.5	10	20	58.541
800	0.5	10	10	57.337
800	0.5	5	20	64.414
800	0.5	5	10	59.077

B. BEiT Structure

Below is an example of what the BEiT model looks like when printing it inside the console. The *out_features* is 2 in this example for the Pediatric Pneumonia Chest X-ray dataset.

```
Beit(
  (patch_embed): PatchEmbed(
    (proj): Conv2d(3, 768, kernel_size=(16, 16), stride=(16, 16))
    (norm): Identity()
  )
  (pos_drop): Dropout(p=0.0, inplace=False)
  (blocks): ModuleList(
    (0-11): 12 x Block(
      (norm1): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
      (attn): Attention(
        (qkv): Linear(in_features=768, out_features=2304, bias=False)
        (attn_drop): Dropout(p=0.0, inplace=False)
        (proj): Linear(in_features=768, out_features=768, bias=True)
        (proj_drop): Dropout(p=0.0, inplace=False)
      )
      (drop_path1): Identity()
      (norm2): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
      (mlp): Mlp(
        (fc1): Linear(in_features=768, out_features=3072, bias=True)
        (act): GELU(approximate='none')
        (drop1): Dropout(p=0.0, inplace=False)
        (norm): Identity()
        (fc2): Linear(in_features=3072, out_features=768, bias=True)
        (drop2): Dropout(p=0.0, inplace=False)
      )
      (drop_path2): Identity()
    )
  )
  (norm): Identity()
  (fc_norm): LayerNorm((768,), eps=1e-06, elementwise_affine=True)
  (head_drop): Dropout(p=0.0, inplace=False)
  (head): Linear(in_features=768, out_features=2, bias=True)
)
```

C. Augmentation for DermNet

```
data_transforms = {
    'train': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.RandomHorizontalFlip(),
        transforms.RandomAffine(degrees=(-10,
10), translate=(0, 0.1),

scale=(0.9, 1), shear=(0, 5, 0, 5)),
        transforms.ToTensor(),
        transforms.Normalize([0.485,
0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'test': transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor(),
        transforms.Normalize([0.485,
0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```

D. Augmentation for Pediatric Pneumonia

```
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485,
0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'test': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485,
0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
}
```

E. Augmentation for HAM10000

```
data_transform = transforms.Compose([
    transforms.Resize((224,224)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomVerticalFlip(),
    transforms.RandomRotation(20),
    transforms.ColorJitter(brightness=0.1,
contrast=0.1, hue=0.1),
    transforms.ToTensor()
    ])
```