DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Robotics, Cognition and Intelligence

Differentially Private Training of Residual Networks with Scale Normalization

Helena Klause

DEPARTMENT OF INFORMATICS

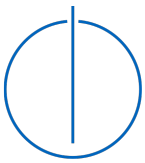
TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Robotics, Cognition and Intelligence

Differentially Private Training of Residual Networks with Scale Normalization

Differentiell Privates Training von Residualen Netzwerken mit Scale Normalisierung

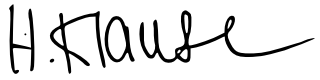
Author:	Helena Klause
Supervisor:	Prof. Dr. Daniel Rückert
Advisor:	Dr. Georgios Kaissis, Dr. Kerstin Hammernik, Alexander Ziller
Submission Date:	15.06.2022



I confirm that this master's thesis in Robotics, Cognition and Intelligence is my own work and I have documented all sources and material used.

Munich, 15.06.2022

Helena Klause



Acknowledgments

I am deeply grateful to my supervisors George, Alex, and Kerstin, without whom this work would have not been possible. They always gave me enough space to express my own ideas, led me to interesting research questions, and invested a lot of time to support me during my work. I offer special appreciation to George, for his trust in me which gave me courage and motivated me tremendously to do good work. His focus and precision, as well as his way of giving constructive feedback, continually inspired me. Big thanks to Alex for keeping our meetings light and fun, while always making sure things are on track and realistically planned. I also appreciate his commitment very much to provide me with a GPU for running my experiments. Kerstin's specialized knowledge has often added a whole different point of view to our weekly discussions that has been very helpful. Thanks also to her for sharing GPU resources with me.

I would also like to express a great thank you to Daniel for having me at his lab and making this thesis possible. I enjoyed the atmosphere at the lab a lot and felt very welcomed and supported by everyone.

Lastly I want to thank my partner Michael for helping me organize my thoughts, proof-reading this thesis, and his support throughout the last months.

Abstract

Training neural networks with sensitive data requires privacy-preserving methods to ensure that no private information is revealed to the public. Differential privacy mathematically guarantees an upper bound on the privacy loss resulting from releasing the results of computations on sensitive data, and is the gold standard for privacy-preserving machine learning. However, there is a privacy-accuracy trade-off when training in a differentially private way. A special optimization technique called Differential Private Stochastic Gradient Descent ensures that the privacy loss stays below a targeted privacy budget when training models. Our work improves on the trade-off curve by introducing a novel architectural adaptation termed Scale Normalization and additionally gives useful guidelines for training differentially private models. Since the widely used Batch Normalization layer can not be used with Differentially Private Stochastic Gradient Descent, we additionally investigate the best choice of a replacement layer, and compare Layer, Instance and Group Normalization with various group sizes. In our work we show that our techniques can improve the current state-of-the-art for CIFAR-10 by achieving 82.5% validation accuracy for a privacy budget of $\varepsilon = 8$.

Contents

1. Introduction

1.1. Motivation

Machine learning (ML) is particularly successful at image analysis when trained with large high-quality datasets. Many of these datasets are crowdsourced and often include sensitive information, depending on the domain. However, research has shown that naive machine learning algorithms do not protect the privacy of their input data. In fact, training data is partially reconstructable given the trained model (Carlini et al., 2021; 2022; Choquette-Choo et al., 2021; Balle et al., 2022; Liu et al., 2021). Neural networks are able to memorize large parts of the training data, especially if the model is over-parameterized. Feldman (2019) even claims that memorization is necessary for good generalization. This raises an important question of how to train effective neural networks while preserving data privacy. Especially in the medical context, this is a major concern that blocks many potential ML applications. Due to the highly sensitive nature of patient data and the valid privacy restrictions around it (Kaissis et al., 2020), publicly available datasets are rare. This makes it difficult to gather the large volume of data needed for high-accuracy results. Privacy-preserving ML addresses this problem by enabling learning algorithms to protect sensitive information while still capturing the underlying patterns.

Differential privacy (Dwork, 2006) – DP for short – is the gold standard technique to prevent the leakage of individual training samples from a model (Kaissis et al., 2020; De et al., 2022). DP is only one amongst several methods used in privacy-preserving ML; a full summary of privacy-enhancing techniques can be found in Kaissis et al. (2021). Note that anonymization does not count as adequate privacy protection, since samples remain re-identifiable with the help of other publicly available data, even if unique features like name and date of birth are removed (Narayanan and Shmatikov, 2008; Schwarz et al., 2019).

Differential privacy provides privacy protection guarantees but also yields a significant reduction in the model's accuracy. This trade-off in privacy-utility is an upshot of the fundamental principle behind differential privacy: the addition of noise during the training process, which leads to a higher loss and lower accuracy. Due to DP, the training dynamics change significantly and adaptations on multiple levels of the training are required to make learning possible. In the last few years, DP research has

mainly focused on simpler tasks and shallow architectures that tend to perform better with private training compared to large ones. That research has resulted in a reduction in the performance gap between naive and DP training for such tasks (Papernot et al., 2021; Dörmann et al., 2021). Only very recent works (Kurakin et al., 2022; De et al., 2022) attempt to tackle deeper networks (e.g. large residual networks) for more challenging tasks, and it is clear that there is still a long way to go to fully unlock the potential of differentially private training and enable broad application in the industry.

1.2. Research question

This work aims to improve the trade-off in privacy-utility for neural networks trained on datasets of moderate to high complexity (CIFAR-10, ImageNette (Howard, 2019) and Tiny ImageNet (Le and Yang, 2015)). Currently, training of private datasets consists of a challenging search for suitable hyperparameters which is very time- and resource-intensive. We are addressing this by finding widely applicable architectural adaptations combined with hyperparameter guidelines. Our main focus is on residual networks that have not yet been studied well under DP training, but represent some of the most successful neural architectures for non-DP training. Previously studied architectures for DP training such as plain convolutional networks with just a few layers train well, but are very limited in their capacity (Papernot et al., 2021; Dörmann et al., 2021). In our work, we strike a balance between increased model capacity to be able to train more challenging tasks, and keeping the network compact enough to avoid over-parameterization, which incurs a peformance decrease due to increased noise addition requirements. In the absence of techniques such as the ones presented in our work, networks with large numbers of parameters trained under DP exhibit such low accuracy that they are unsuitable for use in any task, underscoring the necessity to reduce network parameters for efficient training (Kurakin et al., 2022; De et al., 2022).

1.3. Contributions

Our main contribution is a novel architecture adaptation called Scale Normalization (ScaleNorm) that improves the model accuracy of DP training. ScaleNorm addresses a phenomenon we termed *scale mixing* – the difference in scale between activations along the two branches inside a residual block – that negatively influences the convergence. The concept of ScaleNorm utilizes additional normalization layers to ensure equally scaled activations throughout the network. We apply this concept to different residual networks (ResNets) for image classification, as well as UNets and LinkNets for image

segmentation. We also demonstrate that our architectural adaptation benefits the model's performance for transfer learning.

The standard optimizer for DP training of neural networks is differentially private stochastic gradient descent (DP-SGD), introduced by Abadi et al. (2016), which is not compatible with Batch Normalization (BN). As BN is a main component of residual networks, we analyze the best replacement normalization layers and condense useful guidelines for DP-training. Additionally, we show the benefits of using the Mish activation function (Misra, 2020) for DP training of residual architectures over the rectified linear unit (ReLU), which is traditionally used in ResNets. Along with the research into architecture adaptation, we also propose general guidelines for hyperparameters that benefit DP learning, in particular the batch size and number of epochs. We show how the choice of NAdam (Dozat, 2016) as optimizer significantly decreases the learning rate search due to its robustness.

2. Fundamentals of differential privacy

2.1. Definition

Differential privacy (DP) is a concept introduced in 2006 by Cynthia Dwork that quantifies the privacy loss of an algorithm operating on a dataset. The goal was to find a firm upper bound for the quantity of information revealed (*leaked*) about an individual data point when trying to analyze characteristics of the underlying population. Differential privacy is agnostic to the type of data. Consider an image with a person's face in the foreground and a landscape with trees in the background. A DP algorithm would protect the background and foreground information equally, even though one could argue that trees do not have privacy that needs to be protected in the first place. This seemingly unimportant feature of DP – context freedom – is actually one of its biggest upsides, as it guarantees privacy for all possible use-cases even if additional attributes of the data are only specified as private afterwards. This property, combined with its unambiguous mathematical definition, makes DP the gold standard tool for privacy-preserving data analysis.

Mathematical definition Consider an algorithm that operates on a dataset to produce some output; for example, an algorithm that computes the average population age from a dataset containing personal information. Non-determinism is essential for DP, so this algorithm is randomized by adding noise. The actual average age is therefore produced with a probability < 1, and the output of such a randomized algorithm $\mathcal{M}$, when applied to a dataset d, has a probability distribution $p_{\mathcal{M}}(d)$. Differential privacy quantifies the difference in that probability distribution when $\mathcal{M}$ is applied to two adjacent datasets d_1 and d_2. In this context, *adjacent* means that the datasets differ in exactly one element, i.e. in the data of one individual. $\varepsilon > 0$ and $\delta \in [0, 1]$ are the measures that characterize this difference and give rise to the term (ε, δ)-differential privacy (Dwork and Roth, 2014), in which ε is also referred to as the *privacy budget*. More precisely, ε describes an upper bound on the information leaked about individual elements in the dataset and δ the – usually minuscule, e.g. 10^{-5} – probability that this bound is violated. A high value of δ therefore weakens the privacy guarantee in the sense that it allows for the occurrence of scenarios in which the guaranteed bound is exceeded, and consequently should be kept small. In the case that $\delta = 0$

(ε, δ)-differential privacy is equal to the previously defined version called ε-differential privacy (Dwork, 2006). The preserved privacy increases for decreasing δ and ε.

In other words, DP makes sure that the algorithm's outputs are nearly identical when it is applied to datasets differing only in the data of a single individual. For strongly differentially private algorithms (very low δ and ε), the output probability distribution remains virtually the same if one element is removed. A potential attacker can therefore infer very little information about any element of the dataset.

Definition (Differential Privacy (Dwork and Roth, 2014))**.** A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is (ε, δ)-differentially private if for all $S \subseteq \text{Range}(\mathcal{M})$ and for all $d_1, d_2 \in \mathbb{N}^{|\mathcal{X}|}$ such that $||d_1 - d_2||_1 \leq 1$:

$$\mathbb{P}[\mathcal{M}(d_1) \in S] \leq e^{\varepsilon} \cdot \mathbb{P}[\mathcal{M}(d_2) \in S] + \delta,$$

where $\mathbb{P}$ denotes the probability measure. Differential privacy is by definition robust to post-processing (Dwork and Roth, 2014). This means that any operation – irrespective of any available side information – applied to the output of the differentially private algorithm will not decrease the privacy guarantees; they are only violated if information about the original data is included. Another key property of DP is the composition theorem, which defines the composition of two differentially private algorithms as itself differentially private (Dwork, 2006). This allows for the construction of complex systems from simple differentially private building blocks and enables differentially private deep learning.

2.2. Differential privacy in deep learning

2.2.1. Gaussian Mechanism

Differential privacy is commonly achieved by adding noise to the algorithm that is calibrated to its sensitivity. There exist different definitions of sensitivity, which – intuitively– defines the maximum possible difference in the algorithm's output for any input. For a function f and two adjacent datasets d_1 and d_2, the ℓ_2-sensitivity is defined as $\Delta_2(f) = \max ||f(d_1) - f(d_2)||_2$ and the ℓ_1-sensitivity correspondingly with the ℓ_1-norm (Dwork, 2006).

For differentially private deep learning, we often make use of the Gaussian Mechanism for adding the noise and calculating the respective ε and δ values. We sample from the normal (Gaussian) distribution $\mathcal{N}(0, \Delta_2(f)^2 \cdot \sigma^2)$ with zero-mean and standard deviation of $\Delta_2(f) \cdot \sigma$ to the function $f(d)$ (Abadi et al., 2016). This then defines a randomized algorithm constructed from f with the Gaussian Mechanism as:

$$\mathcal{M}(d) \triangleq f(d) + \mathcal{N}(0, \Delta_2(f)^2 \cdot \sigma^2)$$

The algorithm $\mathcal{M}$ is (ε, δ)-differentially private if:

$$\Phi\left(\frac{\Delta_2(f)}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_2(f)}\right) - e^{\varepsilon}\Phi\left(-\frac{\Delta_2(f)}{2\sigma} - \frac{\varepsilon\sigma}{\Delta_2(f)}\right) \leq \delta(\varepsilon),$$

where $\Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-y^2/2} \mathrm{d}y$ (Balle and Wang, 2018). If the Gaussian Mechanism is applied multiple times, a so-called privacy accountant is employed to keep track of the individual steps and accumulate the spent privacy per step into the overall privacy expenditure (Abadi et al., 2016).

2.2.2. Differentially private stochastic gradient descent

Training a neural network can be seen as an algorithm that computes characteristics of the underlying pattern in a dataset, where the input is the dataset itself and the output is the trained network's parameters. To enable differentially private training one could apply the Gaussian Mechanism directly to the parameters. In practice, this presents a challenge. The parameters are computed from many different samples seen multiple times, since training usually takes multiple epochs. Characterizing these dependencies is difficult, and just assuming the worst-case scenario results in high noise addition, which makes training useless (Abadi et al., 2016). A more sophisticated approach involves adding the noise during training within the stochastic gradient descent (SGD). In non-DP training, we calculate the loss function in the forward pass, and the backward pass then consists of calculating the per-batch gradients from the loss and updating the model's parameters. Abadi et al. (2016) developed an adaptation of SGD for private training – differentially private stochastic gradient descent DP-SGD – which makes use of the Gaussian Mechanism before the update step. For N being the number of samples, x_i, L being the group size (batch size) of a group of samples L_t individually sampled at random (batch), and $\mathcal{L}(\theta_t, x_i)$ being the loss term for a single sample x_i, the algorithm is defined as depicted in 1.

Note above, that since we cannot determine the sensitivity of the gradients (or it is too high to be useful), we clip them in the ℓ_2 norm, which results in a sensitivity equal to the clipping value C. In the Gaussian Mechanism, we then multiply σ with C and a noise multiplier I, which enables us to regulate the multiplied noise. The algorithm is defined for only one layer, but can be applied to every layer individually if networks have multiple layers. The accumulation of the spent privacy will be taken care of by a privacy accountant. DP-SGD uses *Renyi differential privacy* (Mironov, 2017), a natural relaxation of (ε, δ)-differential privacy since it is better suited to analyse the composition of many Gaussian mechanisms.

Importantly, it is necessary to calculate the *per-sample gradients* to ensure the clipping of every individual gradient. In non-DP training, the gradients are calculated per-batch,

Algorithm 1 Differentially private SGD (Abadi et al., 2016)

Initialize models parameter θ_0 randomly
for $t \in T$ **do**:
Sample a random group of points L_t with sampling probability L/N
Compute gradients
For each point $x_i \in L_t$, compute individual gradient: $g_t(x_i) \leftarrow \nabla_\theta \mathcal{L}(\theta_t, x_i)$
Clip gradients
Clip each gradient in the ℓ_2 norm: $\bar{g}_t(x_i) \leftarrow \max\left(1, \frac{||g_t(x_i)||_2}{C}\right)$
Add noise
Apply Gaussian Mechanism to gradients aggregated over L_t:
$\tilde{g}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{g}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$
Descent
Update the model's parameters: $\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{g}_t$
end for
Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

and if gradient clipping is applied, also clipped by batch. This difference results in the incompatibility of batch normalization and DP-SGD, since the notion of per-sample gradients can not be combined with computing the mean and variance statistics over a whole batch of activations. If the additional privacy consumption is calculated for the use of batch normalization and DP-SGD is altered respectively, it could be combined with differentially private training (Davody et al., 2021). Other normalization layers like Group (Wu and He, 2018), Instance (Ulyanov et al., 2016) and Layer Normalization (Ba et al., 2016) only depend on the respective sample and compute statistics over different features of that sample and are therefore compatible with DP-SGD.

3. Preliminaries

3.1. Residual networks

Residual networks (ResNets) represent a successful group of neural network architectures that are widely used for complex image classification tasks. The original work was published by He et al. (2016) and was motivated by the degradation of training accuracy for deep feed-forward networks of the time. Works by He and Sun (2014) and Srivastava et al. (2015) show that increasing network depth can significantly harm the performance of such networks. Residual networks address this phenomenon by introducing residual blocks that are stacked sequentially. In the traditional residual architecture, these blocks consist of the convolutional (normal) path V_F and the residual path V_R. Whereas V_F consists of convolutional layers, normalization layers, and activation functions, V_R, also known as a skip connection, just represents the identity function. Both paths are added at the end of the block. This is visualized in Figure 3.1 A. The introduction of the residual path does not increase the number of parameters or computational complexity. He et al. present ResNets of depth 18, 34, 50, 101 and 152 where the largest one shows the lowest test error on ImageNet. For our work, we use ResNet-50 and additionally constructed a shallow ResNet with 9 layers.

Even though ResNets are widely used in a non-DP setting, only very recent work by Kurakin et al. (2022) and De et al. (2022) show their use with differentially private training. Utilizing ResNets for DP is a big step towards a broader scope of application of private deep learning. Deep residual networks, however, tend to perform worse in differentially private training than shallower ones, and finding hyperparameters that make training efficient is a difficult and lengthy process.

3.2. Scale mixing

As Figure 3.1 A illustrates, the convolutional path V_F undergoes two normalization layers, whereas the residual path V_R does not get normalized at all. This creates differently scaled paths that are added at the end of the block. We call this phenomenon *scale mixing*.

To demonstrate the different scales, we analyzed the activations during training,

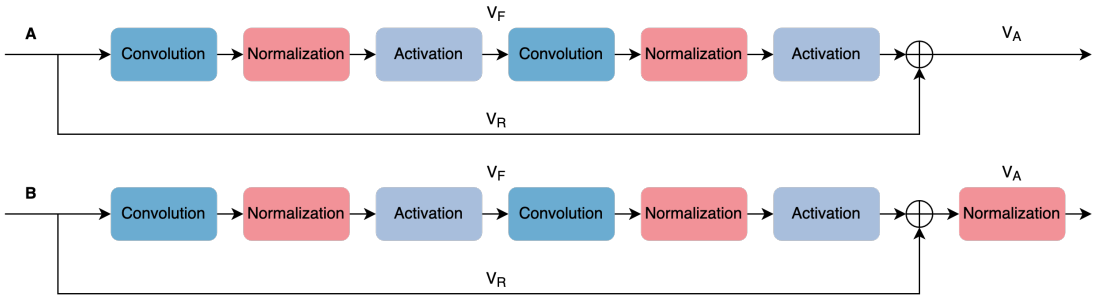


Figure 3.1.: Schematic representation of a residual block (**A**) vs. a residual block with ScaleNorm (**B**). V_R denotes the residual path, V_F the convolutional path, and V_A the activation.

which can be seen in Figure 3.2. After the addition, the activation is skewed with a sample average of 0.72 and a standard deviation of 0.76. Previous research has underscored the importance of symmetric and well-scaled activations for efficient neural network convergence (Glorot and Bengio, 2010; Zhang et al., 2019). We therefore developed a modification of the residual architecture to reduce scale mixing and thereby improve the training dynamics in the setting of differentially private training.

3.3. Scale Normalization

The architectural adaptation termed *Scale Normalization* (ScaleNorm) addresses the phenomenon of scale mixing by an additional normalization layer after the addition operation at the end of each residual block, see Figure 3.1 B. We call residual networks with this modification *ScaleResNets*.

The histograms in Figure 3.2 show the activation of the convolutional path V_F, residual path V_R and after Scale Normalization V_A. The latter is significantly more symmetric compared to the activation without Scale Normalization, and has a sample mean of 0 and standard deviation of 1. The improved distribution explains the performance enhancements that ScaleResNets show. This also results in a faster convergence as visualized in Figure 3.3. The graphs show the curve of the validation accuracy of training a small ResNet with a target epsilon of 10 on the dataset ImageNette and CIFAR-10. For both examples, the red curve, denoting training with ScaleNorm, lies significantly above the blue one (without ScaleNorm). For the same accuracy, training can therefore be stopped early which improves privacy and saves time. Or, in other words, the same privacy budget results in a better validation accuracy.

In an experimental evaluation of the Hessian of the ScaleResNet-9 architecture

compared to the ResNet-9, the trace is significantly lower (6637.3 vs. 9045.7), the condition number slightly lower ($2.1 \cdot 10^{-3}$ vs. $2.3 \cdot 10^{-3}$), and the number of Hessian eigenvalues are lower (115 vs. 119) with a lower maximum value of (2654.2 vs. 3920.4) (Klause et al., 2022). These results indicate an improved loss landscape that further explains the increased performance (Yao et al., 2019).

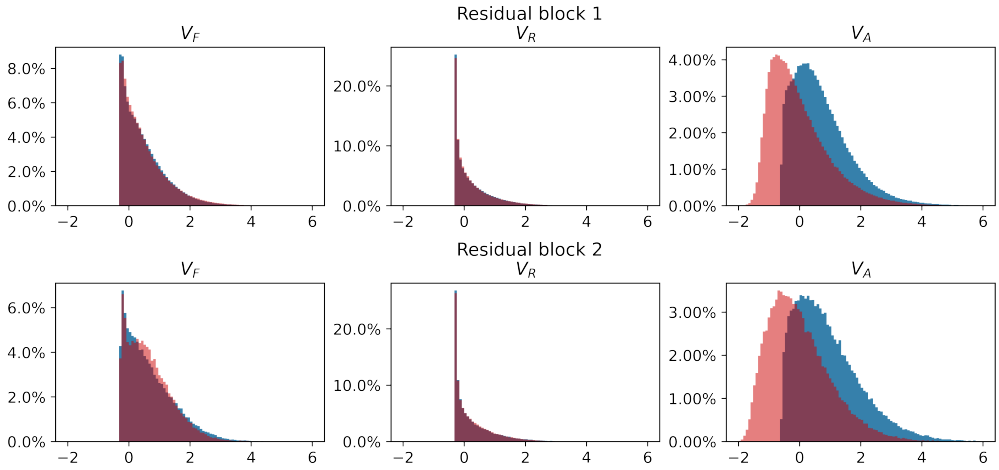


Figure 3.2.: Activation histograms of the first and second residual block of a ResNet-9 (blue) vs. a ScaleResNet-9 (red). V_R (residual path) and V_F (convolutional path) show only minimal difference, whereas V_A (activation) depicts the improved distribution of activations when using ScaleNorm.

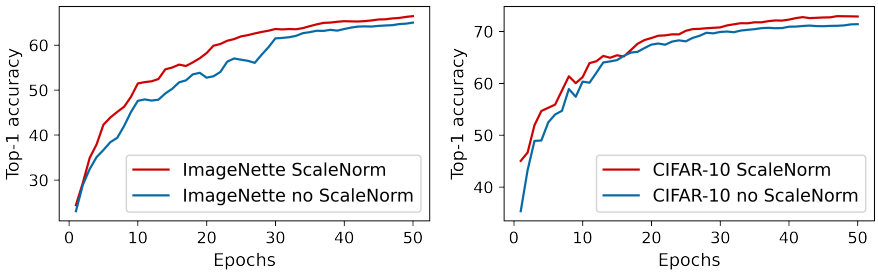


Figure 3.3.: Top-1 validation accuracy (in %) of ResNet-9 trained on ImageNette and CIFAR-10 for $\varepsilon = 10$ with ScaleNorm (red) and without (blue). Enhanced training dynamics with ScaleNorm are visible in faster convergence.

4. Related work

4.1. Transfer learning

Since training difficult tasks from scratch using large neural networks with differential privacy is still intractable, many researchers use transfer learning to overcome these constraints (Abadi et al., 2016; Davody et al., 2021; Luo et al., 2021; Kurakin et al., 2022; De et al., 2022). Publicly available data is used to train a large network without DP, then these pre-trained models are fine-tuned with the actual private dataset for a given privacy guarantee. Luo et al. (2021) argues that minimizing the trainable parameters for fine-tuning is crucial for successful transfer learning with DP. Transfer learning can even leverage the benefits of BN if these layers are frozen for the fine-tuning (Davody et al., 2021). Tramèr and Boneh (2021) study in their work how non-learned feature extractors (ScatterNets; Oyallon et al., 2019) perform with DP and compare these models with transfer learning from public datasets. The recently proposed works by Kurakin et al. (2022) and De et al. (2022) show how transfer learning can be leveraged to train the difficult non-DP benchmark task ImageNet.

4.2. Architectural adaptations

Custom architecture design When DP was first applied to deep learning, researchers used networks that have been shown to train well without DP and simply changed the optimizer from SGD to DP-SGD. The achieved accuracies showed a large privacy-utility trade-off (Papernot et al., 2021). Works by Morsbach et al. (2021) and Papernot et al. (2020) claim that efficient training of private datasets requires modifications to existing architectures or –preferably– entirely custom architectures. If well-performing standard model architectures (e.g. ResNet-18, ResNet-50) are used for differentially private training on a difficult dataset (e.g. ImageNet) without any modifications, the accuracy is typically almost zero for any reasonable privacy budget (Kurakin et al., 2022). Remerscheid et al. (2022) analyze different components of traditional neural network architecture and propose a new model architecture named SmoothNet that is optimized for DP training. The architecture combines their main findings: higher width-depth ratio and dense connections (Huang et al., 2017) are beneficial for private

training. Neural architecture search for differentially private deep learning has been conducted by Cheng et al. (2021) and Singh et al. (2020) to automate the process of designing an architecture that results in the best performance.

Activation function Choosing the right activation function can significantly improve the accuracy of models trained in a differentially private setting (Papernot et al., 2021). Papernot et al. studied the effect of ReLU as activation functions and proposes the hyperbolic tangent (tanh) as a more suitable alternative. Cheng et al. (2021) then show that the scaled exponential linear unit (SELU) (Klambauer et al., 2017) can outperform tanh. In our work, we are using the Mish activation function (Misra, 2020) which exhibits self-regularizing properties and, to the best of our knowledge, has not been used for DP-training so far.

Normalization layers So far, Group Normalization with a group size of 32 has been the default replacement for Batch Normalization (Kurakin et al., 2022; De et al., 2022; Dörmann et al., 2021; Yousefpour et al., 2021). To the best of our knowledge, no systematic study has been published that analyzes different normalization layers with DP-SGD. Davody et al. (2021) showed the importance of normalization layers combined with added noise, but do not analyze Group Normalization. Our work shows the effect of Layer, Instance, and Group Normalization with different group sizes.

4.3. Training adaptations

Batch size Dörmann et al. (2021) showed in their work that the overall noise that a differentially private deep learning model is exposed to consists of the sampling noise introduced by training with mini-batches (resp. random samples of points) and the added noise of the Gaussian Mechanism to ensure privacy. The authors conclude that the training improves by increasing the batch size, due to a reduced effect of the sampling noise. In our work, we therefore treat the batch size as an additional hyperparameter to tune. A performance increase due to large batch sizes can also be seen in Kurakin et al. (2022) and (De et al., 2022).

Hyperparameter search Hyperparameter search significantly improves differentially private training but strictly speaking violates privacy guarantees (Papernot and Steinke, 2022) due to repeated training, which quickly exhausts the privacy budget. Nevertheless, current research usually neglects this fact and performs hyperparameter search without calculating additional privacy (Kurakin et al., 2022; De et al., 2022; Dörmann et al., 2021; Papernot et al., 2021). In our work we similarly train multiple times on the same

dataset to tune hyperparameters. We note that the differential privacy guarantees of the model itself are not affected by this process.

Data augmentation Data augmentation for image classification is a successful tool to decrease memorization in favor of better generalization. De et al. (2022) show that this fact does not hold true for private training. Data augmentation implemented in the traditional way of one augmentation per sample decreases the accuracy of differentially private models (De et al., 2022) due to its regularizing effect. However, the authors show that if multiple augmentations are conducted on each individual sample in the training set (augmentation multiplicity (Fort et al., 2021)), the performance of private training can be improved. Note that the calculation of the per-sample gradients in DP-SGD has to be altered to not violate the privacy guarantees when using augmentation multiplicity.

4.4. Modifications of DP-SGD

The DP-SGD optimizer developed by Abadi et al. (2016) is the standard for differentially private training and implemented in the most widely-used DP-libraries, Opacus (Yousefpour et al., 2021) and Tensorflow Privacy (Google, 2018). Nevertheless many researchers have developed modifications or alternatives techniques to DP-SGD (Zhu et al., 2020; Papernot et al., 2018; Balle and Wang, 2018; Xu et al., 2019; Wang et al., 2020). Our work is implemented with Opacus and uses DP-SGD as the optimizer.

Denoising A specific modification is an additional denoising step applied to the gradient after adding the noise but before updating the weights. Differential privacy is robust to post-processing and therefore denoising the gradient does not violate the privacy guarantees. Works by Nasr et al. (2020) and Balle and Wang (2018) show how denoising can benefit the model accuracy in private training.

5. ScaleResNets for image classification

5.1. Datasets

To evaluate our novel architectural adaptation, we trained our models on three image classification tasks: CIFAR-10 (Krizhevsky et al., 2009), ImageNette (Howard, 2019), and Tiny ImageNet (Le and Yang, 2015). CIFAR-10 belongs to the most widely used datasets in ordinary as well as differentially private deep learning. It consists of 10 classes and 60,000 images of size 32×32, of which 10,000 are validation images (Krizhevsky et al., 2009). ImageNette is a subset of the well-known ImageNet dataset (Deng et al., 2009) that only contains 10 easily classifiable classes and consists of ∼13,000 images of size 160×160 split in a ratio of 70%, 30% into training and validation set. We chose ImageNette since it is significantly more difficult to train than CIFAR-10 while still having the same amount of classes and a high resolution. We picked Tiny ImageNet as the most challenging dataset, with 200 classes and 100,000 training images and 10,000 validation images all of size 64×64. Training a model on Tiny ImageNet is orders of magnitudes faster than on ImageNet while still being a similar challenge (Le and Yang, 2015).

For our experiments on transfer learning, we additionally used CIFAR-100 (Krizhevsky et al., 2009), which has 100 classes each consisting of 500 training and 100 validation images of size 32×32. Despite the similar name, CIFAR-10 and CIFAR-100 are completely disjoint, meaning they do not share the same images nor the same classes.

5.2. Methods

5.2.1. Model architectures

Since differentially private training suffers from using highly over-parameterized models, we developed a ResNet-9 with 2,447,946 parameters whose exact implementation can be found here[1]. We have found this to be a very capable architecture for differentially private training. The reduced parameters compared to traditional ResNet architectures of larger depth result in good accuracy (due to less added noise) and faster

[1] `https://github.com/mrchntia/ScaleNorm/blob/main/image_classification/resnet9.py`

training. As a second architecture we chose the traditional ResNet-50 with 23,528,522 parameters which is roughly 10× more than ResNet-9. For ResNet-50 we used the standard PyTorch implementations (PyTorch, 2021). We constructed modifications of both architectures that contain ScaleNorm and are then respectively called ScaleResNet-[9, 50].

5.2.2. Duration of training

In differentially private training, every additional epoch consumes privacy budget. Consequently, the added noise per step has to be larger for longer training than for shorter if the privacy budget stays the same. Nevertheless, training conditions are more challenging for DP training and make longer training necessary to achieve good generalization. Thus, longer training is preferred over short training up to a reasonable number of epochs considering the trainable parameters and the datasets' difficulty. Table 5.1 demonstrates the performance peak for 50 epochs when training CIFAR-10 on ResNet-9 with an ε value of 7.5.

Table 5.1.: Comparing the model accuracy in % on the CIFAR-10 dataset trained on ResNet-9 for 25, 50 and 90 epochs with a privacy budget of $\varepsilon = 7.5$. All results are reported as median values of three runs.

$\varepsilon = 7.5$				
Model	Dataset	25	50	90
ResNet-9	CIFAR-10	69.55	**71.42**	70.61

5.2.3. Replacement for Batch Normalization

As stated in Chapter 2, Batch Normalization cannot be used when training in a differentially private way without violating privacy guarantees. BN depends on a whole batch of samples and is therefore not compatible with the notion of per-sample gradients. Valid alternatives are e.g. Group Normalization (GN), Instance Normalization (IN), and Layer Normalization (LN), which only depend on individual samples. For all normalization layers, we disabled the collection of running variance and mean wherever applicable. Note that IN can be represented as GN with numbers of groups equal to 1 and LN with the number of groups equal to the number of channels.

Especially for ResNets, BN significantly outperforms other normalization layers, so switching to an alternative normalization layer usually decreases the overall perfor-

mance (Lubana et al., 2021). To mitigate the decrease we analyzed the performance of LN, IN and GN and grid-searched over the number of groups to find the best replacement for batch normalization. Additionally, we ran experiments for varying group sizes throughout the network. To reduce the complexity, we segregated the network into four stages (early, mid, late, and final) and kept the group size equal within a stage. The guiding principle of this compartmentalization is that close stages in the network are likely to benefit from sharing the same group size.

5.2.4. Self-normalising activation function Mish

The choice of the activation function can have a significant influence on the training dynamics of neural networks. Hence, much research has been conducted on optimizing these functions. In the originally presented ResNet architecture, the authors propose to use the ReLU activation function (Zhang et al., 2019). Since Papernot et al. (2021) shows that ReLU is not optimal for DP-training, we investigated the performance of the Mish activation function by Misra. The function has a self-regularizing effect and outperforms well-known alternatives like ReLU, ELU, LeakyReLU and TanH applied to several architectures including ResNets in the non-DP setting (Misra, 2020). Table 5.2 shows the increased accuracy when using Mish over the use of ReLU.

Table 5.2.: Comparing the model accuracy in % on the CIFAR-10 and ImageNette dataset using the ResNet-9 model architectures with ReLU activation function to Mish activation function. All results are reported as median values of three runs.

$\varepsilon = 7.5$		
Model	CIFAR-10	ImageNette
ResNet-9 (ReLU)	69.78	60.41
ResNet-9 (Mish)	**71.42**	**63.77**

The major drawback is the increased training time. The computation of the Mish activation function is almost 4 times slower than ReLU, which results in an overall increase in training time (Misra, 2020). Nevertheless, as the runtime is already drastically increased by the additional computation needed for DP training (Subramani et al., 2021), we considered the added time due to Mish as an acceptable trade-off for better accuracy. Our experiments showed that training ImageNette on ResNet-9 with ReLU takes 25.32s per epoch and $\sim 0.4s$ more with Mish (25.75s). Differentially private

training with the same setup results in 29.50s for ReLU and 29.96s for Mish, which is an increase of $\sim 4s$ compared to the non-private training.

5.2.5. Influence of batch size

Works by Dörmann et al. (2021), Kurakin et al. (2022), and De et al. (2022) show the strong influence of batch size to training performance under DP. Dörmann et al. (2021) argue that there are two different kinds of noise added in the training process: the inherent sampling noise introduced by the use of mini-batches, and the added Gaussian noise from DP-SGD. For ordinary training of neural networks without DP, mini-batches introduce a form of sampling noise that acts as regularization. Since the underlying mechanism that introduces privacy already adds a significant amount of noise (depending on the privacy budget), authors argue that other forms of noise should be reduced. Figure 5.1 shows the results of the grid search over the learning rate and batch size for CIFAR-10 trained on ResNet-9 for 25 epochs and a privacy budget of $\varepsilon = 7.5$. Our experiments show that the optimal batch size for this combination of hyperparameters results in 1024.

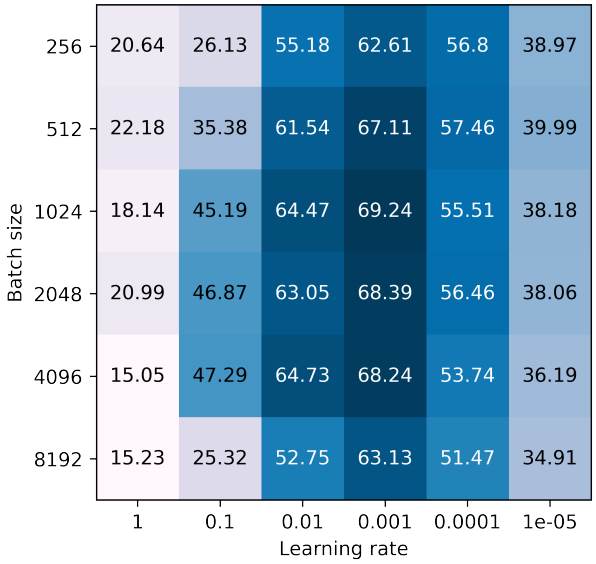


Figure 5.1.: Grid search over the learning rate and batch size of differentially private training ($\varepsilon = 7.5$) of CIFAR-10 with ResNet-9 for 25 epochs. Values are the validation accuracy in %.

5.2.6. Robustness of NAdam optimizer

All experiments were conducted with the NAdam optimizer that combines the well-known Adam optimization algorithm with Nesterov's accelerated gradient (Dozat, 2016). The results in Figure 5.1 show that NAdam is very robust to varying batch sizes and performs best with a learning rate of 0.001 in all scenarios. Especially in private training, finding a suitable combination of hyperparameters is a time-consuming balancing act. Simplifying the search by choosing a robust optimizer significantly reduces computational complexity and time spent.

5.2.7. Transfer learning

A common approach to enhance the performance of differentially private deep learning is to pre-train the model on a publicly available dataset and fine-tune the weights on the private dataset under the given privacy bounds. We evaluate the effect of ScaleNorm on the training performance when transfer learning is applied. For the experiments, we pre-train a ResNet-9 on CIFAR-10 or CIFAR-100 without privacy and fine-tune it on the respective other dataset using DP-SGD.

In the context of transfer learning, pre-training denotes training a model on a specific dataset and saving the weights. It is crucial to avoid overfitting since the model should later generalize well on a completely different dataset. For fine-tuning, the pre-trained weights are loaded into a model of the same architecture, then some layers are frozen (i.e. non-trainable), and the usual training of the second dataset commences. The updates then only happen on the trainable layers. Depending on the similarity of the two datasets, the domain shift varies, which determines the success of transfer learning. A larger domain shift can also be addressed by freezing fewer layers, whereas for a small shift it is often enough to only fine-tune the classifier.

We also conducted another experiment trying out a novel idea of pre-training with DP-SGD as well. This would address the limiting factor for transfer learning that one has to find a public dataset that is similar enough to benefit the private dataset. By showing how transfer learning can be applied to two private datasets, the bottleneck would be reduced. We only show initial experiments and leave it to future research to explore this idea in depth.

5.3. Results

5.3.1. Optimal replacement for Batch Normalisation

Table 5.3 shows the results of different normalization layers (IN, GN with various group sizes and LN) trained on ImageNette and visualizes the great improvements in validation accuracy that can be achieved when grid searching over the possible layers. Results of training on CIFAR-10 and Tiny ImageNet show similar dynamics and can be found in the Appendix A. The overall trend shows that IN is usually a poor choice and LN only benefits the performance of larger networks, which results in difficult learning circumstances as discussed above, and therefore a low accuracy overall. GN with a group size of 16 or 32 achieves the best results for ResNet-9.

The results of varying group sizes within the different stages of the network only show small improvements ($< 1\%$ in all cases, e.g. 0.8% for ResNet-9 on CIFAR-10) and do not justify the large overhead of the hyperparameter searches for trying out all group size combinations. Even with the segregation of the network into 4 parts having an equal group size, it would take 1,296 runs to try out all combinations. We leave it to future research to develop heuristics that help to leverage the benefits of varying group sizes and reduce the added complexity. As a general guideline for choosing the normalization layer, we recommend the following strategy:

- Start with GN and group size 32;
- Grid search over group sizes [8, 16, 32, 64];
- If group size 8 performs best, include IN in grid search.

5.3.2. Effects of ScaleNorm on ResNets

ScaleNorm improves the validation accuracy of ResNet-9 for all three datasets (CIFAR-10, ImageNette and Tiny ImageNet), as reported in Table 5.4. The experiments with ResNet-50 show an increase by ScaleNorm as well but with a large performance drop compared to ResNet-9. This "curse of dimensionality" results from the proportional increase of the added noise with the dimension of the gradient (Subramani et al., 2021; Tramèr and Boneh, 2021; Kurakin et al., 2022; De et al., 2022). Highly over-parameterized models, therefore, suffer from more added noise and show a significant reduction of validation accuracy.

We used the NAdam optimizer with a learning rate of 0.001 and a scheduler that reduces the learning rate by half after no improvement in the validation loss is registered for three epochs. The ℓ_2-clipping-norm is set to 1.5 and the noise multiplier was

Table 5.3.: Model accuracy in % on the ImageNette dataset using the ResNet-9, and ResNet-50 model architectures with varying numbers of GN groups as well as IN and LN. Experiments were conducted either with (✓) or without (✗) ScaleNorm. All results are reported as mean and median values of three runs.

$\varepsilon = 3$								
Model	ScaleNorm		LN	8	16	32	64	IN
ResNet-9	✗	mean	52.6	53.8	**55.1**	54.6	52.8	45.6
	✓		55.0	55.9	56.5	**56.6**	53.7	38.6
	✗	median	52.4	54.1	**55.5**	54.8	52.8	45.6
	✓		55.2	56.0	**56.6**	56.1	53.3	38.7
$\varepsilon = 7.5$								
Model	ScaleNorm		LN	8	16	32	64	IN
ResNet-9	✗	mean	61.7	62.6	**63.7**	62.7	60.6	54.7
	✓		63.2	64.6	**64.7**	64.2	62.0	48.2
	✗	median	61.9	62.1	**63.8**	62.6	61.4	57.2
	✓		63.3	64.4	**64.8**	**64.8**	64.0	50.9
ResNet-50	✗	mean	**43.2**	41.3	39.4	33.4	24.1	10.9
	✓		**44.5**	40.8	34.3	25.1	13.4	10.3
	✗	median	**42.8**	42.2	39.8	33.7	24.9	10.6
	✓		**44.97**	39.9	36.3	24.8	13.4	10.1
$\varepsilon = 10$								
Model	ScaleNorm		LN	8	16	32	64	IN
ResNet-9	✗	mean	63.8	63.6	**64.6**	**64.6**	63.3	59.9
	✓		65.9	66.7	67.1	**67.3**	66.0	53.9
	✗	median	64.2	63.6	64.6	**65.0**	63.6	60.1
	✓		66.1	66.3	66.8	**67.1**	65.9	53.9

Table 5.4.: Model accuracy in % on the CIFAR-10, ImageNette and Tiny ImageNet dataset using the ResNet-9 model architectures with the best performing number normalization layer (IN, GN with different group sizes, and LN). Experiments were conducted either with (✓) or without (✗) ScaleNorm. All results are reported as median values of three runs.

CIFAR-10					
Model	ScaleNorm		3	7.5	10
ResNet-9	✗	Top-1	64.83	71.42	72.41
	✓		**65.03**	**71.74**	**72.95**
ImageNette					
Model	ScaleNorm		3	7.5	10
ResNet-9	✗	Top-1	55.52	63.77	64.99
	✓		**56.59**	**64.84**	**67.11**
ResNet-50	✗	Top-1		42.80	
	✓			**44.97**	
Tiny ImageNet					
Model	ScaleNorm		5	10	70
ResNet-9	✗	Top-1	14.21	18.71	24.71
	✓		**15.18**	**19.39**	**25.81**
	✗	Top-5	34.39	40.47	48.31
	✓		**35.98**	**41.83**	**50.77**

calculated by Opacus (Yousefpour et al., 2021) for the given target ε and δ of 10^{-5}. All experiments were conducted with a batch size of 1024 and Mish as activation function. For the datasets CIFAR-10 and ImageNette, we trained the models for 50 epochs and for Tiny ImageNet for 90, following the guideline that for a given epsilon value longer training and therefore higher noise per step results in a higher overall accuracy than shorter training and less noise per step (Kurakin et al., 2022). We grid-searched over the group size of the Group Normalization layer (including 1 for Layer Normalization and the maximum group size for Instance Normalization) and report all results as median over three runs with fixed randomly selected seeds. Similar to Kurakin et al. (2022) and De et al. (2022), we reset the privacy budget after every hyperparameter search. We

chose the privacy levels according to related works of Tramèr and Boneh (2021) ($\varepsilon = 3$), Dörmann et al. (2021) ($\varepsilon = 7.5$) and Kurakin et al. (2022) ($\varepsilon = 10$). Like ImageNet, Tiny ImageNet is a difficult task to train with DP-SGD under small ε, so we evaluated its performance for ε of 5, 10 and 70 similar to Kurakin et al. (2022).

Of note, our recent work (Klause et al., 2022) shows that ScaleNorm can even improve the current state-of-the-art result by De et al. (2022) on CIFAR-10 by 1.25% and achieve a validation accuracy of 82.5% for a privacy budget of $\varepsilon = 8$. De et al. (2022) make use of the WideResNet architecture with depth 16 and width 4 (WRN-16/4) that has a total of 2,752,506 parameters. Of note, the training took over 16h on a dual GPU system, whereas our ResNet-9 architecture only need 22 minutes on a single GPU (NVidia Quadro RTX 8000), which is a 40$\times$ increase in training time for a 10% gain in accuracy (Klause et al., 2022).

5.3.3. Transfer learning with ScaleNorm

We tried different settings for transfer learning and report the best performing ones in Table 5.5. Here, we used GN with 32 groups for the pre-training and fine-tuning the last four layers with DP ($\varepsilon = 10$). We also ran experiments with BN: first, we used BN in the pre-training and fine-tuned only the classification layers, and second, we loaded the parameters of the BN layer into the respective GN layers and then fine-tuned the last four layers. These attempts try to utilize the better performance of BN for DP. Both setups resulted in slightly lower accuracies than when GN was used for pre-training and fine-tuning. We thus do not generally recommend these techniques.

In a second experiment, we pre-trained CIFAR-10 with DP ($\varepsilon = 10$) and then fine-tuned the model (ResNet-9) on the last four layers on CIFAR-100 also with DP ($\varepsilon = 10$). The result shows only a slight improvement to the baseline of 31.62 top-1 and 57.36 top-5 accuracy. Table 5.5 shows that ScaleNorm increases the validation accuracy in all scenarios. For these experiments, we used the ResNet-9 architecture, NAdam as an optimizer with a learning rate of 0.001, and a scheduler that reduces the learning rate by 0.5 after plateauing for 2 epochs. We pre-trained on CIFAR-10 for 30 epochs and consecutively fine-tuned on CIFAR-100 for 90 epochs. For CIFAR-10, we pre-trained CIFAR-100 for 50 epochs and fine-tuned CIFAR-10 for 50 epochs.

Table 5.5.: Model accuracy in % of transfer learning on CIFAR-10 with a pre-trained model on CIFAR-100 and on CIFAR-100 with a pre-trained model on CIFAR-10. Pre-training either happened without preserving privacy or with a privacy budget of $\varepsilon = 10$. Models were fine-tuned with DP and a budget of $\varepsilon = 10$ in both cases. Results show how networks with (✓) or without (✗) ScaleNorm perform and are reported as median values of three runs.

Dataset	Pre-training dataset	ScaleNorm		non-DP pre-training	DP pre-training
CIFAR-10	CIFAR-100	✗	top-1	76.75	
		✓		**77.41**	
CIFAR-100	CIFAR-10	✗	top-1	43.37	32.31
		✓		**44.68**	**33.59**
		✗	top-5	70.36	58.82
		✓		**71.75**	**61.56**

6. ScaleNorm for image segmentation

6.1. Dataset

As ScaleNorm has shown to be a useful architectural adaptation for training image classifiers with differential privacy, we also evaluated its use for image segmentation. We ran our experiments on the Medical Segmentation Decathlon *liver* dataset (Simpson et al., 2019) that consists of 131 training datasets (patients) and 70 validation datasets of liver computed tomography scans, originally sized 512×512, which we resized to 256×256 pixels. The task is to output a segmentation mask for the liver vs. the background (see Figure 6.1).

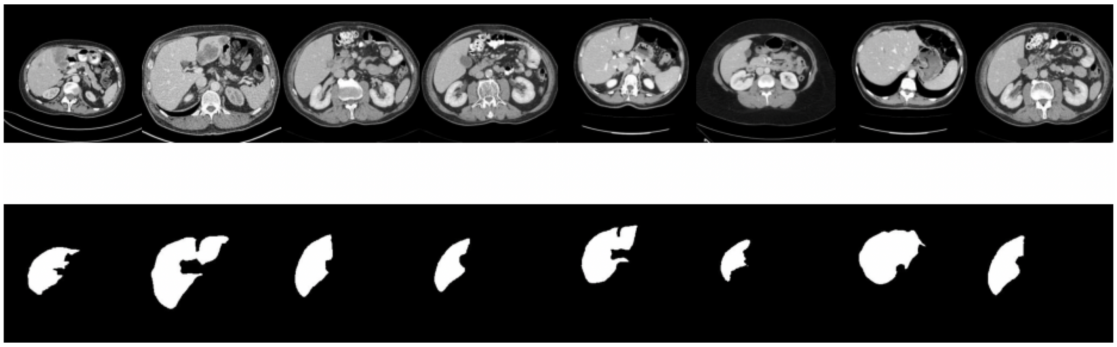


Figure 6.1.: The upper row shows sample images from the liver dataset (Simpson et al., 2019) and the lower row the corresponding masks that are the segmented livers.

6.2. Model architectures

We evaluated the effects of ScaleNorm for segmentation tasks on two different architectures: UNet (Ronneberger et al., 2015) and LinkNet (Chaurasia and Culurciello, 2017). As we found in previous experiments, smaller networks tend to perform better than large ones so we constructed a UNet-9 and LinkNet-9. The networks differ only in the operation that combines the long skip connection with the normal pass, which is either

an addition (LinkNet) or a concatenation (UNet). We added the ScaleNorm layers to both architectures for both the short and long skip connections and call these networks ScaleUNet and ScaleLinkNet respectively. We exemplary show this for UNet9 in Figure 6.2. The PyTorch implementation of the networks can be found here[2][3].

We used the Mish activation function for all experiments. As an optimizer, we chose NAdam (Dozat, 2016) and used the Dice Loss as a loss function, which is region-based and a typical choice for segmentation. To the initial learning rate of 0.001, we applied a learning rate scheduler that reduces the value by half if there is no improvement for 4 epochs.

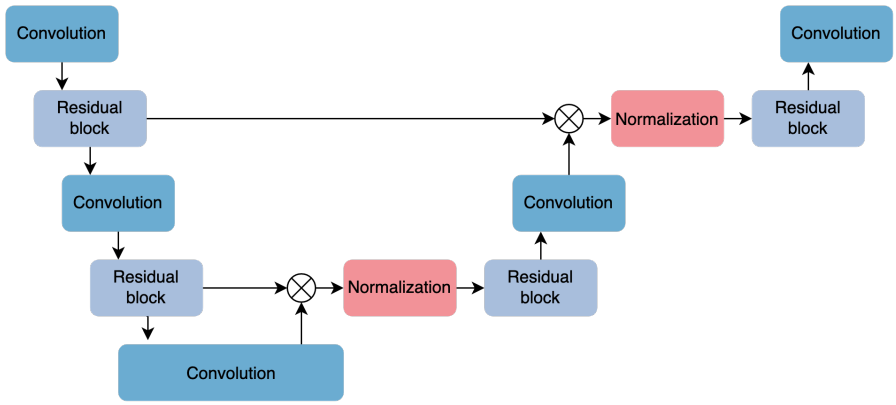


Figure 6.2.: Diagram shows the ScaleUNet-9 architecture with the additional normalization layers after the long skip connections. Inside the residual block Scale Normalization is applied as depicted in Figure 3.1.

6.3. Results

For the low ε value of 3, ScaleNorm performs equal to no ScaleNorm, and for the very low ε value of 0.5, a performance decrease is visible (Table 6.1). We explain these results by the fact that for reasonable values of ε the semantic segmentation is a rather simple task and therefore our architecture modification does not show a great impact. This can also be seen in the different group sizes that result in little to no change in the validation dice coefficient. For a very low ε, the added noise dominates the training and the performance drops. This results in a challenging learning environment that can not be further improved by ScaleNorm.

[2] https://github.com/mrchntia/ScaleNorm/blob/main/semantic_segmentation/unet9.py
[3] https://github.com/mrchntia/ScaleNorm/blob/main/semantic_segmentation/linknet9.py

If the noise is not dominating, as in our experiments for $\varepsilon = 3$, we advise the use of ScaleNorm to get minor performance increases or the same results. The effect of ScaleNorm on the activations can be seen in Figure 6.3, which shows the activations of the LinkNet trained under $\varepsilon = 3$ for a residual block and a long skip connection. Similar to Figure 3.2 we see that ScaleNorm makes the activations more symmetric (standard deviation of 1 and sample mean of 0).

Table 6.1.: Validation dice score for ε of 3 and 0.5 on the liver dataset using UNet-9 and LinkNet-9 with varying group sizes for GN and (✓) or without (✗) ScaleNorm. All results are reported as mean and median values of three runs.

Model	ε	ScaleNorm		8	16	32
UNet-9	3	✗	mean	81.2	81.3	81.0
		✓		**81.4**	81.3	81.1
		✗	median	81.1	81.2	81.0
		✓		**81.4**	**81.4**	81.1
LinkNet-9	3	✗	mean	82.1	82.1	81.5
		✓		82.1	**82.2**	80.6
		✗	median	82.2	82.3	81.6
		✓		82.1	**82.4**	80.6
	0.5	✗	mean	**74.5**	73.1	71.8
		✓		73.5	72.8	71.2
		✗	median	**74.9**	72.9	71.9
		✓		73.8	72.4	71.0

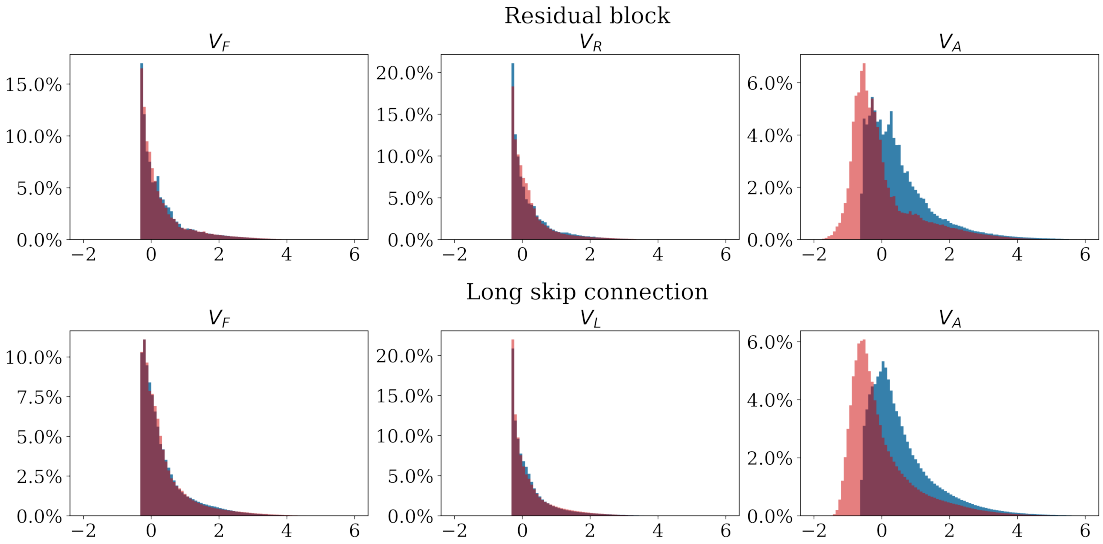


Figure 6.3.: Activation histograms of a residual block and a long skip connection of LinkNet-9 (blue) vs. ScaleLinkNet-9 (red) trained with $\varepsilon = 3$.

7. Discussion and conclusion

Summary Our work proposes a simple yet effective architectural adaptation to improve differentially private learning with residual networks. ScaleNorm requires only a few lines of code changes and can be applied to any ResNet for vision tasks. We show how ScaleNorm can also benefit transfer learning. Our technique may be less effective when applied to UNet and LinkNet used for image segmentation.

Additionally, we studied the effect of different normalization layers (IN, LN and GN with various group sizes) and show their substantial impact on the model accuracy. We also tried varying group sizes throughout the network and conclude that the computational complexity does not justify the small performance gain. As a general note, we attempted to limit the CO_2 and thermal emissions from our experiments by keeping training time as low as possible (all experiments were conducted on a GPU that takes 250W). To conclude our findings on the replacement for batch normalization layers, we provide guidelines on how to choose the right alternative normalization layer that is compatible with DP.

Limitations As a limitation of our work, we note the lack of a theoretical convergence analysis for why ScaleNorm improves model accuracy, which we view as a promising avenue for future work. We also note that the effect of ScaleNorm on image segmentation has only been tested on one dataset. Ideally, these assumptions should be verified by training at least one more dataset with our architectural adaptation. Lastly, our performed hyperparameter searches violate the DP guarantees as already mentioned.

Contextualization Our results on the effects of ScaleNorm show that good normalization is probably very important for effective DP training. Generally speaking, ScaleNorm is only a small step to better understand the architectural needs of differentially private deep learning. The DP research community is slowly gaining knowledge about the different training dynamics of private and non-private training. This makes us hopeful that the privacy-utility gap can be reduced even more as DP gains more popularity.

Future work Future research building directly on this work could analyze the combination of ScaleNorm with DenseNets, which has not been explored so far, or with

ResNet-18, a larger model than our ResNet-9, but still relatively small, and used in prior work (Kurakin et al., 2022). Other interesting experiments would include determining optimal weight initialization strategies for the activation function Mish, and a more thorough evaluation of its self-regularizing effect.

Since the trend of creating larger and larger networks in non-private training very quickly results in significant performance decreases for private training, the question remains: *how can we optimally leverage smaller networks?* Research that has the aim to provide good generalization on difficult datasets with a minimum of trainable parameters would address this important challenge. Tramèr and Boneh (2021) show interesting research in this area and also the work by Remerscheid et al. (2022) demonstrates how a model specially designed for DP training could be constructed. Future work could explore this topic further by investigating modifications to the convolutional layers (e.g. dilated convolutions or different kernel sizes) and their effects on private training. The contrary research field would be to better understand why larger networks train worse. An extensive analysis of the loss landscape under differential privacy would be a natural starting point of such an investigation.

A. Results for optimal group size

Table A.1.: Model accuracy in % on the CIFAR-10 dataset using the ResNet-9 model architectures with varying numbers of GN groups as well as IN and LN. Experiments were conducted either with (✓) or without (✗) ScaleNorm. All results are reported as mean and median values of three runs.

$\epsilon = 3$								
Model	ScaleNorm		LN	8	16	32	64	IN
ResNet-9	✗	mean	62.6	63.0	64.0	64.3	65.0	58.9
	✓		63.5	64.2	64.4	**65.4**	64.9	58.2
	✗	median	62.5	63.2	64.4	64.8	65.0	58.5
	✓		63.7	64.2	64.5	**65.5**	64.8	58.0
$\epsilon = 7.5$								
Model	ScaleNorm		LN	8	16	32	64	IN
ResNet-9	✗	mean	69.8	69.8	71.2	71.3	70.6	66.7
	✓		70.5	71.3	71.3	**71.7**	71.1	65.6
	✗	median	69.7	69.9	71.1	71.4	70.6	66.7
	✓		70.6	71.4	71.3	**71.7**	71.4	65.6
$\epsilon = 10$								
Model	ScaleNorm		LN	8	16	32	64	IN
ResNet-9	✗	mean	71.8	71.3	72.2	72.4	71.6	67.9
	✓		72.1	72.3	**73.0**	72.7	72.1	67.0
	✗	median	71.3	71.2	72.4	72.4	71.8	68.1
	✓		71.9	72.3	**73.0**	72.9	72.1	67.1

Table A.2.: Model accuracy in % on the Tiny ImageNet dataset using the ResNet-9 model architectures with varying numbers of GN groups as well as IN and LN. Experiments were conducted either with (✓) or without (✗) ScaleNorm. All results are reported as median values of three runs.

$\epsilon = 5$								
Model	ScaleNorm		LN	8	16	32	64	IN
ResNet-9	✗	Top-1	13.7	14.2	13.0	12.4	9.3	3.6
	✓		14.7	**15.2**	14.9	13.3	9.8	3.3
	✗	Top-5	32.5	34.4	32.1	30.7	24.2	11.2
	✓		34.0	**36.0**	35.3	32.5	26.2	10.7
$\epsilon = 10$								
Model	ScaleNorm		LN	8	16	32	64	IN
ResNet-9	✗	Top-1	17.2	18.7	17.1	15.9	12.4	6.1
	✓		18.5	**19.4**	18.5	17.7	13.4	5.3
	✗	Top-5	38.1	40.5	38.5	36.9	30.5	17.3
	✓		40.8	**41.8**	40.9	39.4	32.9	15.6
$\epsilon = 70$								
Model	ScaleNorm		LN	8	16	32	64	IN
ResNet-9	✗	Top-1	23.7	24.7	23.6	23.1	19.5	12.2
	✓		**25.8**	**25.8**	24.7	24.5	20.8	12.2
	✗	Top-5	46.7	48.3	47.4	46.4	42.3	28.9
	✓		49.7	**50.8**	48.5	48.9	43.9	29.3

List of Figures

List of Tables

Glossary

BN Batch Normalization (Ioffe and Szegedy, 2015). 3, 11, 15, 22

DP Differential privacy, differentially private (Dwork, 2006). 1–6, 8, 11–13, 15–17, 22, 23, 28, 29, 33

DP-SGD Differentially private stochastic gradient descent (Abadi et al., 2016). 3, 6, 7, 11–13, 17, 18, 22

GN Group Normalization (Wu and He, 2018). 15, 16, 19, 22, 26, 28, 30, 31, 33, 34

IN Instance Normalization (Ulyanov et al., 2016). 15, 16, 19, 20, 28, 30, 31, 34

LinkNet Neural network architecture for image segmentation (Chaurasia and Culurciello, 2017). 2, 24–28, 32, 33, 35

LN Layer Normalization (Ba et al., 2016). 15, 16, 19, 20, 28, 30, 31, 34

ML Machine learning. 1

ResNet Neural network architecture that consists of so-called residual blocks (Zhang et al., 2019). 2, 3, 8–11, 14–19, 22, 28–35

ScaleLinkNet Scale Normalization applied to the LinkNet architecture. 25, 27, 32

ScaleNorm Novel architectural adaptation called Scale Normalization that adds an additional normalization layer after the residual block. 2, 9, 10, 15, 18, 19, 22–26, 28, 30–34

ScaleResNet Scale Normalization applied to the ResNet architecture. 9, 10, 32

ScaleUNet Scale Normalization applied to the UNet architecture. 25, 32

SGD Stochastic gradient descent. 6, 11

UNet Neural network architecture for image segmentation (Ronneberger et al., 2015). 2, 24–26, 28, 33, 35

Bibliography

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*, pages 308–318, 2016.

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv*, 2016.

B. Balle and Y.-X. Wang. Improving the gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. *ICML International Conference on Machine Learning*, 2018.

B. Balle, G. Cherubin, and J. Hayes. Reconstructing training data with informed adversaries. *IEEE Symposium on Security and Privacy (SP)*, 2022.

N. Carlini, F. Tramer, E. Wallace, M. Jagielski, A. Herbert-Voss, K. Lee, A. Roberts, T. Brown, D. Song, U. Erlingsson, A. Oprea, and C. Raffel. Extracting training data from large language models. *arXiv*, 2021.

N. Carlini, S. Chien, M. Nasr, S. Song, A. Terzis, and F. Tramer. Membership inference attacks from first principles. *arXiv*, 2022.

A. Chaurasia and E. Culurciello. LinkNet: Exploiting encoder representations for efficient semantic segmentation. In *IEEE Visual Communications and Image Processing (VCIP)*, 2017.

A. Cheng, J. Wang, X. S. Zhang, Q. Chen, P. Wang, and J. Cheng. Dpnas: Neural architecture search for deep learning with differential privacy. *AAAI Conference on Artificial Intelligence*, 2021.

C. A. Choquette-Choo, F. Tramer, N. Carlini, and N. Papernot. Label-only membership inference attacks. *ICML International Conference on Machine Learning*, 2021.

A. Davody, D. I. Adelani, T. Kleinbauer, and D. Klakow. On the effect of normalization layers on differentially private training of deep neural networks. *arXiv*, 2021.

S. De, L. Berrada, J. Hayes, S. L. Smith, and B. Balle. Unlocking high-accuracy differentially private image classification through scale. *arXiv*, 2022.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

F. Dörmann, O. Frisk, L. N. Andersen, and C. F. Pedersen. Not all noise is accounted equally: How differentially private learning benefits from large sampling rates. In *IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2021.

T. Dozat. Incorporating Nesterov Momentum into Adam. *ICLR Workshop Track*, 2016.

C. Dwork. Differentially privacy. *ICALP International Colloquium on Automata, Languages and Programming*, page 1–12, 2006.

C. Dwork and A. Roth. Self-normalizing neural networks. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

V. Feldman. Does learning require memorization? a short tale about a long tail. *STOC Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 2019.

S. Fort, A. Brock, R. Pascanu, S. De, and S. L. Smith. Drawing multiple augmentation samples per image during training efficiently decreases test error. *arXiv*, 2021.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Google. Tensorflow privacy, 2018. URL `https://github.com/tensorflow/privacy`.

K. He and J. Sun. Convolutional neural networks at constrained time cost. *arXiv*, 2014.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

J. Howard. Imagenette, 2019. URL `https://github.com/fastai/imagenette`.

G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. *CVPR Conference on Computer Vision and Pattern Recognition*, 2017.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML International Conference on Machine Learning*, pages 448–456. PMLR Proceedings of Machine Learning Research, 2015.

G. Kaissis, M. Makowski, D. Rückert, and R. Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6): 305–311, 2020.

G. Kaissis, A. Ziller, J. Passerat-Palmbach, T. Ryffel, D. Usynin, A. Trask, I. Lima, J. Mancuso, F. Jungmann, M.-M. Steinborn, A. Saleh, M. Makowski, D. Rueckert, and R. Braren. End-to-end privacy preserving deep learning on multi-institutional medical imaging. *Nature Machine Intelligence*, 3(6):473–484, 2021. doi: 10.1038/s42256-021-00337-8.

G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. *NeurIPS Conference on Neural Information Processing System, Workshop*, 2017.

H. Klause, A. Ziller, D. Rueckert, K. Hammernik, and G. Kaissis. Differentially private training of residual networks with scale normalisation. *arXiv*, 2022.

A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

A. Kurakin, S. Song, S. Chien, R. Geambasu, A. Terzis, and A. Thakurta. Toward training at imagenet scale with differential privacy. *arXiv*, 2022.

Y. Le and X. S. Yang. Tiny imagenet visual recognition challenge. *Stanford University*, 2015.

Y. Liu, R. Wen, X. He, A. Salem, Z. Zhang, M. Backes, E. D. Cristofaro, M. Fritz, and Y. Zhang. Ml-doctor: Holistic risk assessment of inference attacks against machine learning models. *arXiv*, 2021.

E. S. Lubana, R. P. Dick, and H. Tanaka. Beyond batchnorm: Towards a general understanding of normalization in deep learning. *NeurIPS Conference on Neural Information Processing System*, 2021.

Z. Luo, D. J. Wu, E. Adeli, and L. Fei-Fei. Scalable differential privacy with sparse network finetuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5057–5066, 2021.

I. Mironov. Rényi differential privacy. In *IEEE Computer Security Foundations Symposium (CSF)*, 2017.

D. Misra. Mish: A self regularized non-monotonic activation function. *BMVC 31st British Machine Vision Virtual Conference*, 2020.

F. Morsbach, T. Dehling, and A. Sunyaev. Architecture matters: Investigating the influence of differential privacy on neural network design. *NeurIPS Conference on Neural Information Processing System, Workshop*, 2021.

A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, pages 111–125, 2008.

M. Nasr, R. Shokri, and A. houmansadr. Improving deep learning with differential privacy using gradient encoding and denoising. *arXiv*, 2020.

E. Oyallon, S. Zagoruyko, G. Huang, N. Komodakis, S. Lacoste-Julien, M. Blaschko, and E. Belilovsky. Scattering networks for hybrid representation learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2208–2221, 2019.

N. Papernot and T. Steinke. Hyperparameter tuning with renyi differential privacy. *ICLR International Conference on Learning Representations*, 2022.

N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and U. Erlingsson. Scalable private learning with PATE. *ICLR International Conference on Learning Representations*, 2018.

N. Papernot, S. Chien, S. Song, A. Thakurta, and U. Erlingsson. Making the shoe fit: Architectures, initializations, and tuning for learning with privacy. *OpenReview*, 2020.

N. Papernot, A. Thakurta, S. Song, S. Chien, and U. Erlingsson. Tempered sigmoid activations for deep learning with differential privacy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9312–9321, 2021.

PyTorch. resnet.py, Dec 16, 2021. URL `https://github.com/pytorch/vision/blob/eac3dc7bab436725b0ba65e556d3a6ffd43c24e1/torchvision/models/resnet.py`.

N. W. Remerscheid, A. Ziller, D. Rueckert, and G. Kaissis. Smoothnets: Optimizing cnn architecture design for differentially private deep learning. *arXiv*, 2022.

O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *arXiv*, 2015.

C. Schwarz, W. Kremers, T. Therneau, R. Sharp, J. Gunter, P. Vemuri, A. Arani, A. Spychalla, K. Kantarci, D. Knopman, R. Petersen, and C. Jack. Identification of anonymous mri research participants with face-recognition software. *New England Journal of Medicine*, 381:1684–1686, 2019.

A. L. Simpson, M. Antonelli, S. Bakas, M. Bilello, K. Farahani, B. van Ginneken, A. Kopp-Schneider, B. A. Landman, G. Litjens, B. H. Menze, O. Ronneberger, R. M. Summers, P. Bilic, P. F. Christ, R. K. G. Do, M. Gollub, J. Golia-Pernicka, S. Heckers, W. R. Jarnagin, M. McHugo, S. Napel, E. Vorontsov, L. Maier-Hein, and M. J. Cardoso. A large annotated medical image dataset for the development and evaluation of segmentation algorithms. *arXiv*, 2019.

I. Singh, H. Zhou, K. Yang, M. Ding, B. Lin, and P. Xie. Differentially-private federated neural architecture search. *arXiv*, 2020.

R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *ICML International Conference on Machine Learning, Deep Learning workshop*, 2015.

P. Subramani, N. Vadivelu, and G. Kamath. Enabling fast differentially private sgd via just-in-time compilation and vectorization. *Advances in Neural Information Processing Systems*, 34, 2021.

F. Tramèr and D. Boneh. Differentially private learning needs better features (or much more data). In *ICLR International Conference on Learning Representations*, 2021.

D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv*, 2016.

B. Wang, Q. Gu, M. Boedihardjo, L. Wang, F. Barekat, and S. J. Osher. DP-LSSGD: A stochastic optimization method to lift the utility in privacy-preserving ERM. In J. Lu and R. Ward, editors, *Proceedings of The First Mathematical and Scientific Machine Learning Conference*, volume 107, pages 328–351. PMLR Proceedings of Machine Learning Research, 2020.

Y. Wu and K. He. Group normalization. In *ECCV Proceedings of the European conference on computer vision*, pages 3–19, 2018.

Z. Xu, S. Shi, A. X. Liu, J. Zhao, and L. Chen. An adaptive and fast convergent approach to differentially private deep learning. *IEEE Institute of Electrical and Electronics Engineers*, 2019.

Z. Yao, A. Gholami, K. Keutzer, and M. Mahoney. Pyhessian: Neural networks through the lens of the hessian. *IEEE International Conference on Big Data*, 2019.

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. *arXiv*, 2021.

H. Zhang, Y. N. Dauphin, and T. Ma. Residual learning without normalization via better initialization. In *ICLR International Conference on Learning Representations*, 2019.

Y. Zhu, X. Yu, M. Chandraker, and Y.-X. Wang. Private-KNN: Practical differential privacy for computer vision. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11854–11862, 2020.