Introduction to Machine Learning

Project phase 0
All you need is Privacy!

Dr.R.Amiri

Sharif University of Technology

Electrical Engineering

Nikoo Moradi _ 400101934

Tina Halimi _ 400101078

Date: June 29, 2024

Contents

1 Introduction

As we navigate through the vast ocean of data in this digital age, the power of machine learning and deep learning models has become increasingly evident. These models have revolutionized data analysis, providing us with insights that were previously unimaginable. However, with great power comes great responsibility. As we delve deeper into the world of data, one question becomes increasingly important: How do we ensure the privacy of the data we analyze?

This project aims to address this critical issue. We will explore various aspects of privacy in machine learning, focusing on two key areas.

In the first section, we will introduce you to the concept of Machine Unlearning , a process designed to uphold the right to be forgotten. This right is a crucial aspect of data privacy, ensuring that individuals can request their data to be removed from a model or system.

In the final section, we will delve into the realm of training private models. We will also discuss a specific adversarial attack known as the Membership Inference Attack. This attack poses a significant threat to data privacy, and understanding it is key to developing robust, privacy-preserving machine learning models.

By the end of this project, you will have a solid understanding of these important concepts and be well-equipped to navigate the complex landscape of privacy in machine learning. Let's start this journey together!

2 Machine Unlearning

Theory Question 1. Name 3 aggregation methods you would propose for this algorithm. To your best knowledge, reason each method's strengths and weaknesses.

1. Majority Voting

- **Strengths**: Simple to implement and understand. Ensures that the final prediction is a consensus of multiple models, which can help mitigate the effect of any single model's bias or error.
- **Weaknesses**: Can be inefficient if the models have widely varying outputs, leading to indecision or less accurate results. It may not effectively leverage the strengths of individual models, especially if some models are more accurate than others.

2. Weighted Averaging

- **Strengths**: Takes into account the performance of individual models by assigning higher weights to more accurate models. This method can improve the overall performance by leveraging the strengths of better-performing models.
- **Weaknesses**: Requires an additional step to determine the weights, which can be complex and computationally expensive. If the weights are not accurately determined, it can lead to suboptimal performance.

3. Stacking (Stacked Generalization)

- **Strengths**: Combines multiple models by training a meta-model to make predictions based on the outputs of base models. This method can significantly enhance performance by capturing complex patterns and relationships.
- **Weaknesses**: More computationally intensive and complex to implement compared to majority voting and weighted averaging. Requires a separate validation set to train the meta-model, which can lead to additional data requirements.

Theory Question 2. When will this method be inefficient to use? What do you suggest to mitigate this issue?

The SISA algorithm can be inefficient in scenarios where:

- **High Frequency of Unlearning Requests**: If there are frequent requests to unlearn data, constantly retraining shards and slices can become computationally expensive and time-consuming.
- **Large Datasets**: For very large datasets, the overhead of managing and retraining multiple shards and slices can negate the efficiency gains, leading to significant computational costs and time delays.

Mitigation Suggestions:

- **Incremental Learning Methods**: Implement incremental learning techniques to update models without retraining from scratch. This can help in managing frequent unlearning requests more efficiently.

- **Efficient Data Management**: Use data management strategies like maintaining indices of data points and their corresponding shards/slices to quickly locate and update relevant parts of the model.
- **Batch Unlearning**: Aggregate unlearning requests and process them in batches to minimize the frequency of retraining operations, thereby reducing computational overhead.

Theory Question 3. What metric would you use to evaluate the performance of your unlearning algorithm? Reason your choice!

Metric: **Unlearning Efficiency Score (UES)**

- **Reason**: The Unlearning Efficiency Score would combine factors such as the time taken to unlearn, the computational resources used, and the impact on model accuracy. This composite metric would provide a holistic view of the unlearning process efficiency.
  - **Time Taken**: Measures how quickly the model can unlearn the specified data.
  - **Computational Resources**: Assesses the CPU/GPU time and memory usage during the unlearning process.
  - **Impact on Model Accuracy**: Evaluates the change in model performance before and after unlearning to ensure the unlearning process does not degrade the model's effectiveness.

Using UES, we can balance the trade-off between the efficiency of the unlearning process and the preservation of model accuracy, ensuring a practical and robust evaluation.

Theory Question 4. Discuss the effect of the models' architecture on learning and unlearning time, and performance in both learning and unlearning.

Complexity of Model Architecture:

- **Learning Time**: Complex architectures (e.g., deep neural networks with many layers) typically require more time to train due to increased computational requirements. Simpler models (e.g., linear regression, shallow neural networks) train faster but may not capture intricate patterns as effectively.
- **Unlearning Time**: For complex models, unlearning can be more time-consuming because of the intricate dependencies and the need to retrain multiple layers or components. Simpler models can be unlearned more quickly due to their straightforward structure.

Performance:

- **Learning Performance**: Complex models generally offer higher performance and accuracy as they can learn from large and intricate datasets. However, they are prone to overfitting if not properly regularized.
- **Unlearning Performance**: In complex models, ensuring that unlearning is effective without compromising overall performance is challenging. The unlearning process needs to accurately remove the influence of specific data points while maintaining the model's generalization ability.

To optimize both learning and unlearning performance:

- **Architectural Choices**: Use architectures that balance complexity and efficiency, such as modular networks or ensembles of simpler models.
- **Regularization Techniques**: Implement regularization techniques (e.g., dropout, L2 regularization) to enhance model robustness and facilitate more effective unlearning.
- **Efficient Layer-wise Unlearning**: For deep models, consider unlearning at specific layers where the target data has the most influence, rather than retraining the entire model.

By carefully considering the model architecture and implementing strategies to streamline both learning and unlearning processes, we can achieve efficient and effective privacy-preserving machine learning models.

3 Private Training Models

Theory Question 5. Explain differentially private algorithms and their techniques for training a differentially private model.

Differential Privacy is a framework for quantifying the privacy guarantees provided by an algorithm. It aims to ensure that the removal or addition of a single data point does not significantly affect the output of the algorithm, thereby protecting the privacy of individual data points.

Key Concepts of Differential Privacy

- **Privacy Loss (ϵ)**: It measures the privacy guarantee. A smaller ϵ implies better privacy but may reduce the utility of the model.
- **Sensitivity**: The maximum amount that a single data point can change the output of a function. Lower sensitivity contributes to better privacy.
- **Random Noise Addition**: Noise is added to the data or the output to obscure the contribution of individual data points.

Techniques for Training a Differentially Private Model

1. **Differentially Private Stochastic Gradient Descent (DP-SGD)**
   - **Concept**: A modification of the traditional SGD algorithm that incorporates differential privacy.
   - **Implementation**:
     - **Gradient Clipping**: Each gradient is clipped to a maximum norm to limit the influence of any single data point.
     - **Noise Addition**: Gaussian or Laplace noise is added to the gradient updates to ensure privacy.
   - **Process**:
     (a) Compute gradients for a mini-batch of data.
     (b) Clip the gradients to a specified norm.
     (c) Add noise to the clipped gradients.
     (d) Update the model parameters using these noisy gradients.
   - **Library Support**: Libraries like TensorFlow Privacy and PySyft provide implementations for DP-SGD. For more details, refer to the TensorFlow Privacy library and the PySyft library.
2. **Output Perturbation**
   - **Concept**: Add noise directly to the output of the model.
   - **Implementation**:
     - After training the model, apply a noise mechanism (Gaussian, Laplace) to the final model outputs (e.g., predictions).
     - This technique ensures that the predictions do not reveal sensitive information about individual data points.

3. **Objective Perturbation**
    - **Concept**: Add noise to the objective function used during model training.
    - **Implementation**:
        - Modify the objective function by adding a noise term to it.
        - Train the model using this modified objective function.
    - **Usage**: Commonly used in linear and logistic regression models.

Practical Steps for Implementing Differentially Private Models

1. **Choose the Privacy Budget (ϵ)**: Decide on an appropriate value for ϵ based on the desired trade-off between privacy and utility.
2. **Select an Appropriate Mechanism**: Choose between DP-SGD, output perturbation, objective perturbation, or PATE based on the use case and model type.
3. **Use Existing Libraries**: Leverage libraries like TensorFlow Privacy or PySyft for easy integration of differential privacy into machine learning models.
4. **Monitor and Evaluate**:
    - Continuously monitor the impact of privacy mechanisms on model performance.
    - Evaluate the trade-off between privacy and utility using metrics like accuracy, precision, recall, and the differential privacy budget (ϵ).

Differentially private algorithms provide a robust framework for protecting individual data points in machine learning models. By implementing techniques such as DP-SGD, output perturbation, objective perturbation, and PATE, it is possible to train models that balance the trade-off between privacy and utility effectively. Utilizing these techniques helps ensure that models can perform well while maintaining stringent privacy guarantees.

Theory Question 6. Explain the regularization and normalization techniques used in training a private model. Are these techniques similar to the method of adding noise to the model in differential privacy?

Regularization Techniques

Regularization is a technique used in machine learning to prevent overfitting by adding a penalty to the loss function during model training. This penalty discourages the model from becoming too complex and helps it generalize better to unseen data. In the context of training a private model, regularization can enhance privacy by reducing the model's capacity to memorize individual data points, thus indirectly protecting sensitive information.

1. **L1 Regularization (Lasso)**
    - **Concept**: Adds the absolute value of the magnitude of coefficients as a penalty term to the loss function.
    - **Equation**: $L = L_0 + \lambda \sum |w_i|$
        - L is the total loss.
        - L_0 is the original loss (e.g., mean squared error).

  - λ is the regularization parameter.
  - w_i are the model weights.
- **Effect**: Encourages sparsity, meaning it drives some weights to zero, effectively performing feature selection.
- **Application in Private Models**: By driving some weights to zero, L1 regularization reduces the model complexity. This prevents the model from overfitting to specific sensitive data points, as the model becomes less capable of memorizing the training data. When a model does not memorize specific data points, it becomes harder for an adversary to infer whether a particular data point was used in training, thus enhancing privacy.

2. **L2 Regularization (Ridge)**

- **Concept**: Adds the squared value of the magnitude of coefficients as a penalty term to the loss function.
- **Equation**: $L = L_0 + \lambda \sum w_i^2$
  - Similar notation as L1 regularization.
- **Effect**: Distributes the error among all weights, preventing any single weight from becoming too large.
- **Application in Private Models**: By penalizing large weights, L2 regularization helps in smoothing the decision boundary of the model. This smoothing effect reduces the model's sensitivity to specific data points, as it discourages large adjustments based on individual data points. Consequently, this makes it harder for an adversary to deduce information about individual data points, thus indirectly contributing to data privacy.

Normalization Techniques

Normalization is the process of scaling individual samples to have unit norm. This technique helps in stabilizing and accelerating the training of the model by ensuring that the inputs to each layer have a consistent distribution. In private models, normalization can help in ensuring that no single data point disproportionately influences the model training, thus contributing to privacy.

1. **Batch Normalization**

- **Concept**: Normalizes the input of each mini-batch so that the mean output is close to 0 and the standard deviation is close to 1.
- **Process**:
  (a) Compute the mean and variance of the current mini-batch.
  (b) Normalize the batch using these statistics.
  (c) Apply a scale and shift transformation to the normalized values.
- **Effect**: Reduces internal covariate shift, speeds up training, and can act as a regularizer.
- **Application in Private Models**: By ensuring that each mini-batch is normalized, batch normalization reduces the variance introduced by individual data points. This normalization means that no single data point can disproportionately influence the learning process. As a result, the model is less likely to memorize specific data points, enhancing privacy by making it harder for attackers to extract individual data points from the model's behavior.

2. **Normalization Temperature**

   - **Concept**: Involves adjusting the softmax function used in classification by increasing the normalization temperature.
   - **Effect**: A higher temperature leads to a smoother probability distribution over classes, which can reduce the model's sensitivity to individual data points and thus improve privacy.
   - **Application in Private Models**: Increasing the normalization temperature in the softmax function helps distribute the impact of individual data points more evenly across the output probabilities. This reduces the likelihood that the model will make decisions based on specific, potentially sensitive data points. By smoothing the output distribution, it becomes harder for an attacker to infer whether a particular data point was part of the training set, thus enhancing privacy.

Similarity to Differential Privacy Noise Addition

The techniques of regularization and normalization share similarities with the method of adding noise in differential privacy, particularly in how they limit the model's ability to memorize or overly rely on specific data points. Here are some similarities:

1. **Purpose**

   - **Regularization** and **Normalization**: Primarily aim to improve model generalization and training stability by preventing overfitting and ensuring consistent data distributions.
   - **Differential Privacy Noise Addition**: Aims to protect individual data points' privacy by making it difficult to infer any specific data point's contribution to the model.

2. **Implementation**

   - **Regularization**: Adds penalties based on the magnitude of model parameters, effectively discouraging the model from fitting too closely to any particular data point.
   - **Normalization**: Adjusts the input or intermediate layer distributions to ensure that no single data point has an outsized effect on the model training.
   - **Differential Privacy Noise Addition**: Introduces random noise to gradients, outputs, or data, masking the contributions of individual data points.

3. **Effect**

   - **Regularization**: Limits the capacity of the model to memorize specific data points, thus reducing the risk of overfitting to sensitive data.
   - **Normalization**: Ensures that each data point contributes more uniformly to the training process, reducing sensitivity to individual data points.
   - **Differential Privacy Noise Addition**: Directly obscures the impact of any single data point on the model, providing a strong privacy guarantee.

4. **Indirect Privacy Enhancement**

- Both regularization and normalization, while not explicitly designed for privacy, can contribute to privacy by making models less sensitive to individual data points. This aligns with the goal of differential privacy, which is to prevent the inference of specific data points from the model's output.

In summary, while the primary goals of regularization and normalization are to improve generalization and training stability, their effects indirectly align with the principles of differential privacy by limiting the model's ability to memorize and overly rely on individual data points. This similarity helps in enhancing privacy, making these techniques complementary to differential privacy methods.

Theory Question 7. Find other techniques for training a private model.

In addition to Differently Private Training, regularization, normalization temperature, and adding noise for privacy, several other techniques can be employed to train private models. Here are a few notable methods:

1. Homomorphic Encryption

Concept: Homomorphic encryption allows computations to be performed on encrypted data without needing to decrypt it first. This technique ensures that sensitive data remains confidential during processing.

Implementation:

- Encrypt the data before processing.
- Perform computations on the encrypted data.
- Decrypt the results after the computations are complete.

Effect: This approach ensures that the data remains private throughout the entire processing pipeline, as only encrypted data is exposed during computations.

Application:

- Suitable for scenarios where sensitive data must be processed by potentially untrusted parties or environments.

2. Secure Multi-Party Computation (SMPC)

Concept: SMPC is a cryptographic protocol that enables multiple parties to collaboratively compute a function over their inputs while keeping those inputs private.

Implementation:

- Split data into shares distributed across multiple parties.
- Each party performs computations on their respective shares.
- Combine the results to obtain the final outcome without revealing individual inputs.

Effect: Ensures that no single party has access to the complete dataset, thereby maintaining data privacy.

Application:

- Ideal for collaborative data analysis where parties do not trust each other with their raw data.

3. Federated Learning

Concept: Federated learning is a distributed learning approach where models are trained across multiple devices or servers holding local data samples, without exchanging the data itself.

Implementation:

- Distribute the model to edge devices.
- Train the model locally on each device using local data.
- Aggregate the model updates centrally without transferring the raw data.

Effect: Protects data privacy by keeping raw data localized and only sharing model updates.

Application:

- Commonly used in scenarios where data is distributed across many devices, such as mobile phones or IoT devices.

4. Private Aggregation of Teacher Ensembles (PATE)

Concept: PATE leverages an ensemble of teacher models trained on disjoint subsets of private data. A student model is then trained using the aggregated outputs of these teacher models.

Implementation:

- Train multiple teacher models on different subsets of the data.
- Use a noisy aggregation mechanism to combine the outputs of the teacher models.
- Train a student model using the aggregated outputs as labels.

Effect: Ensures that the student model does not have direct access to the raw data, enhancing privacy.

Application:

- Suitable for training models where the data is sensitive, and direct access to the raw data must be minimized.

5. Knowledge Distillation with Privacy

Concept: Knowledge distillation transfers knowledge from a large, complex model (teacher) to a smaller, simpler model (student). When combined with privacy techniques, it can help in maintaining data privacy.

Implementation:

- Train a teacher model on the full dataset.
- Use differential privacy techniques to ensure the teacher model does not memorize sensitive data.
- Distill the knowledge to a student model by training it on the predictions of the teacher model.

Effect: The student model learns to mimic the teacher model's behavior without directly accessing the raw data, thereby maintaining privacy.

Application:

- Useful in scenarios where model deployment on resource-constrained devices is required while preserving data privacy.

These techniques offer a range of methods for training private models, each with its specific applications and strengths. Homomorphic encryption and SMPC are more suited for cryptographic privacy guarantees, while federated learning focuses on distributing data and adding noise to protect privacy. PATE and knowledge distillation with privacy provide structured approaches to maintaining privacy while training effective models.

4 Membership Inference Attack

Theory Question 8. Explain three ways of generating training data for shadow models.

1. Using Real-World Data: This method involves using real-world datasets that are similar in distribution to the data used to train the target model. By carefully selecting and curating these datasets, shadow models can be trained to closely approximate the target model's behavior.

2. Data Augmentation: This technique involves augmenting existing datasets to create new training examples. Methods such as rotation, scaling, and adding noise can be used to generate diverse training data for the shadow models, ensuring that they capture a wide range of data variations.

3. Synthetic Data Generation: This method involves using generative models to create synthetic datasets that resemble the target model's training data. Techniques like Generative Adversarial Networks (GANs) can be employed to generate high-quality synthetic data that can be used to train shadow models.

Theory Question 9. One of the methods for generating training data for shadow models is using the model to generate synthetic data. Explain the Algorithm of synthetic data generation.

The algorithm for synthetic data generation using Generative Adversarial Networks (GANs) is as follows:

- **Step 1: Initialize the GAN**:
  - Initialize the generator and discriminator networks with random weights.
- **Step 2: Train the Discriminator**:
  - Sample a batch of real data from the training set.
  - Generate a batch of fake data using the generator.
  - Compute the discriminator's loss on the real data and the fake data.
  - Update the discriminator's weights to minimize the loss on the real data and maximize the loss on the fake data.
- **Step 3: Train the Generator**:
  - Sample a batch of random noise vectors.
  - Generate a batch of fake data using the generator.
  - Compute the discriminator's loss on the fake data.
  - Update the generator's weights to minimize the discriminator's loss on the fake data.
- **Step 4: Iterate**:
  - Repeat steps 2 and 3 for a number of iterations until the generator produces high-quality synthetic data that the discriminator cannot easily distinguish from real data.

By following this algorithm, high-quality synthetic data can be generated to train shadow models effectively.

Theory Question 10. Explain how attack model is trained using shadow models.

1. Creating Shadow Models

The first step involves creating several shadow models. These models are meant to mimic the behavior of the target model. Each shadow model is trained on a different subset of data, referred to as the shadow training set. These datasets are disjoint or partially overlapping subsets of data similar to the target model's training data. The idea is to simulate different training scenarios that the target model might have encountered.

2. Training Shadow Models

Using the same machine learning algorithm and platform as the target model, each shadow model is trained independently on its respective shadow training set. This process ensures that the shadow models behave similarly to how the target model would behave if trained on different data.

3. Generating Predictions

After training, each shadow model is used to generate predictions on two types of data:

- Data from its own training set (in-training data).
- Data not seen during training (out-of-training data).

These predictions are collected and labeled. For example, predictions on the in-training data are labeled 'in,' while predictions on out-of-training data are labeled 'out.'

4. Labeling Predictions

The predictions from the shadow models are crucial for training the attack model. Each prediction is paired with a label indicating whether the data point was part of the training set ('in') or not ('out').

5. Training the Attack Model

The labeled predictions from all the shadow models are combined to form the training dataset for the attack model. The attack model is trained to take a prediction vector (output from the shadow models) as input and produce a binary output: 'in' or 'out.' Essentially, the attack model learns to distinguish between the model's behavior on training data and its behavior on new, unseen data.

6. Evaluating the Attack Model

Once trained, the attack model is tested on the target model. This involves querying the target model with new data points and obtaining its prediction vectors. The attack model then processes these prediction vectors to infer whether each data point was part of the target model's training dataset.

Important Concepts

- **Shadow Models**: Multiple models trained to replicate the target model's behavior using different training data subsets.
- **Prediction Vectors**: Outputs from the shadow models for both in-training and out-of-training data, used to train the attack model.
- **In/Out Labels**: Labels indicating whether the data points used to generate predictions were part of the training set or not.
- **Attack Model**: A model trained on labeled predictions from shadow models to infer membership of data points in the training set.

By leveraging shadow models, attackers can effectively train an attack model to recognize patterns in how the target model treats data it has seen before versus data it hasn't, even without direct access to the target model's training data or internal workings. This method is particularly effective in black-box scenarios where the attacker can only interact with the model through its output predictions.

Theory Question 11: Explain the Effect of the Following Concepts on the Accuracy of the Attack Model

1. Effect of the Shadow Training Data Generated Using the Three Methods

- **Using Real-World Data:**
  - **Effect**: Shadow models trained on real-world data tend to produce more accurate attack models. This is because real-world data can capture the true distributions and complexities found in the target model's training data.
  - **Explanation**: When shadow models are trained on data that closely resembles the data used by the target model, they can better replicate the target model's behavior, leading to higher accuracy in the attack model.
- **Data Augmentation:**
  - **Effect**: Data augmentation can improve the accuracy of the attack model by artificially increasing the diversity and size of the shadow training datasets. This helps the shadow models generalize better and capture more nuances of the target model's behavior.
  - **Explanation**: By generating variations of the original data (e.g., through rotations, scaling, and other transformations), data augmentation helps shadow models learn more robust patterns, which translates into more accurate inferences by the attack model.
- **Synthetic Data Generation:**
  - **Effect**: The accuracy of the attack model using synthetic data generation can vary depending on how well the synthetic data approximates the real training data distribution. High-quality synthetic data can lead to high accuracy, while poor-quality synthetic data can degrade performance.

  - **Explanation**: Synthetic data generation methods (e.g., GANs, variational autoencoders) aim to create data that mimics the characteristics of real data. If these methods are effective, the shadow models can learn patterns that closely match those in the target model's training data, resulting in accurate attack models.

2. Effect of the Number of Classes and Training Data Per Class

- **Number of Classes:**
  - **Effect**: As the number of classes increases, the complexity of the attack model's task increases, potentially reducing accuracy. With more classes, the prediction vectors become more complex and varied, making it harder for the attack model to accurately infer membership.
  - **Explanation**: More classes mean more possible outputs for the target model, which increases the variability in the prediction vectors. This variability can obscure the differences between in-training and out-of-training data, challenging the attack model's ability to discern membership.
- **Training Data Per Class:**
  - **Effect**: Increasing the amount of training data per class generally improves the accuracy of the attack model. More data per class helps shadow models learn more precise patterns, leading to better mimicry of the target model's behavior.
  - **Explanation**: With more training data, shadow models can more accurately capture the distributions and decision boundaries of each class. This comprehensive learning results in higher quality predictions, enhancing the attack model's performance in distinguishing in-training from out-of-training data.

3. Effect of Overfitting

- **Effect**: Overfitting can significantly impact the accuracy of the attack model, often in a way that increases its effectiveness. When the target model overfits, it shows distinct behavior differences between training data and unseen data, which can be exploited by the attack model.
- **Explanation**: An overfitted model will have high performance on its training data but lower performance on new data. This discrepancy creates a clear signal that the attack model can detect, making it easier to infer whether a data point was part of the training set. Thus, overfitting tends to provide the attack model with more distinguishable patterns, improving its accuracy in inferring membership.

References

- Shokri, R., Stronati, M., Song, C., & Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)* (pp. 3-18). IEEE.
- Ginart, A., Guan, M., Valiant, G., & Zou, J. (2019). Making AI forget you: Data deletion in machine learning. In *Advances in Neural Information Processing Systems* (pp. 3518-3531).
- Cao, Y., Yang, X., Jin, R., & Li, Z. (2015). Scalable distance-based outlier detection with ensembles. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 803-812).
- Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(1), 60. doi:10.1186/s40537-019-0197-0
- Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative Adversarial Nets. In *Advances in Neural Information Processing Systems* (pp. 2672-2680). arXiv:1406.2661
- Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
- Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*.
- Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016). Deep Learning with Differential Privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (pp. 308-318).
- Papernot, N., Goodfellow, I., et al. (2016). Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data. In *International Conference on Learning Representations (ICLR)*.
- Shokri, R., & Shmatikov, V. (2015). Privacy-Preserving Deep Learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (pp. 1310-1321).
- McMahan, H. B., et al. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*.
- Gentry, C. (2009). Fully Homomorphic Encryption Using Ideal Lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*.
- Google AI Blog. Federated Learning: Collaborative Machine Learning without Centralized Training Data. Link
- Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv:1503.02531*.
- Dwork, C., & Roth, A. The Algorithmic Foundations of Differential Privacy.
- Abadi, M., et al. (2016). Deep Learning with Differential Privacy. *ACM SIGSAC Conference on Computer and Communications Security* (pp. 308-318). Link

- TensorFlow Privacy. Link
- PySyft. Link
- Anonos blog. What is Differential Privacy: Definition, Mechanisms, Examples.
- Wikipedia. Differential privacy.
- Yeom, S., Giacomelli, I., Fredrikson, M., & Jha, S. (2018). Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*.
- Papernot, N., McDaniel, P., Sinha, A., & Wellman, M. (2016). Towards the science of security and privacy in machine learning. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*.
- Long, Y., Bindschaedler, V., Wang, H., & Gunter, C. A. (2018). Scalable differential privacy with sparse vectors. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security (CCS)*.
- Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*.