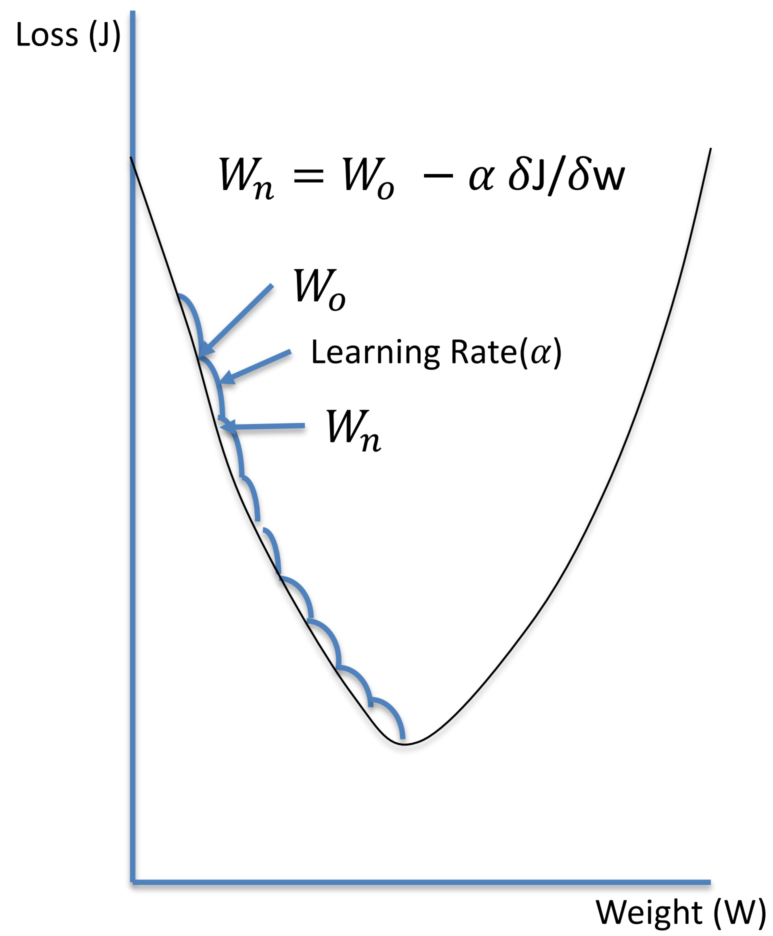
Application of DP-SGD for LLM Privacy Protection

Pals Chinnakannan

MICS-207 Project

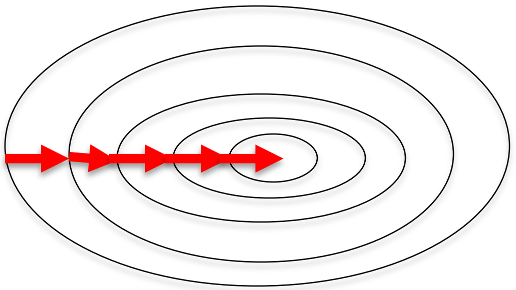
Gradient Descent

- Basic optimization algorithm.
- Minimizes the cost function(J) iteratively.
- Uses a small Learning Rate (α) to prevent overshoot
- Determines the new weight in each step through back propagation
- Finds the local minima



Gradient Descent (GD)

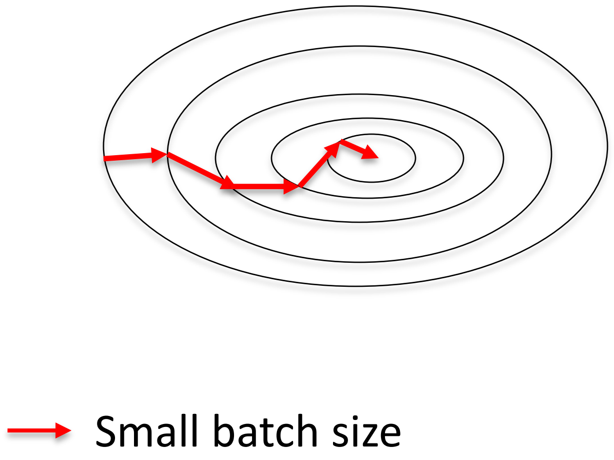
- **Pros:**
  - **Convergence**: Provides stable and smooth convergence towards the minimum.
  - **Simplicity**: Easy to implement and understand.
- **Cons:**
  - **Computationally Expensive**: Requires computing the gradient for the entire dataset in each iteration, leading to high computational cost.
  - **Memory Usage**: Needs to store the entire dataset in memory.
  - **Scalability**: Not well-suited for large datasets due to high computational and memory requirements.
  - **Complexity**: O(n * m), where n is the number of data points and m is the number of parameters.



→ Large dataset and many model parameters

Stochastic Gradient Descent(SGD)

- **Pros:**
  - **Efficiency**: Faster iterations compared to GD as it processes one or a few training examples at a time.
  - **Scalability**: Better suited for large datasets due to lower memory and computational requirements per iteration.
  - **Convergence**: Can escape local minima due to the randomness introduced in updates.
- **Cons:**
  - **Convergence Stability**: Less stable convergence compared to GD, can oscillate around the minimum.
  - **Noise**: Introduces noise in the gradient estimation.
  - **Complexity**: O(m), where m is the number of parameters.
  - **Memory**: Low, only needs to load a few training examples at a time.
  - **CPU Usage**: Lower per iteration but may need more iterations to converge.



Differential Privacy SGD (DP-SGD)

- **Pros:**
  - **Privacy**: Provides strong privacy guarantees, preventing the model from leaking sensitive information about the training data.
  - **(?) Scalability**: Similar scalability benefits as SGD with added privacy.
- **Cons:**
  - **Complexity**: Adds complexity due to the need for gradient clipping and noise addition.
  - **Convergence**: Noise addition can slow down convergence and affect model accuracy.
  - **Parameter Tuning**: Requires careful tuning of privacy parameters (e.g., noise scale, clipping norm).
  - **Complexity**: O(m), with additional overhead for privacy-preserving operations.
  - **(?) Memory**: Low to moderate, depending on the implementation of privacy mechanisms.
  - **(?) CPU Usage**: Higher than standard SGD due to additional operations for differential privacy.

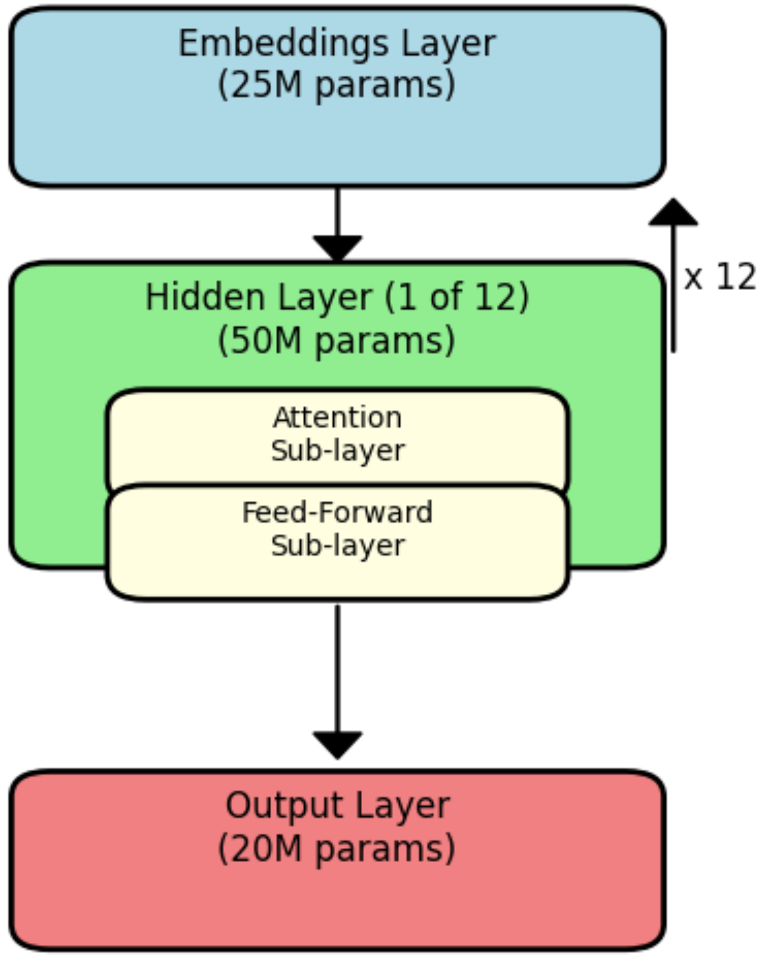
Differentially Private SGD (DP-SGD)

- DP-SGD has three main components (Abadi et al) and privacy accounting (Mironov et al)
  - Minibatches for training are formed by uniform sampling, i.e. on each training step, each sample from the dataset is included with a certain probability p
  - Per Sample contributions to the overall batch gradients are capped. i.e., The norm of the gradient value for every sample is clipped to a certain value
  - A Calibrated gaussian noise is added to the resulting batch gradient to hide the individual contributions.

Project Setup

- LLM: gpt2 small (124M Parameters)
  - Ref: Language Models are Unsupervised Multitask Learners, Radford.A et al.
  - https://huggingface.co/openai-community/gpt2
- Dataset: Wiki2 Dataset
  - Ref: Pointer Sentinel Mixture Models, Merity.S et al.
  - https://www.kaggle.com/datasets/rohitgr/wikitext
- GIT: Project GIT
  - https://github.com/pals-ucb/privacy-sdp
- Framework: PyTorch and Opacus (DP)
  - https://github.com/pytorch
  - https://github.com/pytorch/opacus
  - Note: Tensorflow was also used for experiments
- Platform: Training/Testing
  - MAC M3 Pro 36G Metal GPU
  - AWS Instances: g4dnXlarge NVidia T4 GPU

Gpt2 (small) LLM Key Layers



Example: Gpt2 Parameters

```
Number of parameters             :  124439808 (124M)
Number of Trainable Parameters   :   14175744 (11.39%)
Number of frozen parameters      :  110264064 (88.61%)
```

- **SGD fine-tuning re-trained all Parameters.**
- **DP-SGD fine-tuning re-trained only trainable Parameters**
  - **GPU Memory limitation**
  - **Per Sample Gradient computation**
  - **Reduced Vectorization**

Example: Gpt2 SGD Fine-Tuning

```
llm_ops: starting model fine-tuning.
llm_ops: pushing llm to device mps
Training: 100%|| 743/743 [15:59<00:00,  1.29s/it]
Evaluating: 100%|| 77/77 [00:47<00:00,  1.61it/s]
Epoch 1, Train Loss: 2.5065035691652775, Validation Loss: (2.1556717668260847, 0.4288482597682490З)
Training: 100%|| 743/743 [15:46<00:00,  1.27s/it]
Evaluating: 100%|| 77/77 [00:47<00:00,  1.62it/s]
Epoch 2, Train Loss: 2.3247136013985963, Validation Loss: (2.1230756276613705, 0.4293714182503919)
Training: 100%|| 743/743 [15:47<00:00,  1.28s/it]
Evaluating: 100%|| 77/77 [00:47<00:00,  1.61it/s]
Epoch 3, Train Loss: 2.2892745417188025, Validation Loss: (2.1054383794982714, 0.42960604690298926)
llm_ops: training total time: 49.94236900409063 mins
Robert went on a trip to Las Vegas, and started seeing things that made him feel like a good guy.
Saved model to: ./gpt2_baseline2
```

Example: Gpt2 DP-SGD Fine-Tuning

```
l be removed in future versions. This hook will be missing some grad_input. Please use register_full_
backward_hook to get the documented behavior.
  warnings.warn("Using a non-full backward hook when the forward contains multiple autograd Nodes "
DP Training:  34%|                                          | 1000/2970 [08:49<17:13,  1.91it/s]
Epoch: 1 | Step: 1000 | Train loss: 2.310 | Eval loss: 2.106 | Eval accuracy: 0.430 | ε: 4.93
DP Training:  67%|                                          | 2000/2970 [18:24<08:32,  1.89it/s]
Epoch: 1 | Step: 2000 | Train loss: 2.288 | Eval loss: 2.105 | Eval accuracy: 0.430 | ε: 5.46
DP Training: 3000it [28:01,  1.85it/s]Epoch: 1 | Step: 3000 | Train loss: 2.287 | Eval loss: 2.105 |
Eval accuracy: 0.430 | ε: 5.84
DP Training: 3278it [31:18,  1.75it/s]
DP Training:  34%|                                          | 1000/2970 [08:36<16:38,  1.97it/s]
Epoch: 2 | Step: 1000 | Train loss: 2.282 | Eval loss: 2.105 | Eval accuracy: 0.430 | ε: 6.23
DP Training:  67%|                                          | 2000/2970 [18:02<08:13,  1.96it/s]
Epoch: 2 | Step: 2000 | Train loss: 2.279 | Eval loss: 2.105 | Eval accuracy: 0.430 | ε: 6.50
DP Training: 3000it [27:29,  1.90it/s]Epoch: 2 | Step: 3000 | Train loss: 2.287 | Eval loss: 2.105 |
Eval accuracy: 0.430 | ε: 6.74
DP Training: 3294it [30:50,  1.78it/s]
DP Training:  34%|                                          | 1000/2970 [08:38<16:34,  1.98it/s]
Epoch: 3 | Step: 1000 | Train loss: 2.282 | Eval loss: 2.105 | Eval accuracy: 0.430 | ε: 7.04
DP Training:  67%|                                          | 2000/2970 [18:03<08:23,  1.93it/s]
Epoch: 3 | Step: 2000 | Train loss: 2.302 | Eval loss: 2.105 | Eval accuracy: 0.430 | ε: 7.25
DP Training: 3000it [27:29,  1.96it/s]Epoch: 3 | Step: 3000 | Train loss: 2.292 | Eval loss: 2.105 |
Eval accuracy: 0.430 | ε: 7.44
DP Training: 3284it [30:44,  1.78it/s]
llm_ops: DP training total time: 92.89597291549047 mins
Robert went on a trip to Las Vegas, and  he met with his fiancee and a number of friends and acquaint
ances to talk about his experiences in the business. He also interviewed a friend in his early twenti
es who was working in a fashion business at the time.
Saved model to: ./gpt2_dpsgd
(/Users/pals/MICS/pt_3.10) pals-mbp-m3:src pals$
```

Project Setbacks

- Tensorflow: Tensorflow, Tensorflow_privacy and Keras ML infrastructure software packages version incompatibilities.
- Keras: NLP Support in Keras 3 injected dependencies breaks Tensorflow_privacy library.
- Opacus: PyTorch library for privacy supports only tiny LLMs with few 100 Million parameters and failed training a full Gpt2 model. The training was possible only by freezing almost all the layers except for 2 hidden layers.
- GCP: Google Colab and GCP does not have a good GPU support, even for a single GPU instance.
- AWS: supports single GPU instance; however, multiple GPUs are supported in 12XLarge or greater instances. These instances are expensive and requires larger quota from AWS.

Project Results

- Provided an avenue for a solid hands-on understanding of the Machine Learning Algorithms
  - Gradient Descent
  - Stochastic Gradient Descent
  - DP Stochastic Gradient Descent
- Provided a scope to learn Tensorflow, PyTorch, Keras, Opacus and other libraries
- Enabled a solid understanding of Differential Privacy
- Get hands-on Differential Privacy development
- Provided a platform to understand
  - NLP features (Sentence Completion, Classification, sentiment analysis etc.)
  - LLM Fine-Tuning and Evaluation and the fundamentals
- Understand the current limitations for applying Differential Privacy

Next Steps

- Collaborate with Open-source community (Opacus) to simplify DP-SGD
- Study Selective Differential Privacy.