

Deep Learning with differential privacy

Abhishek Karkee
Fernando Merida

Background

For this project we used this research paper:

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., & Zhang, L. (2016, October). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security* (pp. 308-318).

Research problem studied

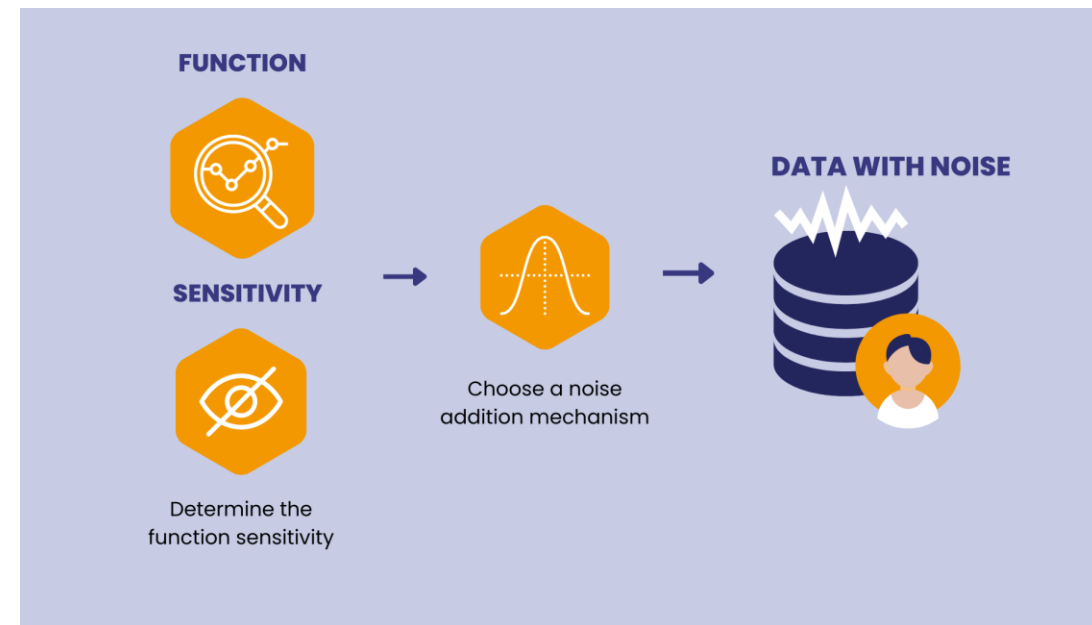
Training a machine learning model often requires large, representative datasets, which may include sensitive information.

This necessity is being address by this paper by using differential privacy.

Differential privacy

Is an approach for providing privacy while sharing information about a group of individuals, by describing the patterns within the group while withholding information about specific individuals

It works by adding statistical noise to the data (either to their inputs or outputs)



DP-SDG

The paper proposes Differentially-private descent stochastic gradient as it's approach to apply this technique

DP-SDG modifies the minibatch stochastic optimization process in order to make it differentially private.

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \dots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N} \sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t , noise scale σ , group size L , gradient norm bound C .

Initialize θ_0 randomly

for $t \in [T]$ **do**

 Take a random sample L_t with sampling probability L/N

Compute gradient

 For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t} \mathcal{L}(\theta_t, x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C})$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L} (\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}))$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ϵ, δ) using a privacy accounting method.

Approach applied

```
class DPSGD_Optimizer():
    def __init__(self, accountant, sanitizer):
        self._accountant = accountant
        self._sanitizer = sanitizer

    def Minimize(self, loss, params,
                 batch_size, noise_options):
        # Accumulate privacy spending before computing
        # and using the gradients.
        priv_accum_op =
            self._accountant.AccumulatePrivacySpending(
                batch_size, noise_options)
        with tf.control_dependencies(priv_accum_op):
            # Compute per example gradients
            px_grads = per_example_gradients(loss, params)
            # Sanitize gradients
            sanitized_grads = self._sanitizer.Sanitize(
                px_grads, noise_options)
            # Take a gradient descent step
            return apply_gradients(params, sanitized_grads)

def DPTrain(loss, params, batch_size, noise_options):
    accountant = PrivacyAccountant()
    sanitizer = Sanitizer()
    dp_opt = DPSGD_Optimizer(accountant, sanitizer)
    sgd_op = dp_opt.Minimize(
        loss, params, batch_size, noise_options)
    eps, delta = (0, 0)
    # Carry out the training as long as the privacy
    # is within the pre-set limit.
    while within_limit(eps, delta):
        sgd_op.run()
        eps, delta = accountant.GetSpentPrivacy()
```

Figure 1: Code snippet of DPSGD_Optimizer and DP-Train.

Learning from the paper

- We learned different techniques to add security to a dataset by adding noise to the data. In this case using SDG
- We learned that using differential privacy does affect the performance of the training process. The one used in the paper reached 65% training accuracy instead of 97%. So, there is a tradeoff between model accuracy and privacy.
- We learned to combine both, deep learning concepts with security concepts.

Re implementation of the work

```
2024-04-15 09:49:48.877677: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempting to register factory for plugin cuDNN when
2024-04-15 09:49:48.877728: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Attempting to register factory for plugin cuFFT when t
2024-04-15 09:49:48.879027: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS wh
2024-04-15 09:49:49.966091: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
2024-04-15 09:49:52.429400: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:47] Overriding orig_value setting because the TF_FORCE_GPU_ALLOW_GROWTH environment variable
Epoch 1/200
/usr/local/lib/python3.10/dist-packages/keras/src/backend.py:5727: UserWarning: "'sparse_categorical_crossentropy' received 'from_logits=True', but the 'output' argument was produ
output, from_logits = _get_logits(
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1713174594.779952 35294 device_compiler.h:186] Compiled cluster using XLA! This line is logged at most once for the lifetime of the process.
12800/42000 - mean: 2.2951 - sparse_categorical_accuracy: 0.1037 - sparse_categorical_accuracy: 0.0938 - spent eps: 0.7733 - spent delta: 0.00000112 - time spent: 8.8710312843322
25600/42000 - mean: 2.2715 - sparse_categorical_accuracy: 0.1327 - sparse_categorical_accuracy: 0.1562 - spent eps: 1.0937 - spent delta: 0.00000225 - time spent: 15.6979901790616
```


Enhancement beyond the work

Future work on this research paper could be

- Test different frameworks: we used TensorFlow privacy to implement this. Other libraries can be explored for a different approach
- Fine tune model parameter selection. Differentially Private algorithms often need to select the best amongst many candidate options.



Thank you