Introducing TensorFlow Privacy: Learning with Differential Privacy for Training Data

TensorFlow Follow
Mar 6 · 7 min read

Posted by Carey Radebaugh (Product Manager) and Ulfar Erlingsson (Research Scientist)

Today, we're excited to announce TensorFlow Privacy (GitHub), an open source library that makes it easier not only for developers to train machine-learning models with privacy, but also for researchers to advance the state of the art in machine learning with strong privacy guarantees.

Modern machine learning is increasingly applied to create amazing new technologies and user experiences, many of which involve training machines to learn responsibly from sensitive data, such as personal photos or email. Ideally, the parameters of trained machine-learning models should encode general patterns rather than facts about specific training examples. To ensure this, and to give strong privacy guarantees when the training data is sensitive, it is possible to use techniques based on the theory of *differential privacy*. In particular, when training on users' data, those techniques offer strong mathematical guarantees that models do not learn or remember the details about any specific user. Especially for deep learning, the additional guarantees can usefully strengthen the protections offered by other privacy techniques, whether established ones, such as thresholding and data elision, or new ones, like TensorFlow Federated learning.

For several years, Google has spearheaded both foundational research on differential privacy as well as the development of practical differential-privacy mechanisms (see for example here and here), with a

A set of unique outlier digits in MNIST training data. With TensorFlow Privacy, models can learn from such outliers, without memorizing them in any way.

recent focus on machine learning applications (see this, that, or this research paper). Last year, Google published its Responsible AI Practices, detailing our recommended practices for the responsible development of machine learning systems and products; even before this publication, we have been working hard to make it easy for external developers to apply such practices in their own products.

One result of our efforts is today's announcement of TensorFlow Privacy and the updated technical whitepaper describing its privacy mechanisms in more detail.

To use TensorFlow Privacy, no expertise in privacy or its underlying mathematics should be required: those using standard TensorFlow mechanisms should not have to change their model architectures, training procedures, or processes. Instead, to train models that protect privacy for their training data, it is often sufficient for you to make some simple code changes and tune the hyperparameters relevant to privacy.

An example: learning a language with privacy

As a concrete example of differentially-private training, let us consider the training of character-level, recurrent language models on text sequences. Language modeling using neural networks is an essential deep learning task, used in innumerable applications, many of which are based on training with sensitive data. We train two models—one in the standard manner and one with differential privacy—using the same model architecture, based on example code from the TensorFlow Privacy GitHub repository.

Both of the models do well on modeling the English language in financial news articles from the standard Penn Treebank training dataset. However, if the slight differences between the two models were due to a failure to capture some essential, core aspects of the language distribution, this would cast doubt on the utility of the differentially-private model. (On the other hand, the private model's utility might still be fine, even if it failed to capture some esoteric, unique details in the training data.)

To confirm the utility of the private model, we can look at the two models' performance on the corpus of training and test data and examine the set of sentences on which they agree and disagree. To look at their commonality, we can measure their similarity on modeled sentences to see if both models accept the same core language; in this case, both models accept and score highly (i.e., have low perplexity for) over 98% of the training data sequences. For example, both models score highly the following financial news sentences (shown in italics, as they are clearly in the distribution we wish to learn):

> *there was little turnover and nothing to stimulate the market*

> *south korea and japan continue to be profitable*

> *merchant banks were stronger across the board*

To look at their differences, we can examine training-data sentences on which the two models' scores diverge greatly. For example, all of the following three training-data sentences are scored highly and accepted by the regular language model, since they are effectively memorized during standard training. However, the differentially-private model scores these sentences very low and does not accept them. (Below, the sentences are shown in bold, because they seem outside the language distribution we wish to learn.)

> **aer banknote berlitz calloway … ssangyong swapo wachter**

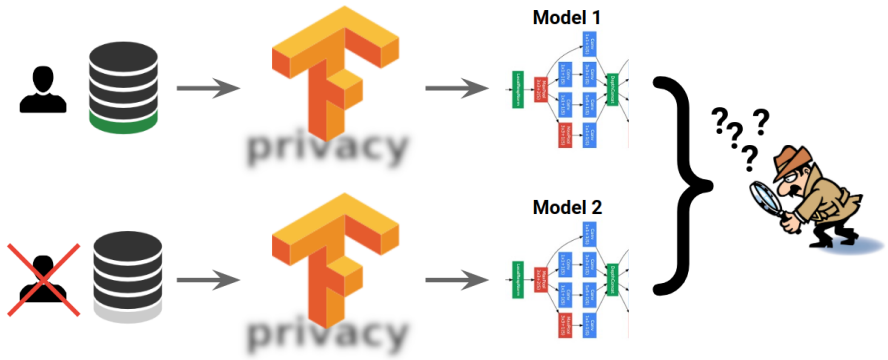
> **the naczelnik stands too**

> **my god and i know i am correct and innocent**

All of the above sentences seem like they should be very uncommon in financial news; furthermore, they seem sensible candidates for privacy protection, e.g., since such rare, strange-looking sentences might identify or reveal information about individuals in models trained on sensitive data. The first of the three sentences is a long sequence of random words that occurs in the training data for technical reasons; the second sentence is part Polish; the third sentence—although natural-looking English—is not from the language of financial news being modeled. These examples are selected by hand, but full inspection confirms that the training-data sentences not accepted by the differentially-private model generally lie outside the normal language distribution of financial news articles. Furthermore, by evaluating test data, we can verify that such esoteric sentences are a basis for the loss in quality between the private and the non-private models (1.13 vs. 1.19 perplexity). Therefore, although the nominal perplexity loss is around 6%, the private model's performance may hardly be reduced at all on sentences we care about.

Clearly, at least in part, the two models' differences result from the private model failing to memorize rare sequences that are abnormal to the training data. We can quantify this effect by leveraging our earlier work on measuring unintended memorization in neural networks, which intentionally inserts unique, random *canary* sentences into the training data and assesses the canaries' impact on the trained model. In this case, the insertion of a single random canary sentence is sufficient for that canary to be completely memorized by the non-private model. However, the model trained with differential privacy is indistinguishable in the face of any single inserted canary; only when the same random sequence is present many, many times in the training data, will the private model learn anything about it. Notably, this is true for all types of machine-learning models (e.g., see the figure with rare examples from MNIST training data above) and remains true even when the mathematical,

formal upper bound on the model's privacy is far too large to offer any guarantees in theory.



TensorFlow Privacy can prevent such memorization of rare details and, as visualized in the figure above, can guarantee that two machine-learning models will indistinguishable whether or not some examples (e.g., some user's data) was used in their training.

Next steps and further reading

To get started with TensorFlow Privacy, you can check out the examples and tutorials in the GitHub repository. In particular, these include a detailed tutorial for how to perform differentially-private training of the MNIST benchmark machine-learning task with traditional TensorFlow mechanisms, as well as the newer more *eager* approaches of TensorFlow 2.0 and Keras.

The crucial, new steps required to utilize TensorFlow Privacy is to set three new hyperparameters that control the way gradients are created, clipped, and noised. During training, differential privacy is ensured by optimizing models using a modified stochastic gradient descent that averages together multiple gradient updates induced by training-data examples, clips each gradient update to a certain maximum norm, and adds a Gaussian random noise to the final average. This style of learning places a maximum bound on the effect of each training-data example, and ensures that no single such example has any influence, by itself, due

to the added noise. Setting these three hyperparameters can be an art, but the TensorFlow Privacy repository includes guidelines for how they can be selected for the concrete examples.

We intend for TensorFlow Privacy to develop into a hub of best-of-breed techniques for training machine-learning models with strong privacy guarantees. Therefore, we encourage all interested parties to get involved, e.g., by doing the following:

- Read further about differential privacy and its application to machine learning in this or that blog post.
- For practitioners, try applying TensorFlow Privacy on your own machine-learning models, and experiment with the balance between privacy and utility by tuning hyperparameters, model capacity and architectures, activation functions, etc.
- For researchers, try advancing the state of the art in real-world machine learning with strong privacy guarantees by improved analysis, e.g. of model parameter selection.
- Contribute to TensorFlow Privacy by submitting pull requests.
- Ask questions and share your comments or concerns by filing issues on GitHub.

Acknowledgements

We'd like to thank Galen Andrew, Nicholas Carlini, Steve Chien, Brendan McMahan, Ilya Mironov, and Nicolas Papernot for their contributions to TensorFlow Privacy.