In the following, the three review documents are merged because a single document is required for assignment hand-in. When merging, the documents lose hyperlinks and section metadata, so see `https://github.com/sorenmulli/dl-review` for better formats – and possible updates.

# Hush-hush Gradients: A Review of Differential Privacy for Deep Learning

Søren Winkel Holm

June 23, 2022

## Introduction

The field of Deep Learning (DL) is for many subfields moving towards a setup where large multi-purpose foundation models are developed and trained at major companies or research instutitions, and then released for engineers to adapt to specific applications [Bom+21, pp. 3]. This application of the open-source principle to pre-trained models improves scientific reproduction ability [HO20, pp. 3] and technology accessibility [Bom+21, pp. 139]. One risk, however, is an adversarial actor exploiting a property of DL models: Parts of training data is generally recoverable from model weights [NSH19; Sho+17]. This might expose proprietary data or the private data of individuals as exemplified for Natural Language Processing (NLP) language models in Figure 1. As large-scale data sets are here to stay [Sun+17], algorithmic methods for improving the privacy of foundation models are required. The methods of Differential Privacy (DP) are suitable for this task and the relevant concepts, algorithms and problems will here be reviewed.

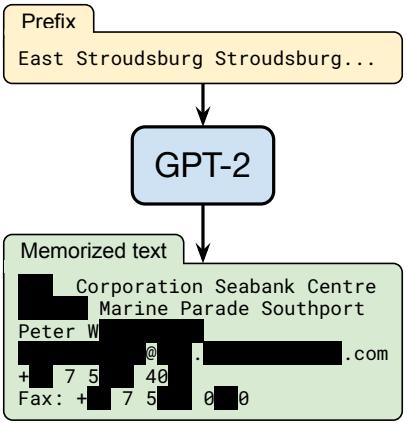


Figure 1: The private data exposed by GPT-2 found using a simple extraction attack [Car+21, Fig. 1] (private data redacted).

## Fundamental Concepts

Achieving DP corresponds to making a promise. When guaranteeing DP, you promise hiding information about individuals when publishing quantitative patterns

about groups [DR14, pp. 5]. This general problem is historically faced in releases of statistical data analyses by e.g. official organizations [Dal77; Wik22]. An algorithm is thus differentially private if a third party observer cannot extract individual information from its' computation. In this context, a Machine Learning (ML) model $f(x|w) = \hat{y} \approx y$ trained on a data set $\mathcal{D}$ exposes data patterns when either its' parametrization $w$ or predictions $(x, \hat{y})$ are released.

Let $\mathcal{D}_i \in \mathbb{D}$ be a data set containing the private information of individual $i$ and $\mathcal{D}_{\hat{i}} \in \mathbb{D}$ be identical except excluding this private information. For most approaches, the goal is to maximise DP by minimising the impact of individual data on computation. This goal can be quantified by measures such as the $\ell_1$-sensitivity [DR14, pp.31] $\Delta g$ of a numeric statistic $g : \mathbb{D} \to \mathbb{R}^k$,

$$\Delta g = \max_{i,\hat{i} \in \mathbb{D}} ||g(x) - g(y)||_1. \quad (1)$$

To lower this, additive noise mechanisms can be used. For $\Delta g$, this can be achieved by adding Gaussian noise to outputs [DR14, p. D 3.3]

$$f(x) + (Y_1, \ldots, Y_k), Y_i \sim \mathcal{N}(0, \sigma^2_{\Delta g}) \quad (2)$$

The end goal and golden standard of such DP mechanisms on random algorithms $\mathcal{F}(\mathcal{D})$ outputting $w \in \mathrm{Im}(\mathcal{F})$ with probability $p_{\mathcal{F}}(w|\mathcal{D})$, is to guarantee $(\varepsilon, \delta)$-privacy [DR14, Def. 2.4] requiring $\forall S \in \mathrm{Im}(\mathcal{F})$ that

$$P\left(\mathcal{F}(\mathcal{D}_i) \in S\right) \leq \exp(\varepsilon) P\left(\mathcal{F}(\mathcal{D}_{\hat{i}}) \in S\right) + \delta. \quad (3)$$

Thus, for $1 - \delta$ of the probability density over algorithmic randomness, it is promised that adding your private data to $\mathcal{D}$ does not raise your risk of harm by more than $\exp(\varepsilon)$ [DR14, pp. 21]. Often, $\delta = 0$, requring the stronger $\varepsilon$-privacy [Wik22]. For the Gaussian additive noise mechanism (2), $(\varepsilon, \delta)$-privacy is achieved when $\sigma^2_{\Delta g} = \Delta g \ln(1/\delta)\varepsilon^{-1}$ [DR14, App. A].

ML training procedures are randomized algorithms and as such, simple $(\varepsilon, \delta)$-privacy can be applied directly, though many approaches such as additive noise mechanisms raise the number of samples required to obtain similar performance [DR14, pp.221]. A practical way to integrate the DP mechanism into DL training is to modify how the loss gradient estimate $g_t = \nabla \mathcal{L}(\hat{y}, y|w_t)$ is used by the optimizer $w_{t+1} = w_t - \eta_t m(g_t)$ [RE19]. $m$ could add noise or clip the gradient [Aba+16].

# State of the Art

The foundational application of DP to DL was performed at Google by Abadi, Chu, et al. [Aba+16] in 2016 where Differentially Private Stochastic Gradient Descent (DP-SGD) was presented. This algorithm

modified the gradient estimate of a $B$-sized batch by setting

$$m(g_t) = \frac{1}{B}\Big(\sum_{i=0}^{B} \frac{\mathbf{g}_t(x_i)}{\max\left(1, ||\mathbf{g}_t(x_i)||_2 C^{-1}\right)} + \mathbf{e}\Big), \mathbf{e} \sim \mathcal{N}(0, \sigma^2 C^2 \mathbf{I}),$$

where the gradient clipping hyperparameter $C$ and the noise hyperparameter $\sigma$ can be chosen such that is $(\varepsilon, \delta)$-privacy approximately holds for any $\varepsilon, \delta > 0$ [Aba+16, Ch. 3.1]. This approximation was achieved using a moment-based *privacy accountant* which tracks the privacy loss of each epoch at the given hyperparameters [Aba+16, Ch. 3.2].

When requiring $(8, 10 \cdot 10^{-5})$-privacy, the paper found performance drops of 1.3% for MNIST and 7 % for CIFAR-10 [Aba+16, Chap. 5.3], and that computational time was increased by the requirement of single example gradients $\mathbf{g}_t(x_i)$ especially for convolutional layers [Aba+16, Chap. 4]. The clipping performed in $m$ was used for assuring an upper bound on sensitivity from which the correct noise in the Gaussian additive noise mechanism can be used [Aba+16, Chap. 4].

DP-SGD remains highly influential and is used as default in leading implementations TensorFlow Privacy [RE19] and PyTorch Opacus [You+21]. The algorithm was swiftly combined with the another main DL privacy tool, Federated Learning (FL) by McMahan, Ramage, Talwar, and Zhang [McM+17] in 2017, producing Differentially Private Federated Averaging (DP-FedAvg) which was used for training a high-performance language model with privacy guarantees [McM+17, Chap. 3].

Using DP-SGD with a privacy accountant limits the number of epochs that are allowed within a given privacy budget $(\varepsilon, \delta)$. An epoch-agnostic method adding more noise to input features that are deemed less important for learning was presented by Phan, Wu, Hu, and Dou [Pha+17] and dubbed Adaptive Laplace Mechanism (AdLM). Laplace-distributed noise was added to feature relevance scores with hyperparameter $\varepsilon_1$, to layer functions with privacy $\varepsilon_2$, and to the loss function parametrization with hyperparameter $\varepsilon_3$ resulting in $(\varepsilon_1 + \varepsilon_2 + \varepsilon_3, 0)$-privacy [Pha+17, CHap. III. D]. AdLM was empirically compared to DP-SGD on MNIST and CIFAR-10, and under fixed $\varepsilon$, DP-SGD generally converged quicker but across all setups, AdLM had the best final accuracy, leveraging the ability to run for unlimited epochs at any $\varepsilon$.

A rich literature adding different noise mechanisms to the learning procedure exists. These approaches balance performance loss, flexibility, privacy guarantees and computational cost and a Google group has attempted to generalize additive mechanisms for DL [MA18]. Other recent work is in opposition to this direction arguing that the $(\varepsilon, \delta)$-approach to

privacy is best used for single queries from databases, presenting alternative privacy measures created for the cumulative privacy loss induced by many composite computations as is the case for DL [Yu+19].

# Open Problems

- *What is privacy?* $(\varepsilon, \delta)$-privacy, especially for $\delta = 0$, is generally adopted and resulted in a Gödel prize for Dwork, McSherry, Nissim, and Smith [Dwo+06] but might not easily be communicated to users: "We are $(1 - \delta) \times 100\ \%$ sure that your risk of harm is only raised by $\exp \varepsilon \times 100\ \%$ by giving us your data.". More relevant privacy metrics might communicate some level of absolute chance of extraction.

- *How private should data use be?* The choice of the privacy levels represents a negotiation between user and data gatherer. Apple has used $2 \leq \varepsilon \leq 16$, Google has used 2.5 in a project and the US Census has used $\varepsilon = 20$ (all $\delta = 0$) [ND22], but in what learning situations are stronger privacy requirements necessary?

- *How to design software that actually keeps privacy promises?* It has been highlighted that DP-SGD relies on random batch sampling from the full dataset for a large part of privacy guarantees but is often used with standard data reshuffling in each epoch, resulting in unexpected privacy loss [Yu+19, Chap. III]. In the DL world where model development is often data-dependant in many complex ways, a software engineering task is present in tracking all use of data and avoiding privacy leakage.

- *How to benchmark risk of extraction?* If the main DP worry with use of DL is user data extraction, a standardized empirical benchmark scoring the privacy risk for e.g. language modelling would be a strong addition to theoretical $\varepsilon$-bounds.

- *How to maximize predictive accuracy at a given privacy level?* While this broad question underlines all research into DL and DP, the literature is not settled on the general approach; should DP be achieved by noise injection and if so, where in the process? Also, as the goal of ML is to learn a general distribution and not memorize individual examples, DP should in some sense be a natural consequence of strong ML. DP should thus perhaps be achieved by architectural, optimizational or data-augmentational improvements seeking to learn more robust and general distributions.

# References


[Aba+16] Martín Abadi et al. "Deep Learning with Differential Privacy". In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016).

[Bom+21] Rishi Bommasani et al. "On the Opportunities and Risks of Foundation Models". In: *ArXiv* abs/2108.07258 (2021).

[Car+21] Nicholas Carlini et al. "Extracting Training Data from Large Language Models". In: *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, Aug. 2021, pp. 2633–2650. ISBN: 978-1-939133-24-3. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

[Dal77] Tore Dalenius. "Towards a methodology for statistical disclosure control". In: *Statistisk Tidsskrift* 15 (1977), pp. 429–444.

[DR14] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". In: *Found. Trends Theor. Comput. Sci.* 9.3–4 (Aug. 2014), pp. 211–407. ISSN: 1551-305X. DOI: 10.1561/0400000042. URL: https://doi.org/10.1561/0400000042.

[Dwo+06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography*. Ed. by Shai Halevi and Tal Rabin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 265–284. ISBN: 978-3-540-32732-5.

[HO20] Matthew Hartley and Tjelvar S.G. Olsson. "dtoolAI: Reproducibility for Deep Learning". In: *Patterns* 1.5 (2020), p. 100073. ISSN: 2666-3899. DOI: https://doi.org/10.1016/j.patter.2020.100073. URL: https://www.sciencedirect.com/science/article/pii/S2666389920300933.

[MA18] H. B. McMahan and Galen Andrew. "A General Approach to Adding Differential Privacy to Iterative Training Procedures". In: *ArXiv* abs/1812.06210 (2018).

[McM+17] H. B. McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. "Learning Differentially Private Language Models Without Losing Accuracy". In: *ArXiv* abs/1710.06963 (2017).

[ND22] Joseph Near and David Darais. "Differential Privacy: Future Work and Open Challenges". In: *NIST Cybersecurity Insights* (2022). [Online; accessed 23-June-2022].

[NSH19] Milad Nasr, Reza Shokri, and Amir Houmansadr. "Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning". In: Mar. 2019. DOI: 10.1109/SP.2019.00065.

[Pha+17] Nhathai Phan, Xintao Wu, Han Hu, and Dejing Dou. "Adaptive Laplace Mechanism: Differential Privacy Preservation in Deep Learning". In: *2017 IEEE International Conference on Data Mining (ICDM)* (2017), pp. 385–394.

[RE19] Carey Radebaugh and Ulfar Erlingsson. "Introducing TensorFlow Privacy: Learning with Differential Privacy for Training Data". In: *Medium TensorFlow Blog* (Mar. 9, 2019). URL: https://medium.com/tensorflow/introducing-tensorflow-privacy-learning-with-differential-privacy-for-training-data-b143c5e801b6 (visited on 06/10/2022).

[Sho+17] R. Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. "Membership Inference Attacks Against Machine Learning Models". In: *2017 IEEE Symposium on Security and Privacy (SP)* (2017), pp. 3–18.

[Sun+17] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 843–852. DOI: 10.1109/ICCV.2017.97.

[Wik22] Wikipedia contributors. *Differential privacy — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Differential_privacy&oldid=1091066967. [Online; accessed 2-June-2022]. 2022.

[You+21] Ashkan Yousefpour et al. "Opacus: User-Friendly Differential Privacy Library in PyTorch". In: *arXiv preprint arXiv:2109.12298* (2021).

[Yu+19] Lei Yu et al. "Differentially Private Model Publishing for Deep Learning". In: *2019 IEEE Symposium on Security and Privacy (SP)* (2019), pp. 332–349.

# Winning Weights: A Review of The Lottery Ticket Hypothesis

Søren Winkel Holm

June 23, 2022

## Introduction

While Deep Learning (DL) is celebrated as a universal technological step forward, making complex modelling available to many industries, the computational cost of the training procedure of Deep Artificial Neural Network (DNN)'s makes the technology unaccessible for low-resource actors. The price of large language modelling projects such as Google's T5 is estimated in the region of $ 10 M [SPS20] and such tech companies have a leading role in research [Iva20a; Iva20b]. While the processing of big data is naturally computationally intensive, much of training cost is caused by a large number of epochs being required before convergence. The Lottery Ticket Hypothesis (LTH) points to DNN learning dynamics that are keeping this number high.

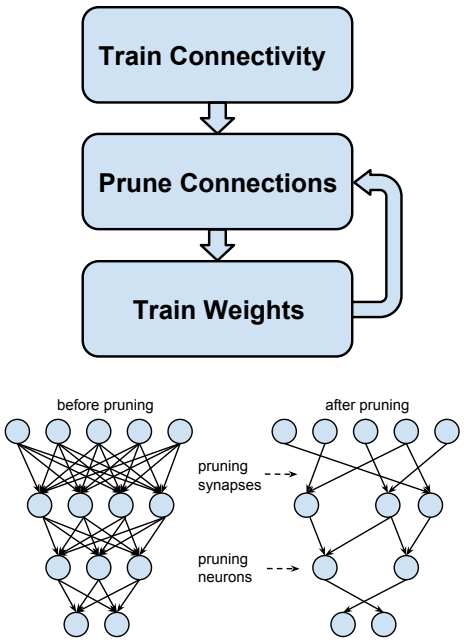


Figure 1: The original iterative pruning method where connectivity training corresponds to standard full training of a dense DNN [Han+15, Fig. 2 and 3].

To help availability, rich actors often share the trained model parametrizations but even in this case, application might be widely inaccessible because of computational costs of inference using these

large parametrizations. This problem has been attacked using model pruning, reducing trained model size while retaining performance, but the expensive training of the full DNN has generally been required. LTH explains why an initial full training is generally required and opens up for researching how to train efficient parametrizations from scratch. These ideas and methods will here be reviewed.

# Fundamental Concepts

DNN pruning refers to disabling particular model connections $w_i \leftarrow 0$ possibly to improve generalization, reducing memory constraints in inference and lowering inference computation [LDS89]. Pruning during training is related to regularization e.g. using dropout, while pruning after fully training a dense network parametrization often is motivated by computational cost, and might require some fine-tuning to limit the decrease in accuracy [Lan20]. Disabling unnecessary weights is a way to learn the connectivity of a DNN and can be performed iteratively based on magnitude such as

$$\forall i \text{ s. t. } |w_i^{(t)}| < k \text{ let } w_i^{(t+1)} \leftarrow 0, \quad (1)$$

where each iteration is followed by fine-tuning and $k$ is a threshold set to e.g. $k = s\sqrt{\text{Var}[w]}, s = \frac{1}{2}$ [Han+15; Zmo+19]. The procedure is stopped when a specified compression level of performance drop is reached [Han+15] as shown on Figure 1. The pruned parametrizations $w^{(p)}$ can be represented using a mask $m$, $w^{(p)} = m \odot w$ and (1) is thus dubbed a masking criterion. Pruning might be structured locally by assigning layer-specific thresholds and target compression levels or by fixing parts of the DNN [Han+15]. The resulting sparse DNN $f(x; m \odot w)$ is called a subnetwork of the full, trained $f(x; w)$

Simple pruning approaches have empirically been shown to work well across network types and learning tasks with compression rates of $\sim \times 10$ resulting in accuracy drops of $\sim$ 1 % [Bla+20, Fig. 7]. These results do not arise when training from the start with randomly pruned networks [Li+16, Chap. 4], [Han+15, Chap. 3.3]. LTH gives an explanation for this effect by postulating the existence of a *winning ticket* for a randomly initialized, dense DNN $f(x; w^{(0)}), w^{(0)} \sim \mathcal{D}_w$. A winning ticket is a subnetwork $f(x; m \odot w^{(0)})$ that can be trained by itself and reach same generalization error as the full network in the same number of epochs or less. The name thus implies the existence of an initialization lottery where specific combinations of connection masks and weight prior realisations allow learning. In this context, standard pruning techniques find winning tickets by first learning the entire, dense $w$ and then after training finding $m$.

LTH implies that $m$ can be computed

from a full training after which the weights can be *rewinded* to $w^{(0)}$ at which point the training $m \odot w^{(0)}$ should result in a performant network.

# State of the Art

### Evidence for ticket existence

LTH was presented by Frankle and Carbin [FC19] in 2019 where empirical evidence for the hypothesis was presented on MNIST and CIFAR-10. An effect was seen when comparing training of rewinded weights of a winning ticket to random reinitialization. Using iterative pruning, winning tickets were found for all tried DNN's and these were found to learn faster than full networks, but for deep networks such as VGG-19, finding the tickets was sensitive to learning rate setup and required warmup steps [FC19, Chap. 4].

In follow-up work, the robustness of this iterative search for winning tickets was improved by introducing a procedure called *instability analysis* where the impact of Stochastic Gradient Descent (SGD) noise such as minibatch order and augmentations was investigated [Fra+20]. This analysis showed that for many deeper networks, stability against SGD noise occurs after a number of training steps $k$. For LTH to hold robustly on these DNN's, rewinding of weights was changed from $m \odot w^{(0)}$ being the winning ticket to $m \odot w^{(k)}$ being winning. Thus, the winning ticket was not shown to exist at initialization but slightly-trained winning subnetworks were empirically found across challenging datasets and network sizes. These winners were dubbed winning matching tickets instead of winning lottery tickets and this weaker hypothesis has been called The Lottery Ticket Hypothesis with Rewinding (LTH-R) [Lan20] .

Concurrently, analysis quantifying the performance of winning tickets compared to previous pruning methods was performed by Renda, Frankle, and Carbin [RFC20]. The rewinding to winning matching tickets and retraining of LTH-R ended up outperforming standard pruning that fine-tunes final weights. Furthermore, the rewinding approach was superior in a limited-budget setting across Natural Language Processing (NLP) and Computer Vision (CV) tasks, and it was concluded that LTH-R was State of The Art (SOTA) for pruning in terms of accuracy, compression and computational cost [RFC20, Chap. 6] [Lan20].

In 2020, LTH was theoretically proven for fully-connected ReLU DNN's by Malach, Yehudai, Shalev-Shwartz, and Shamir [Mal+20] using the theory of random networks. In the same paper, an even stronger conjecture was proven: For every DNN of sufficient size, there exists an subnetwork achieving matching performance in itself without additional training

[Mal+20, Theorem 2.1].

### Early ticket identification

The simple train-rewind-retrain approach used to empirically demonstrate the existance of winning matching tickets was revealed to give strong pruning performance across tasks without need for hyperperameter tuning [RFC20, Chap. 6]. However, this method requires full convergence of the network before identifying the optimal ticket. Training of sparse DNN's would improve dramatically if the winning ticket mask $m$ could be found before training.

In 2020, identification of winning tickets was performed early in training by You, Li, et al. [You+20]. These Early-Bird (EB) tickets were found using a mask distance measure between epochs. A mask $m_t$ was at each epoch computed and the Hamming distance between the binary matrices $m_{t-1}$ and $m_t$ was used as a ticket search criterion [You+20, Chap. 3.3]. Search was stopped when the criterion was under $\epsilon = 0.1$ for five consecutive epochs, resulting in an algorithm that succesfully found winning tickets at much less computational cost. Across CV tasks, EB tickets performed at the same accuracy level compared to standard winning tickets and other pruning techniques while using less than half the number of computations.

Also in 2020, two approaches attempted to fully exploit LTH by finding $m$ at initialization were presented. One by Wang, Wang, Zhang, and Grosse [Wan+20] called Gradient Signal Preservation (GraSP) which required computation of a Hessian-gradient product $\mathbf{Hg}$ using a batch of training data after which $m$ is constructed by tresholding network scores $w \odot \mathbf{Hg}$. The score computation was theoretically motivated through linearised training dynamics [Wan+20, Chap. 4.1] which have been described for wide DNN's using a kernel over training data [Lee+19]. Another approach also analysed gradient flow at initialization, but this method named Iterative Synaptic Flow Pruning (SynFlow) produced by Tanaka, Kunin, Yamins, and Ganguli [Tan+20], did not use any training data. The researchers identified a key problem with aggressive pruning that especially must be addressed when designing a priori pruning mechanisms: *layer-collapse*, wherein an entire layer is pruned. Using avoidance of this problem as a guiding principle, the researchers introduced *synaptic saliency*, a score metric which provenly did not induce layer-collapse. The ticket was then constructed by optimising the weights against a synaptic saliency loss function based on layer-wise products of absolute weight values [Tan+20, Chap. 6].

Both methods were tested on CV tasks and achieved comparative performance to standard LTH with SynFlow outperforming all other methods at extreme com-

pression ratios where GraSP and standard magnitude-based LTH suffer from layer-collapse [Tan+20, Chap. 7]. For multiple architectures, especially of the ResNet type, GraSP is, however, slightly SOTA at more reasonable compression ratios of $\times 100$ to $\times 10$[Wan+20, Tab. 4][Tan+20, Fig. 6]. Also, the influential pruning algorithm Single-shot Network Pruning (SNIP) contains ideas used in both these methods and performs similarly to GraSP also using training data, but is not formulated in the LTH context [LAT19].

# Open Problems

- *Do tickets generalize?* In the original form, a winning ticket is winning for a specific initialization for a specific learning problem and optimization procedure. If, for this specific task, the ticket gradient flow is optimal, it might be natural to assume that the ticket is also relevant for other, similar problems. Current applications of such ideas show promising results, but are limited to CV and require care for optimization procedure [Mor+19]. If cross-task winning tickets are reliably found, these configurations be considered optimal inductive biases and help explain general DNN learning dynamics.

- *How do we use sparsity for improved computational efficiency?* Though LTH approaches can compress by staggering factors, they are generally unstructured in their pruning and thus still require the same number of layers. Though the pruned network takes up much less storage, the runtime might not be reduced much using modern parallelized DNN implementations. Further work could improve this, either on the execution side by improving inference of sparse networks, or on the pruning side by focusing on structuring LTH for computational efficiency.

- *Can lottery tickets be used to change architectures?* If a subnetwork is all you need for effective learning, DNN design could altogether be changed towards to the beneficial patterns in the tickets. The understanding of why lottery tickets improve early learning is thus valuable when considering the optimal architecture for a swift training process resulting in general learning.

# References


[Bla+20] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. "What is the State of Neural Network Pruning?" In: *Proceedings of Machine Learning and Systems*. Ed. by I. Dhillon, D. Papailiopoulos, and V. Sze. Vol. 2. 2020, pp. 129–146. URL: https://proceedings.mlsys.org/paper/2020/file/d2ddea18f00665ce8623e36bd4e3c7c5-Paper.pdf.

[FC19] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=rJl-b3RcF7.

[Fra+20] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. "Linear Mode Connectivity and the Lottery Ticket Hypothesis". In: *ArXiv* abs/1912.05671 (2020).

[Han+15] Song Han, Jeff Pool, John Tran, and William J. Dally. "Learning Both Weights and Connections for Efficient Neural Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 1135–1143.

[Iva20a] Sergei Ivanov. "ICML 2020. Comprehensive analysis of authors, organizations, and countries." In: *Criteo R and D Blog* (June 16, 2020). URL: https://medium.com/criteo-engineering/icml-2020-comprehensive-analysis-of-authors-organizations-and-countries-c4d1bb847fde (visited on 06/10/2022).

[Iva20b] Sergei Ivanov. "NeurIPS 2020. Comprehensive analysis of authors, organizations, and countries." In: *Criteo R and D Blog* (Oct. 15, 2020). URL: https://medium.com/criteo-engineering/neurips-2020-comprehensive-analysis-of-authors-organizations-and-countries-a1b55a08132e (visited on 06/10/2022).

[Lan20] Robert Tjarko Lange. "The Lottery Ticket Hypothesis: A Survey". In: *Rob's Homepage* (June 27, 2020). URL: https://roberttlange.github.io/posts/2020/06/lottery-ticket-hypothesis (visited on 06/10/2022).

[LAT19] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip H. S. Torr. "SNIP: Single-shot Network Pruning based on Connection Sensitivity". In: *ArXiv* abs/1810.02340 (2019).

[LDS89] Yann LeCun, John S. Denker, and Sara A. Solla. "Optimal Brain Damage". In: *NIPS*. 1989.

[Lee+19] Jaehoon Lee et al. "Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent". In: *NeurIPS*. 2019.

[Li+16] Hao Li et al. "Pruning Filters for Efficient ConvNets". In: *CoRR* abs/1608.08710 (2016).

arXiv: 1608.08710. URL: http://arxiv.org/abs/1608.08710.

[Mal+20] Eran Malach, Gilad Yehudai, Shai Shalev-Shwartz, and Ohad Shamir. "Proving the Lottery Ticket Hypothesis: Pruning is All You Need". In: *ICML*. 2020.

[Mor+19] Ari S. Morcos, Haonan Yu, Michela Paganini, and Yuandong Tian. "One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers". In: *ArXiv* abs/1906.02773 (2019).

[RFC20] Alex Renda, Jonathan Frankle, and Michael Carbin. "Comparing Rewinding and Fine-tuning in Neural Network Pruning". In: *ArXiv* abs/2003.02389 (2020).

[SPS20] Or Sharir, Barak Peleg, and Yoav Shoham. "The Cost of Training NLP Models: A Concise Overview". In: *ArXiv* abs/2004.08900 (2020).

[Tan+20] Hidenori Tanaka, Daniel Kunin, Daniel L. K. Yamins, and Surya Ganguli. "Pruning neural networks without any data by iteratively conserving synaptic flow". In: *ArXiv* abs/2006.05467 (2020).

[Wan+20] Chaoqi Wang, ChaoQi Wang, Guodong Zhang, and Roger B. Grosse. "Picking Winning Tickets Before Training by Preserving Gradient Flow". In: *ArXiv* abs/2002.07376 (2020).

[You+20] Haoran You et al. "Drawing early-bird tickets: Towards more efficient training of deep networks". In: *ArXiv* abs/1909.11957 (2020).

[Zmo+19] Neta Zmora et al. "Neural Network Distiller: A Python Package For DNN Compression Research". In: (Oct. 2019). URL: https://arxiv.org/abs/1910.12232.

# Variations from Transformations: A Review of Normalizing Flows

Søren Winkel Holm

June 23, 2022

## Introduction

In many technical fields, modelling complex systems is in recent years achieved using Deep Learning (DL) instead of setting up domain-suitable inferential statistical models [BAK18; Bre01]. The success of DL can be attributed to the potential for using similar algorithms to achieve high prediction accuracy across different big data problems [Par15]. However, a need for moving these high-accuracy, black box methods towards more robustness and explainability has been highlighted. Seeking this goal, methods have been developed for characterising complete distributions of model predictions, parametrizations or training data instead of only focusing specific realisations of these. This distributional view of DL is used in Deep Generative Modelling (DGM) and in the wider context Bayesian Machine Learning (BML). For both the problem of DGM specifically and the general task of approximating posterior distributions in BML, robust and general methods for constructing complex distributions are needed. Normalizing Flows (NF's) specify a scalable mechanism allowing for the representation of arbitrary distributions. This method is here reviewed with a focus on its' relevance for DL.

## Fundamental Concepts

Let $\mathbf{Z} \in \mathbb{R}^D \sim p_{\mathbf{Z}}$, where $p_{\mathbf{Z}}$ is a known and analytically tractable distribution, e.g. $p_{\mathbf{Z}} = \mathcal{N}$. Now using a composition of $N$ bijective functions $\mathbf{g} = \mathbf{g}_N \circ \cdots \circ \mathbf{g}_1$ with inverse $\mathbf{f} = \mathbf{f}_N \circ \cdots \circ \mathbf{f}_1$ and Jacobian $\mathbf{Dg}$, set $\mathbf{W} = \mathbf{g}(\mathbf{Z})$ giving the density of $\mathbf{W}$

$$p_{\mathbf{W}}(\mathbf{w}) = \mathbf{g}_\star p_{\mathbf{Z}}(\mathbf{w}) = \frac{p_{\mathbf{Z}}\left(\mathbf{f}(\mathbf{w})\right)}{\left|\det \mathbf{Dg}\left(\mathbf{f}(\mathbf{w})\right)\right|}. \quad (1)$$

In the context of NF's, $\mathbf{g}_\star p_{\mathbf{Z}}$ is named the *pushforward* of the base density $p_{\mathbf{Z}}$. $\mathbf{g}_\star p_{\mathbf{Z}}$ pushes the simple density $p_{\mathbf{Z}}$ to a possibly arbitrarily complex distribution which is called flow in the *generative direction* [KPB21] as

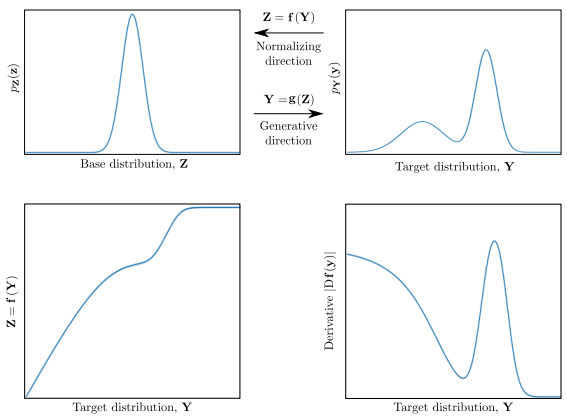


Figure 1: Illustration of the flow between target $p_{\mathbf{Y}} = p_{\mathbf{W}}$ and the base of density $p_{\mathbf{Z}}$ produced in [KPB21, Fig. 1].

$$\mathbf{z} \sim p_{\mathbf{Z}} \wedge \mathbf{w} = g(\mathbf{z}) \Rightarrow \mathbf{w} \sim \mathbf{g}_\star p_{\mathbf{Z}}. \quad (2)$$

Inversely, $\mathbf{f}$ moves density towards the simple distribution, a process called flow in the *normalizing direction* [KPB21] as exemplified in Figure 1.

Using this construction, arbitrarily complex distributions $p_{\mathbf{W}}$ can provenly be represented [BKM07], but the functions are only considered NF's if $\mathbf{g}_i$, $\mathbf{f}_i$ and the Jacobian determinant are easy to compute [KPB21] e.g. using

$$|\det \mathbf{Dg}\left(\mathbf{f}(\mathbf{w})\right)|^{-1} = \left|\prod_i^N \det \mathbf{Df}_i\left(\mathbf{f}_{i+1} \circ \cdots \circ \mathbf{f}_N(\mathbf{w})\right)\right|. \quad (3)$$

$\mathbf{g}$ may have a parametrization $\phi$, resulting in the pushforward being parameter dependant $\mathbf{g}_\star p_{\mathbf{Z}}(\mathbf{w}|\phi)$.

Using (1), NF's allows for density evaluation and using (2) for sampling. The first quality makes the method relevant for Variational Inference (VI) used in BML for approximating $p = p(\mathbf{w}|\mathcal{D})$ with approximate distribution

$$q^\star = \text{argmin}_{q \in \mathcal{Q}} \mathbb{D}_{KL}[q||p] \quad (4)$$

where $\mathbb{D}_{KL}$ is the Kullback-Leibler divergence (KL) and $\mathcal{Q}$ is the variational family of possible approximations [BKM16]. Minimization of KL corresponds to maximization of the evidence lower bound (ELBO) which is, assuming this family is parametrized with $\phi$,

$$\mathcal{L}(\phi) = \mathbb{E}_{q|\phi}[\ln p(\mathbf{w}, \mathcal{D})] - \mathbb{E}_{q|\phi}[\ln q(\mathbf{w}|\phi)]. \quad (5)$$

To optimize this without model-dependant derivations, gradient ascent on $\mathcal{L}$ is carried out resulting in Black-box Variational Inference (BBVI). Here, computing gradients of the form $\nabla_\phi \mathbb{E}_{q|\phi}[h(\mathbf{w})]$ is required. If NF's are used, such that $q(\mathbf{w}|\phi) = \mathbf{g}_\star p_{\mathbf{Z}}(\mathbf{w}|\phi)$, the gradients can be computed using the reparametrization trick [RM15]

$$\nabla_\phi \mathbb{E}_{q|\phi}[h(\mathbf{w})] = \nabla_\phi \mathbb{E}_{p_{\mathbf{Z}}}[h(\mathbf{g}(\mathbf{z}|\phi))]. \quad (6)$$

An alternative use for NF's is directly modelling data as a type of density esti-

mation. Here, data likelihood is

$$\ln p(\mathcal{D}|\phi) = \sum_{i=1}^{M} \ln \mathbf{g}_\star p_{\mathbf{Z}}(\mathbf{y}_i|\phi) = \sum_{i=1}^{M} \left(\ln p_{\mathbf{Z}}(\mathbf{f}(\mathbf{y}_i|\phi)) + \ln \left|\det \mathbf{D}\mathbf{f}(\mathbf{y}_i|\phi)\right|\right).$$

This model is generative using (2) and can be fitted using Maximum Likelihood Estimation (MLE).

# State of the Art

NF's build on basic probablistic rules and have been used in the current form since 2010 [KPB21], but their application to BBVI was made popular in 2015 by Rezende and Mohamed [RM15]. Here, main NF's used were called planar flows and were of the form

$$\mathbf{g}(\mathbf{z}|(\mathbf{u}, \boldsymbol{\theta}, b)) = \mathbf{z} + \mathbf{u}h\left(\boldsymbol{\theta}^T\mathbf{z} + b\right) \quad (7)$$

where $h$ is a smooth non-linearity. The term added to $\mathbf{z}$ can be considered as a single neural network unit motivating stacking this function $N$ times to get more expressiveness in the composition. These flows have the strength of linear-time determinant computation but do not have closed form inverses [RM15, Chap. 4.1]. For running VI, however, the inverse is not needed and fast computation is key.

Empirical tests were performed, modelling the posterior distribution of deep latent Gaussian models fitted to MNIST and CIFAR-10 [RM15, Chap. 6.2]. As base density, an isotropic Gaussian was used [RM15, Chap. 6.1] and NF's show competetive performance on this task with KL and $-\ln p(\mathcal{D}_{test})$ falling systematically for higher $N$ [RM15, Fig. 4, Tab. 2 and 3]. The choice of $N$ governing complexity and possibility to set $\mathbf{g}$ to match distributional assumptions were highlighted as strengths compared to e.g. mean-field fixed-form BBVI.

As each flow of the form (7) has limited expressivity, a further version similar to a neural network layer with $L$ hidden units has been proposed on the form

$$\mathbf{g}(\mathbf{z}|(\mathbf{U}, \boldsymbol{\Theta}, \mathbf{b})) = \mathbf{z} + \mathbf{U}h\left(\boldsymbol{\Theta}^T\mathbf{z} + \mathbf{b}\right) \quad (8)$$

where $\mathbf{U}, \boldsymbol{\Theta} \in \mathbb{R}^{D\times L}, \mathbf{b} \in \mathbb{R}^L$ [Ber+18, Chap. 3]. The flow was named Sylvester's flow after a determinant identity allowing the determinant computation to be efficient for low $L$ [Ber+18, Theorem 1]. Empirical results show better approximations than the planar flows on most tasks including MNIST with parameters such as $L = 16, N = 4$ compared to planar $N = 16$ [Ber+18, Tab. 3]. NF's were also compared to plain Variational Autoencoders (VAE) with fully factorized Gaussians with NF's winning on all tasks [Ber+18, Tab. 1, Tab.2].

Same year as the renewed interest in NF's for VI, Dinh, Krueger, and Ben-

gio [DKB15] presented Non-linear Independent Components Estimation (NICE), using NF's for DGM though not referring to the model as NF's. In this context, easy invertibility is required and expressivity has to be high, motivating the introduction of *coupling flows* defined as

$$\mathbf{g}(\mathbf{z}) = \mathbf{w}; \mathbf{w}_{\mathbb{I}} = \mathbf{h}(\mathbf{z}_{\mathbb{I}}|m(\mathbf{z}_{\mathbb{J}})), \mathbf{w}_{\mathbb{J}} = \mathbf{z}_{\mathbb{J}}, \quad (9)$$

where $\mathbf{w}$ is partitioned disjointyly into $(\mathbf{w}_{\mathbb{I}}, \mathbf{w}_{\mathbb{J}})$, $\mathbf{h}(\cdot, \theta)$ is a bijection and $m$ is any function, often a shallow neural network [DKB15, Chap. 3][KPB21, Chap. 3.4]. For different tasks, different partitionings can be used, possibly inducing structure such as pixel neighbourhoods [KPB21, Chap. 3.4]. Coupling flows along with autoregressive flows, introduced as Inverse Autoregressive Flows (IAF) by Kingma, Salimans, and Welling [KSW17], are State of The Art (SOTA) for NF's density estimation of many tabular datasets [KPB21, Tab. 2] and close to SOTA on image datasets [KPB21, Tab. 3].

# Open Problems

- *How to choose the base density?* Much focus in the literature is on the choice of flows $\mathbf{g}$. The choice of base density $p_{\mathbf{Z}}$ is often not analysed, usually being a standard Gaussian. For tail behaviour, the choice of base density has been discovered to impact results [Jai+19] and $p_{\mathbf{Z}}$ should possibly seen as a form of prior on modelling behaviour [KPB21, Chap. 5.1.1] and could be adapted to the task at hand.

- *How to handle discrete distributions?* To expand the use of NF's to more tasks such as Natural Language Processing (NLP), discrete target distributions should be modelled. According to Kobyzev, Prince, and Brubaker [KPB21], this is currently an open problem. Some approximations have been succesful in specific cases such as transforming discrete variables to continous by using VAE's or adding continous noise [KPB21, Chap. 5.2.2].

- *How compute efficient are flows?* For VI and most BML, computational cost of producing a posterior distribution is the largest problem. While all NF's are presented with theoretical computational considerations for the Jacobian, comparative analysis of NF's often only focuses on final approximation accuracy. A possible lack of empirical performance analyses might stem from the software implementations being newly developed and lacking maturity or adequate hardware acceleration. However with the continued development of unifying frameworks such as Pyro Normalizing Flows [Bin+18], a timed comparison might be relevant for an-

swering the question of what NF's to use when operating under a constrained compute budget.

# References


[BAK18] Danilo Bzdok, Naomi S. Altman, and Martin Krzywinski. "Points of Significance: Statistics versus machine learning". In: *Nature Methods* 15 (2018), pp. 233–234.

[Ber+18] Rianne van den Berg, Leonard Hasenclever, Jakub M. Tomczak, and Max Welling. "Sylvester Normalizing Flows for Variational Inference". In: *UAI*. 2018.

[Bin+18] Eli Bingham et al. "Pyro: Deep Universal Probabilistic Programming". In: *Journal of Machine Learning Research* (2018).

[BKM07] Vladimir Bogachev, Alexander Kolesnikov, and Kirill Medvedev. "Triangular transformations of measures". In: *Sbornik: Mathematics* 196 (Oct. 2007), p. 309. DOI: 10.1070/SM2005v196n03ABEH000882.

[BKM16] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. "Variational Inference: A Review for Statisticians". In: *Journal of the American Statistical Association* 112 (2016), pp. 859–877.

[Bre01] Leo Breiman. "Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author)". In: *Statistical Science* 16.3 (2001), pp. 199–231. DOI: 10.1214/ss/1009213726. URL: https://doi.org/10.1214/ss/1009213726.

[DKB15] Laurent Dinh, David Krueger, and Yoshua Bengio. "NICE: Non-linear Independent Components Estimation". In: *CoRR* abs/1410.8516 (2015).

[Jai+19] Priyank Jaini, Ivan Kobyzev, Marcus A. Brubaker, and Yaoliang Yu. "Tails of Triangular Flows". In: *ArXiv* abs/1907.04481 (2019).

[KPB21] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.11 (2021), pp. 3964–3979. DOI: 10.1109/TPAMI.2020.2992934.

[KSW17] Diederik P. Kingma, Tim Salimans, and Max Welling. "Improved Variational Inference with Inverse Autoregressive Flow". In: *ArXiv* abs/1606.04934 (2017).

[Par15] Roger Parloff. "Why Deep Learning Is Suddenly Changing Your Life". In: *Fortune* (Sept. 28, 2015). URL: https://fortune.com/longform/ai-artificial-intelligence-deep-machine-learning/ (visited on 06/10/2022).

[RM15] Danilo Jimenez Rezende and Shakir Mohamed. "Variational Inference with Normalizing Flows". In: *ICML*. 2015.