TRENDS AND ADVANCEMENTS IN DEEP NEURAL NETWORK COMMUNICATION

Felix Sattler[1], Thomas Wiegand[1,2], Wojciech Samek[1]
[1]Department of Video Coding & Analytics, Fraunhofer Heinrich Hertz Institute, 10587 Berlin, Germany,
[2]Department of Electrical Engineering & Computer Science, Technische Universität Berlin, 10587 Berlin, Germany

Abstract – *Due to their great performance and scalability properties neural networks have become ubiquitous building blocks of many applications. With the rise of mobile and IoT, these models now are also being increasingly applied in distributed settings, where the owners of the data are separated by limited communication channels and privacy constraints. To address the challenges of these distributed environments, a wide range of training and evaluation schemes have been developed, which require the communication of neural network parametrizations. These novel approaches, which bring the "intelligence to the data" have many advantages over traditional cloud solutions such as privacy-preservation, increased security and device autonomy, communication efficiency and high training speed. This paper gives an overview over the recent advancements and challenges in this new field of research at the intersection of machine learning and communications.*

Keywords – Neural networks, federated learning, model compression, distributed training, on-device inference.

1. INTRODUCTION

Neural networks have achieved impressive successes in a wide variety of areas of computational intelligence such as computer vision [30][82][41], natural language processing [6][42][64] and speech recognition [26] among many others and, as a result, have become a core building block of many applications.

As mobile and internet of things (IoT) devices become ubiquitous parts of our daily lives, neural networks are also being applied in more and more distributed settings. These distributed devices are getting equipped with ever more potent sensors and storage capacities and collect vast amounts of personalized data, which is highly valuable for processing in machine learning pipelines.

When it comes to processing of data from distributed sources, the *"Cloud ML"* paradigm [33] has reigned supreme in the previous decade. In Cloud ML, local user data is communicated from the often hardware constrained mobile or IoT devices to a computationally potent centralized server where it is then processed in a machine learning pipeline (e.g. a prediction is made using an existing model or the data is used to train a new model). The result of the processing operation may then be sent back to the local device. From a communication perspective, methods which follow the Cloud ML paradigm make use of centralized intelligence and

"Bring the data to the model."

While the Cloud ML paradigm is convenient for the clients from a computational perspective, as it moves all the workload for processing the data to the computationally potent server, it also has multiple severe drawbacks and limitations, which all arise from the fact that user data is processed at a centralized location:

Privacy: Data collected by mobile or IoT devices is often of private nature and thus bound to the local device. Medical data, text messages, private pictures or footage from surveillance cameras are examples for data which can not be processed in the cloud. New data protection legislations like the European GDPR [72] or the Cyber Security Law of the People's Republic of China [20] enforce strong regulations on data privacy.

Ownership: Attributing and claiming ownership is a difficult task if personal data is transfered to a central location. Cloud ML leaves users in the dark about what happens with their data or requires cumbersome rights management from the cloud service provider.

Security: With all data being stored at one central location, Cloud ML exposes a single point of failure. Multiple cases of data leakage in recent times[1] have demonstrated that the centralized processing of data comes with an unpredictable security risk for the users.

Efficiency: Transferring large records of data to a central compute node often is more expensive in terms of time and energy than the actual processing of the data. For instance, single records of medical image data can already be hundreds of Megabytes in size [70]. If the local data is large and/or the communication channels

This work was partly supported by the German Ministry for Education and Research as BIFOLD - Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A and ref. 01IS18037A).

[1] A comprehensive list of documented breaches can be found at `https://en.wikipedia.org/wiki/List_of_data_breaches`.

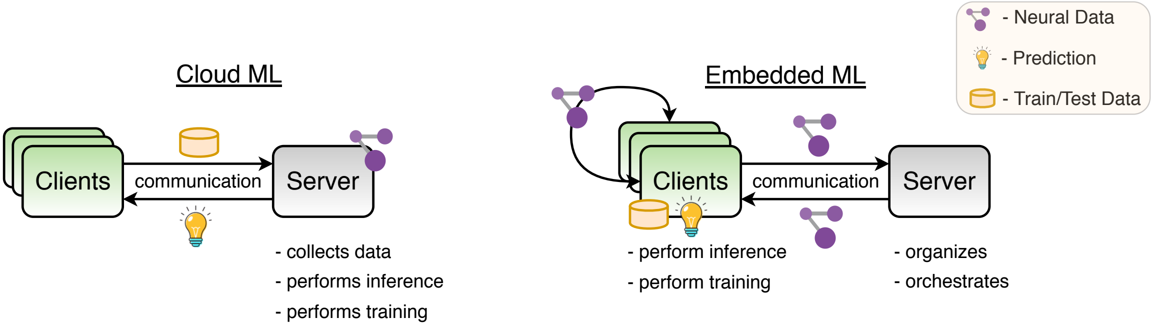

Fig. 1 – Comparison between the two paradigms for machine learning from distributed data. In cloud ml, data from users is collected and processed by a centralized service provider. In Embedded ML, data never leaves the user device. To perform inference and collaborative training, neural network parametrizations are communicated and data is processed locally.

are limited, moving data to the cloud might thus become inefficient or unfeasible.

Autonomy: Many distributed devices need to act fully autonomously and are not allowed to depend on slow and unreliable connections to a cloud server. For instance, in a self-driving car, intelligence responsible for making driving decisions needs to be available at all times and thus has to be present on-device.

As awareness for these issues increases and mobile and IoT devices are getting equipped with ever more potent hardware, a new paradigm, which we term *"Embedded ML"*, arises with the goal to keep data on device and

"Bring the model to the data."

Multi-party machine learning workflows that follow this paradigm all have one principle in common: In order to avoid the shortcomings of Cloud ML and achieve data locality they communicate neural network parametrizations ("neural data") instead of raw data. This may include not only trained neural network models, but also model updates and model gradients.

Since neural networks are typically very large, containing millions to billions of parameters [59], and mobile connections are slow, unreliable and costly the communication of neural data is typically one of the main bottlenecks in applications of Embedded ML. As a result, recently a vast amount of research has been conducted, which aims to reduce the size of neural network representations and a wide range of domain specific compression methods have been proposed.

In this work, we provide an overview on machine learning workflows which follow the Embedded ML paradigm through the unified lens of communication efficiency. We describe properties of the "neural data" communicated in Embedded ML and systematically review the current state of research in neural data compression. Finally, we also enumerate important related challenges, which need to be considered when designing efficient communication schemes for Embedded ML applications.

2. SURVEY ON NEURAL NETWORK COMMUNICATION

We currently witness the emergence of a variety of applications of Embedded ML, where neural networks are being communicated. In this section we will review the three most important settings, namely on-device inference, federated learning and peer-to-peer learning. These settings differ with respect to their communication topology, frequency of communication and network constraints. We will also review distributed training in the data center, as many methods for neural data compression have been proposed in this domain. Figure 2 illustrates the flow of (neural) data in these different settings. Table 1 summarizes the communication characteristics of the different distributed ML pipelines in further detail and gives an overview on popular compression techniques in the respective applications.

2.1 On-device Inference

Inference is the act of using a statistical model (e.g. a trained neural network) to make predictions on new data. While cloud-based inference solutions can certainly offer a variety of benefits, there still exists a wide range of applications that require quick, autonomous and failure-proof decision making, which can only be offered by on-device intelligence solutions.

For instance, in a self-driving car, intelligence responsible for making driving decisions needs to be available at all times and thus has to be present on-device. At the same time, the models used for inference might be continuously improving as new training data becomes available and thus need to be frequently communicated from the compute node to a potentially very large number of user devices. Since typical modern DNNs consists of exorbitant numbers of parameters this constant streaming of models can impose a high burden on the communication channel, potentially resulting in prohibitive delays and energy spendings.

Compression for On-Device Inference: The field of neural network compression has set out to mitigate this

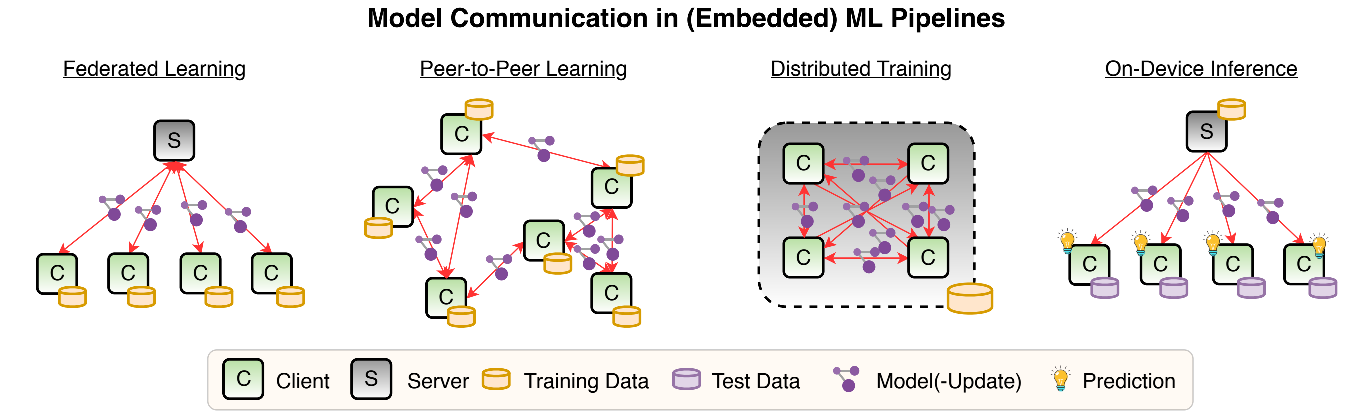

Fig. 2 – Model communication at the training and inference stages of different Embedded ML pipelines. From left to right: (1) Federated learning allows multiple clients to jointly train a neural network on their combined data, without any of the local clients having to compromise the privacy of their data. This is achieved by iteratively exchanging model updates with a centralized server. (2) In scenarios where it is undesirable to have a centralized entity coordinating the collaborative training process, *peer-to-peer learning* offers a potential solution. In peer-to-peer learning the clients directly exchange parameter updates with their neighbors according to some graph predefined topology. (3) In the data center setting, training speed can be drastically increased by splitting the workload among multiple training devices via distributed training. This however requires frequent communication of model gradients between the learner devices. (4) On-device inference protects user privacy and allows fast and autonomous predictions, but comes at the cost of communicating trained models from the server to the individual users.

problem by reducing the size of trained neural network representations. The goal in this setting is typically to find a compressed neural network representation with minimal bit-size, which achieves the same or comparable performance as the uncompressed representation. To achieve this end, a large variety of methods have been proposed which vary w.r.t. the computational effort of encoding and compression results. We want to stress that neural network compression is a very active field of research and considers issues of communication efficiency, alongside other factors such as memory- and computation complexity, energy efficiency and specialized hardware. While we only focus on the communication aspect of neural network compression, a more comprehensive survey can be found e.g. in [19].

In neural network compression it is usually assumed that the sender of the neural network has access to the entire training data and sufficient computational resources to retrain the model. By using training data during the compression process the harmful effects of compression can be alleviated. The three most popular methods for trained compression are Pruning, Distillation and trained quantization.

Pruning techniques [40][12][28][83] aim to reduce the entropy of the neural network representation by forcing a large number of elements to zero. This is achieved by modifying the training objective in order to promote sparsity. This is typically done by adding a ℓ_1 or ℓ_2 regularization penalty to the weights, but also Bayesian approaches [52] have been proposed. Pruning techniques have been shown to be able to achieve compression rates of ore than one order of magnitude, depending on the degree of overparameterization in the network [28].

Distillation methods [31] can be used to transfer the knowledge of a larger model into a considerably smaller architecture. This is achieved by using the predictions of the larger network as soft-labels for the smaller network. Trained quantization methods restrict the bitwidth of the neural network during training, e.g., reducing the precision from 32 bit to 8 bit [76]. Other methods generalize this idea and aim to directly minimize the entropy of the neural network representation during training [79]. It is important to note however, that all of these methods require re-training of the network and are thus computationally expensive and can only be applied if the full training data is available.

In situations where compression needs to be fast and/or no training data is available at the sending node, trained compression techniques can not be applied and one has to resort to ordinary lossy compression methods. Among these, (vector) quantization methods [18][17] and efficient matrix decompositions [67][86] are popular. A middle-ground between trained and ordinary lossy compression methods are methods which only require some data to guide the compression process. These approaches use different relevance measures based e.g. the diagonal of the Hessian [29], Fisher information [68] or layer-wise relevance [84][4] to determine which parameters of the network are important.

Many of the above described techniques are somewhat orthogonal and can be combined. For instance the seminal "Deep Compression" method [27] combines pruning with learned quantization and Huffman coding to achieve compression rates of up to x49 on a popular VGG model, without any loss in accuracy. More recently the DeepCABAC [78] algorithm, developed within the MPEG standardization initiative on neural network compression[2], makes use

[2] https://mpeg.chiariglione.org/standards/mpeg-7/compression-neural-networks-multimedia-content-description-and-analysis

of learned quantization and the very efficient CABAC encoder [50] to further increase the compression rate to x63.6 on the same architecture.

2.2 Federated Learning

Federated learning [51][47][37] allows multiple parties to jointly train a neural network on their combined data, without having to compromise the privacy of any of the participants. This is achieved by iterating over multiple communication rounds of the following three step protocol:

(1) The server selects subset of the entire client population to participate in this communication round and communicates a common model initialization to these clients.

(2) Next, the selected clients compute an update to the model initialization using their private local data.

(3) Finally, the participating clients communicate their model updates back to the server where they are aggregated (by e.g. an averaging operation) to create a new master model which is used as the initialization point of the next communication round.

Since private data never leaves the local devices, federated learning can provide strong privacy guarantees to the participants. These guarantees can be made rigorous by applying homomorphic encryption to the communicated parameter updates [9] or by concealing them with differentially private mechanisms [24].

Since in most federated learning applications the training data on a given client is generated based on the specific environment or usage pattern of the sensor, the distribution of data among the clients will usually be "non-iid" meaning that any particular user's local dataset will not be representative of the whole distribution. The amount of local data is also typically unbalanced among clients, since different users may make use of their device or a specific application to different extent. Many scenarios are imaginable in which the total number of clients participating in the optimization is much larger than the average number of training data examples per client. The intrinsic heterogeneity of client data in federated learning introduces new challenges when it comes to designing (communication efficient) training algorithms.

A major issue in federated learning is the massive communication overhead that arises from sending around the model updates. When naively following the federated learning protocol, every participating client has to download and upload a full model during every training iteration. Every such update is of the same size as the full model, which can be in the range of gigabytes for modern architectures with millions of parameters. At the same time, mobile connections are often slow, expensive and unreliable, aggravating the problem further.

Compression for Federated Learning: The most widely used method for reducing communication overhead in federated learning (see Table 1) is to delay synchronization by letting the clients perform multiple local updates instead of just one [38]. Experiments show that this way communication can be delayed for up to multiple local epochs without any loss in convergence speed if the clients hold iid data (meaning that all client's data was sampled independently from the same distribution) [51]. Communication delay reduces both the downstream communication from the server to the clients and the upstream communication from the clients to the server equally. It also reduces the total number of communication rounds, which is especially beneficial under the constraints of the federated setting as it mitigates the impact of network latency and allows the clients to perform computation off-line and delay communication until a fast network connection is available.

However, different recent studies show that communication delay drastically slows down convergence in non-iid settings, where the local client's data distributions are highly divergent [88][57]. Different methods have been proposed to improve communication delay in the non-iid setting, with varying success: FedProx [55] limits the divergence of the locally trained models by adding a regularization constraint. Other authors [88] propose mixing in iid public training data with every local client's data. This of course is only possible if such public data is available. The issue of heterogeneity can also be addressed with Multi-Task and Meta-Learning approaches. First steps towards an adaptive federated learning schemes have been made [56][36], but the heterogeneity issue is still largely unsolved.

Communication delay produces model-updates, which can be compressed further before communication and a variety of techniques have been proposed to this end. In this context it is important to remember the asymmetry between upstream and downstream communication during federated learning: During upstream communication, the server receives model updates from potentially a very large number of clients, which are then aggregated using e.g. an averaging operation. This averaging over the contributions from multiple clients allows for a stronger compression of every individual update. In particular, for unbiased compression techniques it follows directly from the central limit theorem, that the individual upstream updates can be made arbitrarily small, while preserving a fixed error, as long as the number of clients is large enough. Compressing the upstream is also made easier by the fact that the server is always up-to-date with the latest model, which allows the clients to send difference models instead of full models. These difference models contain less information and are thus less sensitive to

Table 1 – Communication characteristics of different Embedded ML pipelines and popular respective compression techniques used in the literature to reduce communication.

	On-Device Inference	Distributed Training	Federated Learning	Peer-to-Peer Learning
Communication:				
• Objects	trained models/ model updates	model gradients	models/ model updates	models/ model updates
• Flow	server → clients	all clients → all clients	some clients ↔ server	all clients → some clients
• Frequency	low	high	medium	high
• Redundancy	low	high	medium	low
Compression Techniques:				
• Trained Compression:				
→ Pruning	[28][73][83]	-	[44]	-
→ Trained Quantization	[76][28][79]	-	[44]	-
→ Distillation	[31]	-	-	-
• Lossy Compression:				
→ Quantization	[18][17]	[3][77][74][8]	[44][11][57]	[54] [43]
→ Sparsification	-	[48][2]	[57][44][11]	[43]
→ Sketching	-	[35]	[46]	[35]
→ Low-Rank Approx.	-	[71]	[44]	-
• Error Accumulation	-	[48][62][39]	[58]	[65]
• Communication Delay	-	[85][61][58]	[51]	[75]
• Loss-Less Compression	[78][80]	[58]	[57]	-

compression. As clients typically do not participate in every communication round, their local models are often outdated and thus sending difference models is not possible during downstream.

For the above reasons, most existing works on improving communication efficiency in federated learning only focus on the upstream communication (see Table 1). One line of research confines the parameter update space of the clients to a lower dimensional subspace, by imposing e.g. a low-rank or sparsity constraint [44]. Federated dropout [11] reduces communication in both upstream and downstream by letting clients train smaller sub-models, which are then assembled into a larger model at the server after every communication round. As the empirical benefits of training time compression seem to be limited, the majority of methods uses post-hoc compression techniques. Probabilistic quantization and sub-sampling can be used in addition to other techniques such as DeepCABAC [78] or sparse binary compression [58].

Federated Learning typically assumes a star-shape communication topology, where all clients directly communicate with the server. In some situations it might however be beneficial to consider also hierarchical communication topologies where the devices are organized at multiple levels. This communication topology naturally arises for instance in massively distributed IoT settings, where geographically proximal devices are connected to the same edge server. In these situations, hierarchical aggregation of client contributions can help to reduce the communication overhead by intelligently adapting the communication to the network constraints [49][1].

2.3 Peer-to-Peer Learning

Training with one centralized server might be undesirable in some scenarios, because it introduces a single point of failure and requires the clients to trust a centralized entity (at least to a certain degree). Fully decentralized peer-to-peer learning [69][66][7][45] overcomes these issues, as it allows clients to directly communicate with one another. In this scenario it is usually assumed that the connectivity structure between the clients is given by a connected graph. Given a certain connectivity structure between the clients, peer-to-peer learning is typically realized via a gossip communication protocol, where in each communication round all clients perform one or multiple steps of stochastic gradient descent and then average their local model with those from all their peers. Communication in peer-to-peer learning may thus be high frequent and involve a large number of clients (see Table 1). As clients typically are embodied by mobile or IoT devices which collect local data, peer-to-peer learning shares many properties and constraints of federated learning. In particular, the issues related to non-iid data discussed above apply in a similar fashion. A unique characteristic of peer-to-peer learning is that there is no central entity which orchestrates the training process. Making decisions about training related meta parameters may thus require additional consensus mechanisms, which could be realized e.g. via block chain technology [14].

Compression for Peer-to-Peer Learning: Communication efficient peer-to-peer learning of neural networks is a relatively young field of research, and thus the number of proposed compression methods is still limited. However, first promising results have already been achieved with quantization[54], sketching techniques [35] and biased compression methods in conjunction with error accumulation [43][?].

2.4 Distributed Training in the Data Center

Training modern neural network architectures with millions of parameters on huge datasets such as ImageNet can take prohibitively long time, even on the latest high-end hardware. In distributed training in the data center, the computation of stochastic mini-batch gradients is parallelized over multiple machines in order to reduce training time. In order to keep the compute devices synchronized during this process, they need to communicate their locally communicated gradient updates after every iteration, which results in very high-frequent communication of neural data. This communication is time consuming for large neural network architectures and limits the benefits of parallelization according to Amdahl's law [60].

Compression for Training in the Data-Center: A large body of research has been devoted to the development of gradient compression techniques. These methods can be roughly organized into two groups: Unbiased and biased compression methods. *Unbiased* (probabilistic) compression methods like QSGD [3], TernGrad [77] and [74] reduce the bitwidth of the gradient updates in such a way that the expected quantization error is zero. Since these methods can be easily understood within the framework of stochastic gradient based optimization, establishing convergence is straight forward. However the compression gains achievable with unbiased quantization are limited, which makes these methods unpopular in practice. *Biased* compression methods on the other hand empirically achieve much more aggressive compression rates, at the cost of inflicting a systematic error on the gradients upon quantization, which makes convergence analysis more challenging. An established technique to reduce the impact of biased compression on the convergence speed is error accumulation. In error accumulation the compute nodes keep track of all quantization errors inflicted during training and add the accumulated errors to every newly computed gradient. This way, the gradient information which would otherwise be destroyed by aggressive quantization is retained and carried over to the next iteration. In a key theoretical contribution is was shown [62][39] that the asymptotic convergence rate of SGD is preserved under the application of all compression operators which satisfy a certain contraction property. These compression operators include random sparsification [62], top-k sparsification [48], low rank approximations [71], sketching [35] and deterministic binarization methods like signSGD [8].

All these methods come with different trade-offs with respect to achievable compression rate, computational overhead of encoding and decoding and suitability for different model aggregation schemes. For instance, compression methods based on top-k sparsification with error accumulation [48] achieve impressive compression rates of more than ×500 at only marginal loss of convergence speed in terms of training iterations, however these methods also have relatively high computational overhead and do not harmonize well with all-reduce based parameter aggregation protocols [71].

The most typical connectivity structure in distributed training in the data center, is an all-to-all connection topology where all computing devices are directly connected via hard-wire. An all-to-all connection allows for efficient model update aggregation via all-reduce operations [22]. However, to efficiently make use of these primitives, compressed representations need to be summable. This property is satisfied for instance by sketches [35] and low-rank approximations [71].

3. RELATED CHALLENGES IN EMBEDDED ML

Despite the recent progress made in efficient deep neural network communication, many unresolved issues still remain. Some of the most pressing challenges for Embedded ML include:

Energy Efficiency: Since mobile and IoT devices usually have very limited computational resources, Embedded ML solutions are required to be energy efficient. Although many research works aim to reduce the complexity of models through neural architecture search [81], design energy-efficient neural network representations [80], or tailor energy-efficient hardware components [15], the energy efficiency of on-device inference is still a big challenge.

Convergence: An important theoretical concern when designing compression methods for distributed training schemes is that of convergence. While the convergence properties of vanilla stochastic gradient descent based algorithms and many of their distributed variants are well understood [10][38][45], the assumption of statistical non-iid-ness of the clients data in many Embedded ML applications still pose a set of novel challenges, especially when compression methods are used.

Privacy and Robustness: Embedded ml applications promise to preserve the privacy of the local datasets. However, multiple recent works have demonstrated that in adversarial settings information about the training data can be leaked via the parameter updates [32]. A combination of cryptographic techniques such as Secure Multi-Party Computation [25] and Trusted Execution Environments [63], as well a quantifiable privacy guarantees provided by differential privacy [23] can help to overcome these issues. However it is still unclear how these techniques can be effectively combined with methods for compressed communication and what optimal trade-offs can be made between communication-efficiency and privacy guarantees.

Since privacy guarantees conceal information about the participating clients and their data, there is also an inherent trade-off between privacy and robustness, which needs to be more thoroughly investigated. For instance, it has been shown that it is possible for an adversary to introduce hidden functionality into the jointly trained model [5] or disturb the training process [16]. Detecting these adversarial behaviors becomes much more difficult under privacy constraints. Future methods for data-local training will have to jointly address the issues of efficiency, privacy and robustness.

Synchrony: In most distributed learning schemes of Embedded ML, communication takes place at regular time intervals such that the state of the system can always be uniquely determined [13]. This has the benefit that it severely simplifies the theoretical analysis of the properties of the distributed learning system. However synchronous schemes may suffer dramatically from delayed computation in the presence of slow workers (stragglers). While countermeasures against stragglers can usually be taken (e.g. by restricting the maximum computation time per worker), in some situations it might still be beneficial to adopt a asynchronous training strategy (e.g. [53]), where parameter updates are applied to the central model directly after they arrive at the server. This approach avoids delays when the time required by workers to compute parameter updates varies heavily. The absence of a central state however makes convergence analysis far more challenging (although convergence guarantees can still be given [21]) and may cause model updates to become "stale" [87]. Since the central model may be updated an arbitrary number of times while a client is computing a model update, this update will often be out-of-date when it arrives at the server. Staleness slows down convergence, especially during the final stages of training.

Standards: To communicate neural data in an interoperable manner, standardized data formats and communication protocols are required. Currently, MPEG is working towards a new part 17 of the ISO/IEC 15938 standard, defining tools for compression of neural data for multimedia applications and representing the resulting bitstreams for efficient transport. Further steps are needed in this direction for a large-scale implementation of Embedded machine learning solutions.

4. CONCLUSION

We currently witness a convergence between the areas of machine learning and communication technology. Not only are today's algorithms used to enhance the design and management of networks and communication components [34], ML models such as deep neural networks themselves are being communicated more and more in our highly connected world. The roll-out of data-intensive 5G networks and the rise of mobile and IoT applications will further accelerate this development, and it can be predicted that neural data will soon account for a sizable portion of the traffic through global communication networks.

This paper has described the four most important settings in which deep neural networks are communicated and has discussed the respective proposed compression methods and methodological challenges. Our holistic view has revealed that these four seemingly different and independently developing fields of research have a lot in common. We therefore believe that these settings should be considered in conjunction in the future.

REFERENCES

[1] M. S. H. Abad, E. Ozfatura, D. Gunduz, and O. Ercetin. Hierarchical federated learning across heterogeneous cellular networks. *arXiv preprint arXiv:1909.02362*, 2019.

[2] A. F. Aji and K. Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.

[3] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1707–1718, 2017.

[4] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE*, 10(7):e0130140, 2015.

[5] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov. How to backdoor federated learning. *arXiv preprint arXiv:1807.00459*, 2018.

[6] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[7] A. Bellet, R. Guerraoui, M. Taziki, and M. Tommasi. Personalized and private peer-to-peer machine learning. *arXiv preprint arXiv:1705.08435*, 2017.

[8] J. Bernstein, Y.-X. Wang, K. Azizzadenesheli, and A. Anandkumar. signsgd: compressed optimisation for non-convex problems. *arXiv preprint arXiv:1802.04434*, 2018.

[9] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for federated learning on user-held data. *arXiv preprint arXiv:1611.04482*, 2016.

[10] L. Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

[11] S. Caldas, J. Konečny, H. B. McMahan, and A. Talwalkar. Expanding the reach of federated learning by reducing client resource requirements. *arXiv preprint arXiv:1812.07210*, 2018.

[12] G. Castellano, A. M. Fanelli, and M. Pelillo. An iterative pruning algorithm for feedforward neural networks. *IEEE Transactions on Neural Networks*, 8(3):519–531, 1997.

[13] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

[14] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li. When machine learning meets blockchain: A decentralized, privacy-preserving and secure design. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1178–1187. IEEE, 2018.

[15] Y. Chen, T. Chen, Z. Xu, N. Sun, and O. Temam. Diannao family: energy-efficient hardware accelerators for machine learning. *Communications of the ACM*, 59(11):105–112, 2016.

[16] Y. Chen, L. Su, and J. Xu. Distributed statistical machine learning in adversarial settings: Byzantine gradient descent. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):44, 2017.

[17] Y. Choi, M. El-Khamy, and J. Lee. Towards the limit of network quantization. *arXiv preprint arXiv:1612.01543*, 2016.

[18] Y. Choi, M. El-Khamy, and J. Lee. Universal deep neural network compression. *arXiv preprint arXiv:1802.02271*, 2018.

[19] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, pages 1–43, 2020.

[20] R. Creemers. Cybersecurity law of the people's republic of china (third reading draft). *China Copyright and Media*, 2016.

[21] C. M. De Sa, C. Zhang, K. Olukotun, and C. Ré. Taming the wild: A unified analysis of hogwild-style algorithms. In *Advances in Neural Information Processing Systems*, pages 2674–2682, 2015.

[22] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[23] C. Dwork, A. Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[24] R. C. Geyer, T. Klein, and M. Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.

[25] O. Goldreich. Secure multi-party computation. *Manuscript. Preliminary version*, 78, 1998.

[26] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[27] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[28] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[29] B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems*, pages 164–171, 1993.

[30] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[31] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[32] B. Hitaj, G. Ateniese, and F. Perez-Cruz. Deep models under the gan: information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 603–618. ACM, 2017.

[33] K. Hwang. *Cloud Computing for Machine Learning and Cognitive Applications*. MIT Press, 2017.

[34] M. Ibnkahla. Applications of neural networks to digital communications–a survey. *Signal Processing*, 80(7):1185–1215, 2000.

[35] N. Ivkin, D. Rothchild, E. Ullah, I. Stoica, R. Arora, et al. Communication-efficient distributed sgd with sketching. In *Advances in Neural Information Processing Systems*, pages 13144–13154, 2019.

[36] Y. Jiang, J. Konečnỳ, K. Rush, and S. Kannan. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[37] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

[38] M. Kamp, L. Adilova, J. Sicking, F. Hüger, P. Schlicht, T. Wirtz, and S. Wrobel. Efficient decentralized deep learning by dynamic model averaging. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 393–409. Springer, 2018.

[39] S. P. Karimireddy, Q. Rebjock, S. U. Stich, and M. Jaggi. Error feedback fixes signsgd and other gradient compression schemes. *arXiv preprint arXiv:1901.09847*, 2019.

[40] E. D. Karnin. A simple procedure for pruning back-propagation trained neural networks. *IEEE Transactions on Neural Networks*, 1(2):239–242, 1990.

[41] A. Karpathy and L. Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[42] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush. Character-aware neural language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2741–2749, 2016.

[43] A. Koloskova, S. U. Stich, and M. Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. *arXiv preprint arXiv:1902.00340*, 2019.

[44] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[45] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar. Peer-to-peer federated learning on graphs. *arXiv preprint arXiv:1901.11173*, 2019.

[46] T. Li, Z. Liu, V. Sekar, and V. Smith. Privacy for free: Communication-efficient learning with differential privacy using sketches. *arXiv preprint arXiv:1911.00972*, 2019.

[47] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873*, 2019.

[48] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.

[49] L. Liu, J. Zhang, S. Song, and K. B. Letaief. Edge-assisted hierarchical federated learning with non-iid data. *arXiv preprint arXiv:1905.06641*, 2019.

[50] D. Marpe and T. Wiegand. A highly efficient multiplication-free binary arithmetic coder and its application in video coding. In *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, volume 2, pages II–263. IEEE, 2003.

[51] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, et al. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2016.

[52] D. Molchanov, A. Ashukha, and D. Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org, 2017.

[53] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

[54] A. Reisizadeh, H. Taheri, A. Mokhtari, H. Hassani, and R. Pedarsani. Robust and communication-efficient collaborative learning. In *Advances in Neural Information Processing Systems*, pages 8386–8397, 2019.

[55] A. K. Sahu, T. Li, M. Sanjabi, M. Zaheer, A. Talwalkar, and V. Smith. On the convergence of federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.

[56] F. Sattler, K.-R. Müller, and W. Samek. Clustered federated learning: Model-agnostic distributed multi-task optimization under privacy constraints. *arXiv preprint arXiv:1910.01991*, 2019.

[57] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek. Robust and communication-efficient federated learning from non-iid data. *IEEE Transactions on Neural Networks and Learning Systems*, 2019. in press.

[58] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek. Sparse binary compression: Towards distributed deep learning with minimal communication. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[59] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro. Megatron-lm: Training multi-billion parameter language models using gpu model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[60] D. B. Skillicorn. *Foundations of Parallel Programming*. Number 6. Cambridge University Press, 2005.

[61] S. U. Stich. Local sgd converges fast and communicates little. *arXiv preprint arXiv:1805.09767*, 2018.

[62] S. U. Stich, J.-B. Cordonnier, and M. Jaggi. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems*, pages 4447–4458, 2018.

[63] P. Subramanyan, R. Sinha, I. Lebedev, S. Devadas, and S. A. Seshia. A formal foundation for secure remote execution of enclaves. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2435–2450. ACM, 2017.

[64] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[65] H. Tang, X. Lian, S. Qiu, L. Yuan, C. Zhang, T. Zhang, and J. Liu. Deepsqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression. *arXiv preprint arXiv:1907.07346*, 2019.

[66] H. Tang, X. Lian, M. Yan, C. Zhang, and J. Liu. D^2: decentralized training over decentralized data. *arXiv preprint arXiv:1803.07068*, 2018.

[67] A. Tjandra, S. Sakti, and S. Nakamura. Tensor decomposition for compressing recurrent neural network. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[68] M. Tu, V. Berisha, Y. Cao, and J.-s. Seo. Reducing the model order of deep neural networks using information theory. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 93–98. IEEE, 2016.

[69] P. Vanhaesebrouck, A. Bellet, and M. Tommasi. Decentralized collaborative learning of personalized models over networks. *arXiv preprint arXiv:1610.05202*, 2016.

[70] D. R. Varma. Managing dicom images: Tips and tricks for the radiologist. *The Indian Journal of Radiology & Imaging*, 22(1):4, 2012.

[71] T. Vogels, S. P. Karimireddy, and M. Jaggi. Powersgd: Practical low-rank gradient compression for distributed optimization. *arXiv preprint arXiv:1905.13727*, 2019.

[72] P. Voigt and A. Von dem Bussche. The eu general data protection regulation (gdpr). *A Practical Guide, 1st Ed.*, 2017.

[73] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck. Deep learning for identifying metastatic breast cancer. *arXiv preprint arXiv:1606.05718*, 2016.

[74] H. Wang, S. Sievert, S. Liu, Z. Charles, D. Papailiopoulos, and S. Wright. Atomo: Communication-efficient learning via atomic sparsification. In *Advances in Neural Information Processing Systems*, pages 9850–9861, 2018.

[75] J. Wang and G. Joshi. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. *arXiv preprint arXiv:1808.07576*, 2018.

[76] N. Wang, J. Choi, D. Brand, C.-Y. Chen, and K. Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In *Advances in Neural Information Processing Systems*, pages 7675–7684, 2018.

[77] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *arXiv preprint arXiv:1705.07878*, 2017.

[78] S. Wiedemann, H. Kirchoffer, S. Matlage, P. Haase, A. Marban, T. Marinc, D. Neumann, T. Nguyen, A. Osman, D. Marpe, et al. DeepCABAC: a universal compression algorithm for deep neural networks. *arXiv preprint arXiv:1907.11900*, 2019.

[79] S. Wiedemann, A. Marban, K.-R. Müller, and W. Samek. Entropy-constrained training of deep neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[80] S. Wiedemann, K.-R. Müller, and W. Samek. Compact and computationally efficient representation of deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(3):772–785, 2020.

[81] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[82] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[83] T.-J. Yang, Y.-H. Chen, and V. Sze. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5687–5695, 2017.

[84] S.-K. Yeom, P. Seegerer, S. Lapuschkin, S. Wiedemann, K.-R. Müller, and W. Samek. Pruning by explaining: A novel criterion for deep neural network pruning. *arXiv preprint arXiv:1912.08881*, 2019.

[85] H. Yu, S. Yang, and S. Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5693–5700, 2019.

[86] X. Yu, T. Liu, X. Wang, and D. Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379, 2017.

[87] W. Zhang, S. Gupta, X. Lian, and J. Liu. Staleness-aware async-sgd for distributed deep learning. *arXiv preprint arXiv:1511.05950*, 2015.

[88] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.