Towards Federated Learning with Attention Transfer to Mitigate System and Data Heterogeneity of Clients

Hongrui Shi
hshi21@sheffield.ac.uk
University of Sheffield

Valentin Radu
valentin.radu@sheffield.ac.uk
University of Sheffield

ABSTRACT

Federated learning is a method of training a global model on the private data of many devices. With a growing spectrum of devices, some slower than smartphones, such as IoT devices, and others faster, such as home data boxes, the standard Federated Learning (FL) method of distributing the same model to all clients is starting to break down – slow clients inevitably become strugglers. We propose a FL approach that spores different size models, each matching the computational capacity of the client system. There is still a global model, but for the edge tasks, the server trains different size student models with attention transfer, each chosen for a target client. This allows clients to perform enough local updates and still meet the round cut-off time. Client models are used as the source of attention transfer after their local update, to refine the global model on the server. We evaluate our approach on non-IID data to find that attention transfer can be paired with training on metadata brought from the client side to boost the performance of the server model even on previously unseen classes. Our FL with attention transfer opens the opportunity for smaller devices to be included in the Federated Learning training rounds and to integrate even more extreme data distributions.

CCS CONCEPTS

• **Computing methodologies → Model development and analysis**; • **General and reference → Performance**.

KEYWORDS

federated learning, attention transfer, non-IID data, heterogeneous hardware, student-teacher learning

ACM Reference Format:
Hongrui Shi and Valentin Radu. 2021. Towards Federated Learning with Attention Transfer to Mitigate System and Data Heterogeneity of Clients. In *4th International Workshop on Edge Systems, Analytics and Networking (EdgeSys'21), April 26, 2021, Online, United Kingdom.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3434770.3459739

1 INTRODUCTION

Federated Learning (FL) is seen as the best solution for training a machine learning model on private data of many devices. The wide adoption of smartphones, which generate a vast amount of data, has encourage the emergence of this privacy preserving learning method [12]. FL is used for training the Google Keyboard to perform next word prediction [5] and for other tasks in mobile applications [1]. The financial and health sectors are also considering FL as a viable solution for training machine learning models on privacy sensitive data.

In traditional machine learning, data is brought to a central computing hub for processing. By contrast, FL copies a global model to many devices for local training under the coordination of a server. Thus, client train instances of the same model without budging the data from its device. Only the parameters of updated models are transferred to the server for aggregation into a global model.

Clients selected to participate in a training round perform model updates over a few local epochs. They are expected to finish their epochs before a deadline. But as our computing ecosystem flourishes with a wider spectrum of computing devices – such as in the Internet of Things (IoT) space – system heterogeneity rifts a gap in the theoretical validation of FL, if more diverse devices are to be used as clients. Recent work have shown that FL converges on the assumption that clients perform a similar amount of local updates [15, 18]. In the current FL settings, slower clients need to perform less local updates or risk slowing down the entire training process. With more small devices producing vast amounts of data, which could be clients in FL settings, we need to design other solutions for clients to train for a similar amount of epochs on their local data and still manage to finish before a reasonable deadline.

The best solution we see fit for this problem is to adapt the amount of work each client performs based on their system computational performance. Different hardware size devices cannot train a standard global model in the same amount of time. Thus, we propose to replace the standard model distributed to clients with a set of smaller models that match the computing performance of each system. For this, we rely on the student-teacher learning setting with Attention Transfer [23]. By this method, student models are forced to mimic the attention maps of the teacher model at different levels in the network.

The important question we need to answer in order for FL with Attention Transfer to be a viable solution is: *Can student (client) models updated on client local data* ***pass their acquired knowledge*** *to the teacher (server) model?* We experiment with a combination of Attention Transfer and training on metadata to facilitate the knowledge exchange from the client models to the global (server) model.

In this paper we make the following contributions:

- We propose FL with Attention Transfer. Small client models are selected to match the computing performance of target clients. These models are trained with attention transfer

from the global model and generic data on the server. We compare our approach with related work in Section 2.
- Knowledge accumulated by the client models from the local data is aggregated in the global model. We rely on a training method combining attention transfer and training on metadata to update the global model with the knowledge gathered by the client models. The details of this training process are presented in Section 3.
- This training method is evaluated to show that smaller student models are capable of boosting the performance of the teacher model when trained both on *IID* (*Independent and Identically Distributed*) data and *non-IID* data. These experiments are presented in Section 4.

2 RELATED WORK

Federated Learning (FL). FL aims to train a global model in massively distributed networks [12] at scale [1], over heterogeneous data of many sources [9], and with different training approaches [3, 14]. Different from traditional FL, we abandon the single client model paradigm and replace this with many smaller client models designed to match the computing capability of the client.

System Heterogeneity. The difference in hardware characteristics (processor size, frequency, memory, etc.) across clients and the amount of local data for training produce large variations in the number of local updates performed by clients [19]. In the common FL setting, a deadline is imposed by the server on the time allowed for the clients to finish their local round. Those clients who manage only a few local updates or none at all in the given time are called *strugglers* and often their work is discarded. FedNova [19] aggregates even the updates from straggler by factoring in the number of local updates performed by each client for weighting their contribution to the global model update. Li et al. [8] eliminate the round deadline and allow global asynchronous updates whenever a client finishes. But clients with more computing power skew the global model towards their non-IID data.

The other problem with system heterogeneity is that it causes objective inconsistency. The federated optimisation convergence is built on the assumption that clients perform a similar amount of updates from their local data [15, 18]. The best approach to assure convergence in heterogeneous systems is to allow all clients to finish their round. Nevertheless, waiting for the slow clients can significantly increase the training time [19]. By choosing a client model size that matches the system characteristics, we make sure that clients finish their local round before a reasonable deadline.

Non-IID data. FL is expected to learn from non-IID data across devices that generate vastly different distribution data. Zhao et al. [24] show that the accuracy of federated learning reduces significantly, by up to 55% on highly skewed non-IID data, due to weight divergence within the rounds. They address the reduced accuracy by sharing a small subset of data globally to all edge devices. An accuracy increases of 30% on the CIFAR-10 dataset is achieved by sharing only 5% globally data. We adopt a similar approach for the training on the client side. Other methods such as FedProx [9], VRLSGD [11] and SCAFFOLD [6] have been designed to handle non-IID local data. But these methods either result in slower convergence or require additional communication and memory.

Training paradigms. Different training paradigms have been explored for FL to address system and data heterogeneity. Formulating the training as a multi-task learning [14] results in larger global models that can accumulate the multiple perspectives of client models trained on non-IID data as virtual tasks. Knowledge transfer is used by Liang et al. [10] to train split networks. Knowledge transfer is performed at the network separation point, running the first part of the model on the client and the final part on the server. Unlike [10], we see benefit in having the entire global model on the server. This full model can be shared and used by clients that have no local data to train a portion for their model.

Attention Transfer has been used in speech recognition to enable incremental speech models [13, 25] and in computer vision to significantly improve the performance of student models [23]. This forces the student models to mimic the attention maps of the teacher model at different levels in the network. Meta-learning [4] relies on meta-knowledge extracted as generic information across tasks to generalise for a new task. We build on this concept by extracting meta-knowledge as activation maps at different levels of the student models.

No previous work has considered attention transfer for training smaller client models. We propose a training paradigm that is efficient in distilling the knowledge learned by the client models from the local data into the larger global model. The following section presents this training process.

3 FEDERATED LEARNING WITH ATTENTION TRANSFER

We extend the framework of Federated Learning to distribute different size client models, which learn from local data and distil their acquired knowledge back to the global model. This section presents our approach.

3.1 The Training Process

Figure 1 illustrates the steps involved in the training process. As in most Federated Learning scenarios, the server holds a global model. A generic dataset is also available on the server to facilitate the training on the server side. For each client, a student model is selected such that its size matches the computing capability of the client system. Attention transfer is used to train these client models with knowledge from the global model. This is explained further in Section 3.2.

After training, the student models are transferred to the clients. There, the models are used for producing inferences and/or for further training on the local data. Section 3.3 provides more details.

The refined client model is returned to the server. In addition, we transfer some reduced representations from the client data in the form of Metadata to be used in refining the global model. More details about this process are presented in Section 3.4.

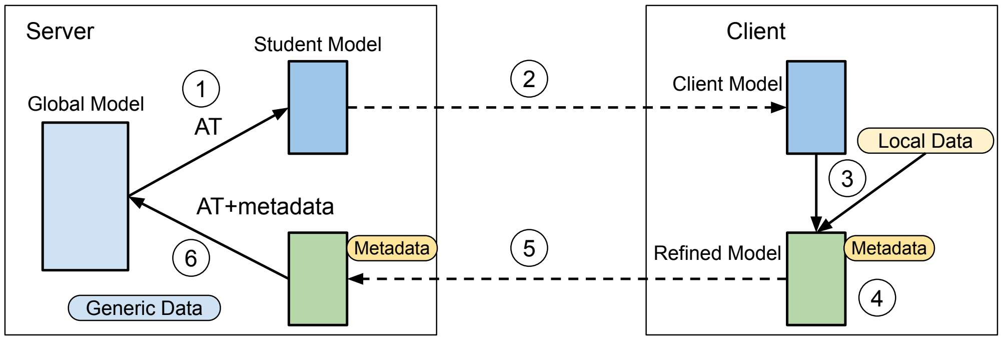

Figure 1: The training steps performed between the server and a client: (1) the global model trains a custom student model with attention transfer; (2) the model parameters are sent to the client; (3) local data is used to update the client model; (4) attention maps over the local data are extracted from a determined level in the client model and aggregated in metadata; (5) the refined model and metadata are then sent to the server to finish the round; (6) on the server, refined client models train the global model with attention transfer and metadata.

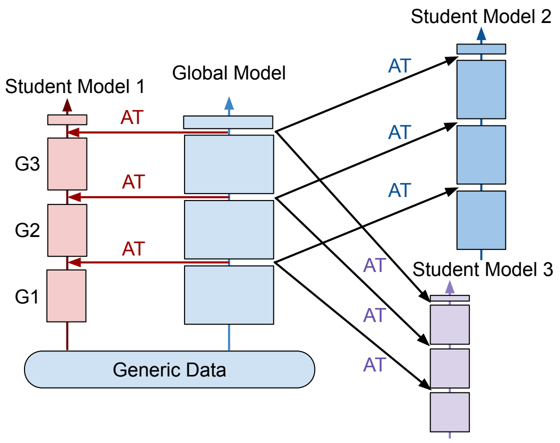

Figure 2: The global model trains different size student (client) models using Attention Transfer (AT) with the generic data from the server. Client models and global model have the same number of convolutional groups (G1, G2, G3). Convolutional groups of student models are smaller and narrower, but the activation maps between groups have the same size across models to facilitate attention transfer.

3.2 Training the Client Models with Attention Transfer

Figure 2 presents the Attention Transfer (AT) training process from the global model (M_G) to the client models (M_C) performed on the server. Each client models is selected considering the computing capability and load of the client system, such that all clients finish their local round within a given budget of time. How these client models are designed to meet the client computing requirements is outside the scope of this work. The literature reveals solutions for producing such models via hardware-aware Neural Architecture Search (NAS) [2, 16, 21], reducing the block size from teacher models [17] and specialising the computing model to match the computing budget [20]. The only consideration is that the client (student) models and the global (teacher) model have the same number of convolutional groups (G_i) and the same size of activation maps ($|A^{[i]}|$) after each group. But generally, client models have fewer and narrower layers in their convolutional groups.

AT is applied using the generic data (D_G) of the server. Each sample is propagated through the global model and the client model, with the optimiser adjusting the client weights such that it minimising the loss at the paired activation maps:

$$\mathcal{L}_{AT_{G\rightarrow C}} = \frac{\beta}{2} \sum_{j \in \mathcal{I}} \left\| \frac{Q_C^{[j]}}{\|Q_C^{[j]}\|_2} - \frac{Q_G^{[j]}}{\|Q_G^{[j]}\|_2} \right\|_2 \tag{1}$$

where $Q_C^{[j]} = vec(F(A_C^{[j]}))$ is the vectorised form of the activation map $A_C^{[j]}$ of a client model, and the respective j pair in the global model $Q_G^{[j]} = vec(F(A_G^{[j]}))$; $\mathcal{I}$ is the set of levels in the two networks from where activation maps are transferred. In Figure 2, we have $\mathcal{I} = \{G1, G2, G3\}$.

3.3 Refining the Client Model with Local Data

The trained client model M_{C_k} is transferred to the k-th client for further training on the client local data D_{C_k}. The local training refines the client model over a few epochs. The optimiser adjusts the weights $\mathbf{W}_k$ of the client model such that it minimises the loss function ℓ_k over the local data:

$$\arg\min_{\mathbf{W}_k} F(\mathbf{W}_k \boldsymbol{x}) = \frac{1}{N} \sum_{i=1}^{N} \ell_k(\mathbf{W}_k x_k^{(i)}, y_k^{(i)}) \tag{2}$$

where N is the size of D_{C_k}, and $(x_k^{(i)}, y_k^{(i)})$ is the i-th instance of input-target pairs in the local dataset.

To address the non-IID data challenge, we need to exercise the upper layers of the network for classes that may not be included in D_G. We achieve this by collecting activation maps from the upper layers of M_C. These activation maps collected from a predetermined level j in the network are aggregated and sent to the server as metadata $D_M = \bigcup_{i=1}^{N} A_i^{[j]}$.

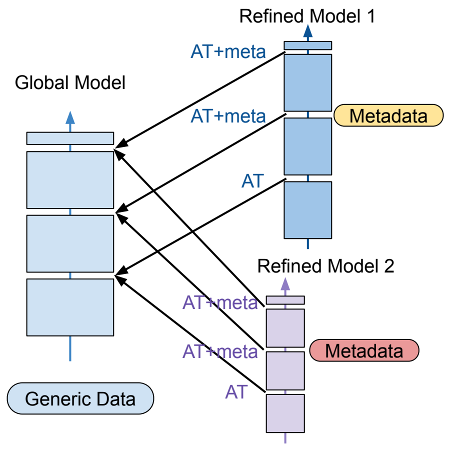

Figure 3: Client refined models together with some metadata brought from the client help to train the global model. This training involves three steps: (1) attention transfer from the client models to the global model using the generic data. The activation maps of client models are aggregated for transfer; (2) metadata is used to refine the upper layers of the global model to compensate for non-IID data of clients; (3) refining the model on the server generic data.

3.4 Refining the Global Model

Each k client sends its refined client model M_{C_k} to the server in order to update the global model M_G in the following three steps:

(1) Attention Transfer from M_{C_k} to M_G. This uses the D_G data of the server to calibrate the parameters of M_G to the aggregated scales of the M_{C_k} models, minimising the loss:

$$\mathcal{L}_{AT_{C\to G}} = \frac{\beta}{2}\frac{1}{K}\sum_{j\in\mathcal{I}}\sum_{k}\left\| \frac{Q_G^{[j]}}{\|Q_G^{[j]}\|_2} - \frac{Q_{C_k}^{[j]}}{\|Q_{C_k}^{[j]}\|_2} \right\|_2 \quad (3)$$

where K is the number of clients that finished this training round and $Q_{C_k}^{[j]} = vec(F(A_{C_k}^{[j]}))$ is the vectorised activation map of the client model M_{C_k}, at group level j in the network.

(2) Fine-tuning the upper layers of M_G using activation maps from D_{M_k}. The activation maps from D_{M_k} are propagated through M_G starting from a predefined level in $\mathcal{I}$ convolutional groups. The role of this phase is to specialise the features in the upper part of the network, including for the under-represented classes due to non-IID data distribution, but captured in D_{M_k} as higher level activation maps.

(3) Refining the lower layers with AT by freezing the upper layers previously trained on D_M. For this we reduce the size of $\mathcal{I}$ from that used in step 1 to include only the first few groups of the models $\mathcal{I}' \subset \mathcal{I}$.

The training round finishes with an updated global model M_G. The next round will rely on this M_G to update the client models.

4 EVALUATION

This section presents our preliminary results that validate our proposed approach of Federated Learning with Attention Transfer.

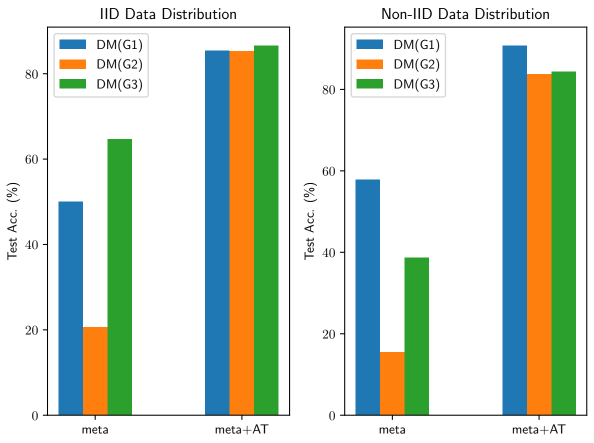

Figure 4: The effect of extracting metadata at different levels of the network is compared in term of their test accuracy yield of the global model. D_M^{G1}, D_M^{G2} and D_M^{G3} are the activation maps produced by group 1, 2 and 3 resp. in the client model. The left chart presents the IID data distribution split and the right chart presents the non-IID data distribution.

Table 1: The baseline performance of the global model M_G, trained only on the generic data in IID (D_G^v) and non-IID (D_G) data distributions from the CIFAR-10 dataset.

Model	Model Config	Training data	Test Acc.
M_G	WRN-40-1	D_G^v	85.30%
		D_G	84.53%

4.1 Experiment Setup

We aim to show that a client model trained on the local data of a client can transfer the accumulated knowledge to the global model. The global model M_G and client model M_C are based on the Wide Residual Networks architecture (WRN) [22]. The M_G model has a depth of 40 and width of 1 (WRN-40-1). For the M_C model we experiment with two reduced depth WRN networks (WRN-16-1 and WRN-10-1). We find that both produce very similar results, so we only report the results of as WRN-16-1 as M_C model. We extract activation maps from 3 equivalent positions in M_G and in M_C for attention transfer (Figure 2).

Experiments are conducted on the CIFAR-10 dataset [7]. CIFAR-10 has a total number of 60,000 images of size 32x32 pixels, with 10 classes and 6,000 images for each class. In our experiments, each class is referred to based on its label (a number between 0 to 9).

4.2 Training in IID Data

We split the training set of CIFAR-10 into generic data D_G^v, with a volume of 20% of the training set and client local data D_C holding the other 80% of the training data. Both D_G^v and D_C have a similar distribution of classes.

Figure 4 gives us a better understanding of the level in the network from where metadata needs to be extraction for M_C and where to apply the attention transfer of step (3) as presented in Section 3.4. The chart to the left in Figure 4 presents the two last phases of training the global model M_G on the IID data distribution. Training on just the generic data D_G^v, the M_G model achieves an accuracy of 85.30% on the test set. We can see that using just metadata learning is not enough to boost the performance of M_G. But applying attention transfer of step (3) takes the accuracy of M_G to 86.59% for the case of attention maps extracted at the level of group 3. The additional data on the client side improves the performance of the client model M_C to 88.18%. Although not the entire knowledge gathered on the local data is passed to the global model, our training method achieves an improvement of M_G over the baseline of 1.3%.

4.3 Training in Non-IID Data

In the second evaluation scenario, we allocate the training instances for classes 0 to 8 (45,000 images) to the generic data D_G, and the local data D_C holds training images for class 9 on top of samples from the other classes (0..8). This is intended to expose the performance of our FL training method in data heterogeneity, where some classes are unbalanced in representation across server and clients (non-IID data). Training only on D_G, the test accuracy of the M_G model is 84.53%, because it is never exposed to images of class 9.

Similar to the previous data split, we want to understand what is the best position in the networks to extract the activation maps from as metadata. The right chart of Figure 2 presents this scenario. Again, we see that using just step (2) with Metadata is not enough to boost the performance of M_G. The client model M_C trained on the local data of the client increases its performance from 83% to 91.61% after the local update on the client data. This increased performance is then transferred to M_G in the attention transfer and metadata learning process. As effect of this training, the M_G model increases its performance to 90.76% on the test dataset, an increase of 6.23% over its training on D_G only.

4.4 Training with Fractions of the Metadata Activation Maps

We continue with the non-IID data distribution as this is the more realistic scenario of what can be expected in practice. The next experiment aims to determine how much of the metadata needs to be transferred from the client to the server for achieving an efficient update of M_G on the server side.

Fractions of D_M are transferred to the server. D_M is constructed by activation maps obtained at a fix level in the client model M_C when passing instances from the local data D_C. We consider the best positing in M_C found in the previous section, after group 1.

Figure 5 presents the performance of M_G trained by M_C in the 3 steps of attention transfer and metadata on the server. We see that the best performance is achieved with the entire set of activation maps D_M brought from the client side, more than 90% on the test set. However, very good accuracy is still achieved by M_G when training with as little as 10% of the entire D_M. The evaluation point at 1% of D_M sets a noticeable drop in accuracy. Similar as before, this chart also shows that AT alone and Metadata based learning are not enough on their own. But in combination these boost the performance of M_G, showing that we can transfer knowledge from the smaller client model M_C to the global model M_G.

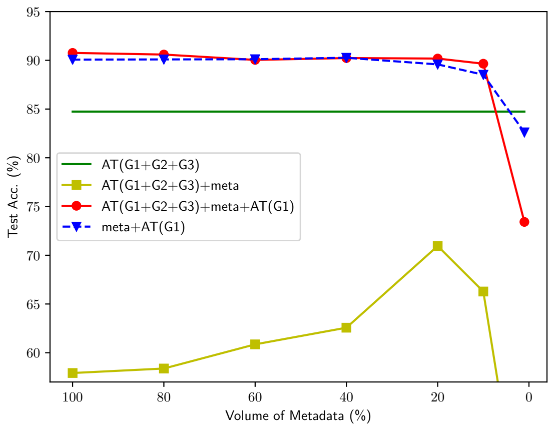

Figure 5: We compare the approaches of using AT(G1+G2+G3)+meta+AT(G1) and meta+AT(G1) in step (6) of updating the global model M_G.

Table 2: Knowledge transfer from the client model M_C to the server model M_G. This presents the performance of training on metadata after the preliminary attention transfer step. The effect of fine-tuning the upper layers of M_G is presented from each epoch over the test set and over the instances of class 9. This shows the growing capacity to recognise previously unseen instances of class 9 (in non-IID data distribution).

Model	Training data	Training Process in P6	Test Acc.	Test Acc. on class 9
M_G	D_M^{G3}	Meta 1 Epo.	84.47%	0%
		Meta 2 Epo.	84.14%	0.1%
		Meta 3 Epo.	77.01%	19.8%
		Meta 4 Epo.	36.43%	86.7%

4.5 Ablation Study on the Impact of Metadata based Learning

From the previous experiments we see that metadata has a more detrimental effect on the accuracy level of M_G when applied alone. This evaluation aims to expose the utility of using Metadata based learning in our training.

Transferred from the client side, D_M is the only representation available to the server that holds information about class 9, as chosen in our earlier data split. This information is in the form of activation maps taken from the output of group 1, based on the observations discussed above. Intuitively, the role of this training phase is to instruct M_G of how to recognise the missing class not available in the server data D_G.

Table 2 presents the performance of M_G trained first with attention transfer from M_C (step 1 of refining M_G as presented in Section 3.4), followed by metadata learning (step 2 of the training process). With each epoch, the accuracy of M_G over the test set (including all classes) decreases. However, its performance over the portion of the test set corresponding to class 9 (previously unseen in the training of M_G) increases fast. That is because the network specialises on the features characteristic to class 9 and forgets representations that are general for the rest of the classes. In this process, the upper layers of M_G produce their first representation of class 9. This knowledge remains with M_G in step 3, when AT is applied again but using the generic data D_G, which has no samples of class 9. This final step boosts back the recognition capability for the other classes with the lower layer features.

Our observation is that both AT and metadata learning are required to distill information from a small model to a larger model. The former scales the weights of the larger model in the first step and fine-tunes them in the third step, whereas the latter in step 2 adjusts the features of the upper layers of the network where representations from the missing are never formed otherwise. In the final stage (step 3), the accuracy of M_G is boosted back up on the rest of the classes with AT, and benefiting from not forgetting the upper features corresponding to the missing class.

5 CONCLUSIONS AND FUTURE WORK

We extend the Federated Learning setting by introducing different size client models that match the computing performance of each client. These are trained on the server with attention transfer from the global model. Our solution addresses the system challenge of FL by allowing more heterogeneous devices to be included in the training process without becoming straggler. We also show that our approach based on attention transfer and meta-learning mitigates the statistical challenge by transferring the client model gained knowledge to the global model in non-IID data.

In future work we will extend the evaluation to include more client models and add other compression techniques for the client models.

REFERENCES

[1] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloe Kiddon, Jakub Konečnỳ, Stefano Mazzocchi, H Brendan McMahan, et al. 2019. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046* (2019).

[2] Han Cai, Ligeng Zhu, and Song Han. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332* (2018).

[3] Chaoyang He, Murali Annavaram, and Salman Avestimehr. 2020. Group Knowledge Transfer: Federated Learning of Large CNNs at the Edge. *Advances in Neural Information Processing Systems* 33 (2020).

[4] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. 2001. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*. Springer, 87–94.

[5] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. 2019. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977* (2019).

[6] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. SCAFFOLD: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*. PMLR, 5132–5143.

[7] Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. (2009).

[8] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. 2014. Scaling distributed machine learning with the parameter server. In *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*. 583–598.

[9] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2018. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127* (2018).

[10] Paul Pu Liang, Terrance Liu, Liu Ziyin, Nicholas B Allen, Randy P Auerbach, David Brent, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Think locally, act globally: Federated learning with local and global representations. *arXiv preprint arXiv:2001.01523* (2020).

[11] Xianfeng Liang, Shuheng Shen, Jingchang Liu, Zhen Pan, Enhong Chen, and Yifei Cheng. 2019. Variance reduced local SGD with lower communication complexity. *arXiv preprint arXiv:1912.12844* (2019).

[12] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *AIStat*. PMLR.

[13] Sashi Novitasari, Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura. 2020. Sequence-to-sequence learning via attention transfer for incremental speech recognition. *arXiv preprint arXiv:2011.02127* (2020).

[14] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet Talwalkar. 2017. Federated multi-task learning. *arXiv preprint arXiv:1705.10467* (2017).

[15] Sebastian U Stich. 2018. Local SGD converges fast and communicates little. *arXiv preprint arXiv:1805.09767* (2018).

[16] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proc. CVPR*.

[17] Jack Turner, Elliot J Crowley, Michael O'Boyle, Amos Storkey, and Gavin Gray. 2019. Blockswap: Fisher-guided block substitution for network compression on a budget. *arXiv preprint arXiv:1906.04113* (2019).

[18] Jianyu Wang and Gauri Joshi. 2018. Cooperative SGD: A unified framework for the design and analysis of communication-efficient SGD algorithms. *arXiv preprint arXiv:1808.07576* (2018).

[19] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *arXiv preprint arXiv:2007.07481* (2020).

[20] Yuan Wen, Andrew Anderson, Valentin Radu, Michael FP O'Boyle, and David Gregg. 2020. TASO: Time and Space Optimization for Memory-Constrained DNN Inference. In *Proc. SBAC-PAD)*. IEEE.

[21] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. 2019. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proc. CVPR*.

[22] Sergey Zagoruyko and Nikos Komodakis. [n.d.]. *Wide Residual Networks*. Technical Report. arXiv:1605.07146v4

[23] Sergey Zagoruyko and Nikos Komodakis. 2016. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928* (2016).

[24] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. 2018. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582* (2018).

[25] Ziping Zhao, Zhongtian Bao, Zixing Zhang, Jun Deng, Nicholas Cummins, Haishuai Wang, Jianhua Tao, and Björn Schuller. 2019. Automatic assessment of depression from speech via a hierarchical attention transfer network and attention autoencoders. *IEEE Journal of Selected Topics in Signal Processing* 14, 2 (2019), 423–434.